

**Brent ByungHoon Kang
Taesoo Kim (Eds.)**

LNCS 10763

Information Security Applications

**18th International Conference, WISA 2017
Jeju Island, Korea, August 24–26, 2017
Revised Selected Papers**



Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, Lancaster, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Zurich, Switzerland

John C. Mitchell

Stanford University, Stanford, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

C. Pandu Rangan

Indian Institute of Technology Madras, Chennai, India

Bernhard Steffen

TU Dortmund University, Dortmund, Germany

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbrücken, Germany


More information about this series at <http://www.springer.com/series/7410>

Brent ByungHoon Kang · Taesoo Kim (Eds.)

Information Security Applications

18th International Conference, WISA 2017
Jeju Island, Korea, August 24–26, 2017
Revised Selected Papers

Editors

Brent ByungHoon Kang 
KAIST
Daejeon
Korea (Republic of)

Taesoo Kim
Georgia Institute of Technology
Atlanta, GA
USA

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-93562-1 ISBN 978-3-319-93563-8 (eBook)
<https://doi.org/10.1007/978-3-319-93563-8>

Library of Congress Control Number: 2018947332

LNCS Sublibrary: SL4 – Security and Cryptology

© Springer International Publishing AG, part of Springer Nature 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG
part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

The 18th World International Conference on Information Security and Application (WISA 2017) was held at Jeju Island, Korea, during August 24–26, 2017. The conference was supported by three research institutes: KISA (Korea Internet and Security Agency), ETRI (Electronics and Telecommunications Research Institute), and NSR (National Security Research Institute). Especially this year at WISA 2017, the US ONRG (Office of Naval Research Global) and ITC-PAC sponsored and also organized a Tri-Service cyber security panel to share each service's cyber security research interests and directions, provide engagement opportunities, and increase potential collaborations with international cyber communities.

The program chairs, Brent Byunghoon Kang, KAIST, and Taesoo Kim, Georgia Tech, prepared a valuable program along with the Program Committee members listed here, many of whom have served in top security conferences such as IEEE S&P, ACM CCS, Usenix Security, and NDSS. We received 59 submissions, covering all areas of information security, and finally selected 12 outstanding full papers and an additional 15 short papers out of 53 papers that were carefully reviewed by the Program Committee. The excellent arrangements for the conference venue were led by the WISA 2017 general chair, Prof. Donghoon Lee, and organizing chair, JungTaek Seo.

We were specially honored to have the two keynote talks by Professor Virgil Gligor, Carnegie Mellon University (Former Cylab Director) on “Establishing and Maintaining Root of Trust on Commodity Computer Systems,” and Dr. J. Sukarno Mertoguno, Office of Naval Research on “Autonomic Computing and Hybrid Formal-Statistical Reasoning (for Cyber Interaction and Attack).” Both talks were geared toward the roles of security in the fourth Industrial Revolution focusing on core platform systems and automated reasoning and the security protection required for millions of cyber interactions.

Many people contributed to the success of WISA 2017. We would like to express our deepest appreciation to each of the WISA Program Committee and Organizing Committee members as well as the paper contributors. Thanks to their invaluable support and sincere dedication, WISA 2017 was a success. We believe the following selected papers from WISA 2017 will contribute to new directions in cyber security, handling the core security issues of the fourth Industrial Revolution.

August 2017

Brent Byunghoon Kang
Taesoo Kim

Organization

The 18th World International Conference on Information Security Applications
August 24–26, 2017, Lotte City Hotel, Jeju Island, Korea
<http://www.wisa.or.kr>

General Chair

Donghoon Lee Korea University, Korea

Organizing Committee Chair

JungTaek Seo Soonchunhyang University, Korea

Program Committee Co-chairs

Brent ByungHoon Kang KAIST, Korea
Taesoo Kim Georgia Tech, USA

Program Committee

Amir Houmansadr	University of Massachusetts Amherst, USA
Bogdan Warinschi	University of Bristol, UK
Byoungyoung Lee	Purdue University, USA
Cristina Nita-Rotaru	Northeastern University, USA
David Lie	University of Toronto, Canada
Dooho Choi	ETRI, South Korea
Eul Gyu Im	Hanyang University, South Korea
Gang Tan	Pennsylvania State University, USA
Harry Wechsler	George Mason University, USA
Howon Kim	Pusan National University, South Korea
Huy Kang Kim	Korea University, South Korea
Jason Hong	Carnegie Mellon University, USA
Ji Sun Shin	Sejong University, South Korea
John Junghwan Rhee	NEC Laboratories America, USA
Jong Kim	POSTECH (Pohang University of Science and Technology), South Korea
Kevin Butler	University of Florida, USA
Long Lu	Stony Brook University, USA
Marcus Peinado	Microsoft Research
Min Suk Kang	National University of Singapore, Singapore
Sang Kil Cha	KAIST, South Korea

Seong-je Cho	Dankook University, South Korea
Seungwon Shin	KAIST, South Korea
Soeul Son	Google
SooHyung Kim	ETRI, South Korea
Ulrich Rührmair	University of Bochum, Germany
Virgil D. Gligor	Carnegie Mellon University, USA
Yinqian Zhang	The Ohio State University, USA

Contents

Attack and Defense I

Lightweight Fault Attack Resistance in Software Using Intra-instruction Redundancy, Revisited.	3
<i>Hwajeong Seo, Taehwan Park, Janghyun Ji, and Howon Kim</i>	
Exposing Digital Forgeries by Detecting a Contextual Violation Using Deep Neural Networks.	16
<i>Jong-Uk Hou, Han-Ul Jang, Jin-Seok Park, and Heung-Kyu Lee</i>	
Robust 3D Mesh Watermarking Scheme for an Anti-Collusion Fingerprint Code.	25
<i>Jong-Uk Hou, In-Jae Yu, Hyun-Ji Song, and Heung-Kyu Lee</i>	

Theory in Security

The Search Successive Minima Problem Is Equivalent to Its Optimization Version	39
<i>Haoyu Li and Yanbin Pan</i>	
An Improved Algorithm to Solve the Systems of Univariate Modular Equations	51
<i>Jingguo Bi, Mingqiang Wang, and Wei Wei</i>	
O ² TR: Offline Off-the-Record (OTR) Messaging.	61
<i>Mahdi Daghmehchi Firoozjaei, Sang Min Lee, and Hyounghick Kim</i>	
ARM/NEON Co-design of Multiplication/Squaring	72
<i>Hwajeong Seo, Taehwan Park, Janghyun Ji, Zhi Hu, and Howon Kim</i>	

Web Security and Emerging Technologies

WheelLogger: Driver Tracing Using Smart Watch.	87
<i>Joon Young Park, Jong Pil Yun, and Dong Hoon Lee</i>	
Evolution of Spamming Attacks on Facebook.	101
<i>Minsu Lee, Hyungu Lee, and Ji Sun Shin</i>	

Systems Security and Authentication

Model Parameter Estimation and Inference on Encrypted Domain:
Application to Noise Reduction in Encrypted Images. 115
Saetbyeol Lee and Jiwon Yoon

Breaking Text CAPTCHA by Repeated Information 127
Jae Hyeon Woo, Moosung Park, and Kyungho Lee

Automatic Mitigation of Kernel Rootkits in Cloud Environments 137
*Jonathan Grimm, Irfan Ahmed, Vassil Roussev, Manish Bhatt,
and ManPyo Hong*

Glitch Recall: A Hardware Trojan Exploiting Natural Glitches
in Logic Circuits. 150
Jungwoo Joh, Yezeo Seo, Hoon-Kyu Kim, and Taekyoung Kwon

Design and Implementation of Android Container Monitoring
Server and Agent 162
Kwon-Jin Yoon, Jaehyeon Yoon, and Souhwan Jung

Improved EM Side-Channel Authentication Using Profile-Based
XOR Model. 173
Momoka Kasuya and Kazuo Sakiyama

Holistic Tracking of Products on the Blockchain Using
NFC and Verified Users. 184
Vanesco A. J. Boehm, Jong Kim, and James Won-Ki Hong

Attack and Defense II and Network Security

Abusing TCP Retransmission for DoS Attack Inside Virtual Network 199
Son Duc Nguyen, Mamoru Mimura, and Hidema Tanaka

Improving Detection of Wi-Fi Impersonation by Fully Unsupervised
Deep Learning 212
Muhamad Erza Aminanto and Kwangjo Kim

Cyber Influence Attack: Changes in Cyber Threats Seen in the Russian
Hacking Incident. 224
Mookyu Park, Moosung Park, and Kyungho Lee

A Protection Technique for Screen Image-Based Authentication Protocols
Utilizing the SetCursorPos Function 236
Insu Oh, Kyungroul Lee, and Kangbin Yim

Crypto Protocols

A General Two-Server Cryptosystem Supporting Complex Queries 249
Sha Ma and Yunhao Ling

Efficient Software Implementation of Modular Multiplication in Prime
 Fields on TI’s DSP TMS320C6678 261
Eito Miyamoto, Takeshi Sugawara, and Kazuo Sakiyama

Key Managements of Underwater Acoustic Communication Environments . . . 274
Hyunki Kim, Jaehoon Lee, and Okyeon Yi

Parallel Implementations of SIMON and SPECK, Revisited 283
*Taehwan Park, Hwajeong Seo, Garam Lee, Md. Al-Amin Khandaker,
 Yasuyuki Nogami, and Howon Kim*

Attact Detections and Legal Aspects

Detecting Online Game Chargeback Fraud Based on Transaction Sequence
 Modeling Using Recurrent Neural Network 297
Namsup Lee, Hyunsoo Yoon, and Daeseon Choi

The Digits Hidden in the Virtual World: Approximate Estimation Applying
 Capture and Recapture. 310
Da Mi Hwang and In Seok Kim

Legal Consideration on the Use of Artificial Intelligence Technology
 and Self-regulation in Financial Sector: Focused on Robo-Advisors. 323
Keun Young Lee, Hun Yeong Kwon, and Jong In Lim

Author Index 337

Attack and Defense I



Lightweight Fault Attack Resistance in Software Using Intra-instruction Redundancy, Revisited

Hwajeong Seo¹, Taehwan Park², Janghyun Ji², and Howon Kim²(✉)

¹ IT Department, Hansung University, 116 Samseong Yoro-16gil, Seongbuk-gu, Seoul 136-792, Republic of Korea
hwajeong84@gmail.com

² School of Computer Science and Engineering, Pusan National University, San-30, Jangjeon-Dong, Geumjeong-Gu, Busan 609-735, Republic of Korea
{pth5804, jjh0819, howonkim}@pusan.ac.kr

Abstract. Fast implementations of block cipher is fundamental building block to achieve the high-speed and secure communication between IT platforms. Even though the communication is securely encrypted, the system can be exploited by malicious users if the attackers inject fault signal to the system and extract the user's secret information. For this reason, we need to ensure the high performance encryption together with secure countermeasures against side channel attacks. In this paper, we present a novel countermeasure against fault attack on Single Instruction Multiple Data (SIMD) architecture (e.g., ARM-NEON, INTEL-SSE, INTEL-AVX2). The methods achieved the fault attack resistance with intra-instruction redundancy feature in SIMD instruction set. Finally, we applied the new fault attack countermeasures on the block cipher LEA and achieved the intra-instruction redundancy and high performance over modern ARM-NEON architectures.

Keywords: Fault attack countermeasure
Intra-instruction redundancy · Block cipher · SIMD · LEA
ARM-NEON

1 Introduction

In WISA'13, Lightweight Encryption Algorithm (LEA) was announced by the Attached Institute of ETRI [6]. The LEA algorithm selected the Addition-Rotation-exclusive-or (ARX) architecture and shows high speed in software and

This work was supported by the Energy Efficiency & Resources Core Technology Program of the Korea Institute of Energy Technology Evaluation and Planning (KETEP), granted financial resource from the Ministry of Trade, Industry & Energy, Republic of Korea. (No. 20152000000170). Hwajeong Seo was supported by the ICT R&D program of MSIP/IITP. [B0717-16-0097, Development of V2X Service Integrated Security Technology for Autonomous Driving Vehicle].

hardware implementations and security against timing attacks (due to the nature of constant timing). In ICISC'13, LEA implementations on high-end ARM-NEON and General-Purpose computing on Graphics Processing Units (GPGPU) platforms are introduced [19]. The works present efficient techniques for Single Instruction Multiple Data (SIMD) and Single Instruction Multiple Thread (SIMT) based parallel computations. In [14], LEA block cipher is also evaluated in web languages such as JavaScript. The results show that the LEA implementation can achieve the high performance over web-browsers as well. In FDSE'14, Van Nguyen et al. implemented the secure Near Field Communication (NFC) protocols by using the LEA block cipher [21]. In WISA'15, LEA implementation is also evaluated on low-end 8-bit AVR processor [15]. The author presented both speed and size optimized techniques on 8-bit low-end processors and compared the performance with representative ARX block ciphers including SPECK and SIMON. The work shows that LEA is the most optimal block cipher for low-end embedded applications. In WISA'16, Seo et al. improved the parallel implementation on ARM-NEON processors [20]. The work shows that the fine combination of ARM and NEON instruction sets can further achieve the performance enhancements in parallelism. As listed above, many LEA related works proved that LEA is the most promising block cipher for both high-end computers and low-end microprocessors. However, majority of works mainly considered the high-speed implementations on the platforms, which is vulnerable to side channel attacks. In this paper, we propose new secure implementation of LEA on SIMD architecture against fault injection attacks. Several new techniques are employed and the implementations are successfully protected from fault injection attacks. Furthermore, the proposed techniques are not limited to LEA block cipher implementations. The proposed generic methods can be easily applied to the other ARX block ciphers such as SIMON and SPECK with simple modifications.

The remainder of this paper is organized as follows. In Sect. 2, we recap the fault model, fault attack countermeasures, LEA block cipher, ARM-NEON platform and previous implementations of LEA on ARM-NEON platform. In Sect. 3, we present the secure fault attack countermeasure and secure implementations of LEA on ARM-NEON platform. In Sect. 4, we evaluate the performance of proposed methods in terms of clock cycle and security. Finally, we conclude the paper in Sect. 5.

2 Related Works

2.1 Fault Attack Model

In this paper, we used the identical fault models used in previous works to evaluate our secure implementations [12]. The fault model can perform the fault injection on a cryptosystem and manipulate the instruction opcodes (i.e. instruction faults) or data (i.e. computation faults). First, the computation faults cause errors in the data that is processed by a program. The adversary can inject the

faults and the target data can be manipulated. The detailed descriptions of four computational fault models are as follows.

- **Random Word:** The adversary can target a specific word in a program and change its value into a random value unknown to the adversary.
- **Random Byte:** The adversary can target a specific word in a program and change a single byte of it into a random value unknown to the adversary.
- **Random Bit:** The adversary can target a specific word in a program and change a single bit of it into a random value unknown to the adversary.
- **Chosen Bit Pair:** The adversary can target a chosen bit pair of a specific word in a program, and change it into a random value unknown to the adversary.

Second, the instruction faults can change the program flow (the opcode of an instruction) by fault injection. The common instruction fault model is the instruction skip fault model. This model replaces the opcode of original instruction with a no-operation (`nop`) instruction. With this fault attack, the adversary can skip the execution of specific instructions in the program and change the program flow.

2.2 Previous Fault Attack Countermeasures

We can prevent the fault attack by using detection based countermeasures in software. The techniques are based on time redundancy of encryption (namely duplicate encryption). This technique figures out the fault attempts by comparing the consistency of the redundant executions or computations. To achieve the higher robustness than the duplication approach, some previous works perform the instruction triplication [1]. However, the instruction duplication and triplication cause 3.4 and 10.6 times performance overheads, respectively. Furthermore, the sophisticated fault injection setup can easily break the instruction duplication or triplication through consistent fault injection attacks. Another line of detection based fault attack countermeasures is information redundancy, which evaluates the additional check variables or parity bits. This approach detects the fault attack when the parity bits output wrong results. The techniques are generic and we can apply to general algorithms with simple modifications. However, the approach also causes 3.5–4.7 times performance overheads and cannot capture the instruction set level faults. In SAC’16, the intra-instruction redundancy based fault attack countermeasure is suggested [12]. The method implemented the redundant bit-slicing and provides the ability to detect both instruction faults and computation faults. However, the bit-slicing implementation is only efficient over certain computers with a number of general purpose registers and the modern computer architectures mainly support powerful SIMD instruction set. Under this modern computer environments, the bit-slicing method cannot be the best choice. Furthermore, previous works failed to describe how to generate the random sequences for random shuffling and perform random shuffling on the word. In this paper, we propose the efficient intra-instruction redundancy over

SIMD instruction set. This new method simultaneously generates the random sequences when the random number is needed. Finally, we applied to the LEA encryption to achieve high security against fault attacks.

Table 1. Instruction set summary for ARM-NEON

Mnemonics	Operands	Description	Cycles
VADD	Qd, Qn, Qm	Vector Addition	1
VORR	Qd, Qn, Qm	Vector OR	1
VEOR	Qd, Qn, Qm	Vector Exclusive-or	1
VSWP	Qd, Qn	Vector Re-ordering	1
VSHL	Qd, Qm, #imm	Vector Left Shift	1
VSRI	Qd, Qm, #imm	Vector Right Shift with Insert	2

2.3 Target Block Cipher: LEA

In 2013, new block cipher LEA was announced by the Attached Institute of ETRI [6]. This new algorithm has simple Addition-Rotation-eXclusive-or (ARX) and non-S-box architecture for high performance on both software and hardware environments. The LEA has 128-bit block size and the word size is 32-bit wise. Three different security levels including 128-bit, 192-bit and 256-bit are available. The number of rounds is 24, 28 and 32 for 128-, 192- and 256-bit keys, respectively. The algorithm consists of key schedule, encryption and decryption steps.

2.4 Target SIMD Platform: ARM-NEON

NEON engine is a 128-bit SIMD architecture for the ARM Cortex-A series. The difference between traditional ARM processors and new ARM-NEON processors is NEON based SIMD architecture. The NEON engine offers 128-bit wise registers and instruction sets. Each register is considered as a vector of elements of the same data type and this data type can be signed/unsigned 8-bit, 16-bit, 32-bit, or 64-bit. The detailed instructions for ARX operations are described in Table 1. A number of general arithmetic operations (e.g., Addition, logical or, logical exclusive-or) are defined. The SIMD feature provides precise operation in various vector sizes, performing multiple data in single instruction. With SIMD features, the NEON engine can accelerate data processing by at least 3X that provided by ARMv5.

The SIMD implementation can boost previous implementations by using SIMD instruction sets. In CHES'12, NEON-based cryptography implementations including Salsa20, Poly1305, Curve25519, and Ed25519 were presented [3]. To improve the performance, the authors provided novel bit-rotation, integration of multiplication and reduction operations in NEON instructions. In CT-RSA'13, ARM-NEON implementation of Grøst1 shows that 40 % performance

enhancements than the previous fastest ARM implementation [5]. In HPEC 2013, finite field multiplication is optimized by using a multiplicand reduction method. Particularly, they used ARM-NEON platform and improved the NIST curves [4]. In CHES 2014, the Curve41417 implementation adopts 2-level Karatsuba multiplication in the redundant representation [2]. In ICISC 2014, Seo et al. introduced a novel 2-way Cascade Operand Scanning (COS) multiplication for RSA implementation, which avoids the pipeline stall problems [16, 17]. In ICISC 2015, Seo et al. introduced an efficient modular multiplication and squaring operations for NIST P-521 curve, which combines Karatsuba algorithm and modular reduction efficiently [18]. Recently, Ring-LWE implementation is also accelerated by taking advantages of NEON instructions [9].

2.5 Previous Implementations of LEA on ARM-NEON

In [6], LEA implementation takes advantages of 32-bit word wise instruction sets and multiple load/store operations on ARM processor. The implementation achieved higher performance than AES implementations [10]. In [19], the first LEA implementation on NEON engine is suggested, which accelerates the performance of LEA encryption in SIMD instructions. For performance enhancements, the interdependency between each instruction set is removed and multiple operations are performed in parallel way. In [20], several new techniques to improve the LEA block cipher in parallel way further were studied. The author suggested the efficient combination of ARM and NEON instruction sets, which fully utilizes both instruction sets. Another line of study is side channel attack. In [8], Kim and Yoon performed the first experimental result of power analysis attacks on LEA implementation on FPGA boards. They perform the correlation power analysis (CPA) based on a linear relationship between measured power consumption and an intermediate value. The attack showed that the straightforward implementation of LEA is very vulnerable to CPA. In [7], Jap and Breier proposed a Differential Fault Analysis attack on LEA. By injecting random bit faults in the last round and penultimate round, they were able to recover the secret key by using 258 faulty encryptions in average. The works showed that LEA implementation should consider the countermeasures against side channel attacks. However, previous works mainly concerned on high-speed implementation rather than secure implementation. By considering the importance of secure LEA implementation, we need to further study on secure implementations of LEA. In this paper, we propose the several new techniques to ensure the high security of LEA block cipher in SIMD architecture. The proposed countermeasures are not limited to the LEA block cipher but they can be applied to the other block ciphers including SPECK and SIMON with simple modifications.

3 Proposed Method

In this paper, we propose new secure design of LEA implementation against fault injection attack. The target platform (ARM-NEON) supports 128-bit vectorized

instruction sets. We implement the LEA operations in 32×4 vector type, which performs one 32-bit wise operation for four 32-bit data sets. To ensure security against fault injection attack, we construct the intra-instruction redundancy in SIMD architecture. In the 32×4 vector type, we put four different word types including random number generator (*RNG*), redundant random number generator (*RRNG*), plaintext (*W*), and redundant plaintext (*RW*). First, the random number is generated through LEA encryption with counter mode of operation [13], performing the main LEA encryption. In every encryption, the random number is used to randomly shuffle the intermediate results in the NEON register. Second, the duplicated random number (*RRNG*) is placed in second word slot. After the encryption, the original random number and duplicated random number are compared to check whether the target platform is attacked by adversaries or not. Third, the original plaintext (*W*) is stored in one word slot. Lastly, the duplicated plaintext (*RW*) is placed in last word slot. After the encryption, the original plaintext and duplicated plaintext are compared to check whether the target platform is attacked by adversaries or not. With above data packet setting, we established the secure LEA implementation as follows.

Parameter Loading \rightarrow Message Duplication \rightarrow
 Message Shuffling #1 \rightarrow Round Function #1 $\rightarrow \dots \rightarrow$
 Message Shuffling #24 \rightarrow Round Function #24 \rightarrow Last Message Shuffling

First, we loaded all parameters including random number generator seed and plaintext from the memory to the NEON register. Among them, the loaded plaintext is duplicated and stored into redundant plaintext. In the 32-bit ARM register, we load 32-bit random number. Afterward, the message is randomly shuffled with the random number. The process also shuffles the round key, which ensures the correct round keys for encryption. When both message and round key are properly shuffled, the round function is performed. For optimized ARX operations for LEA encryption, we follow the implementation techniques covered in [20]. The combination of message shuffle and round function steps perform 24 times to complete the 128-bit LEA encryption. Lastly, the shuffled results are reordered in last message shuffle step.

The detailed message shuffle step is described in Fig. 1. First, we generate the shuffled message by using `swap` instruction. Afterward, both original and shuffled messages are masked with random number bit (*rb*). Depending on the random bit (*rb*), we generate two different mask words ($0 - rb$, $2^{32} - 1 \oplus (0 - rb)$). When the random bit is set to 1, it generates mask words ($2^{32} - 1$, 0). Otherwise, it generates opposite mask words (0, $2^{32} - 1$). After the mask operation, only one message format maintains the value. Afterward, we perform `OR` operation to choose and output one message format.

The detailed message shuffling codes are described in Algorithm 1. In Step 1, the random number (*r4*) is rotated to right by 1. This always keeps the random number bit fresh during 24 rounds. In Step 2–3, we extract the least significant bit and subtract the zero with the bit to generate the value (0 or $2^{32} - 1$). In Step 4, the output of previous step is exclusive-ored with $2^{32} - 1$. In Step 5, the

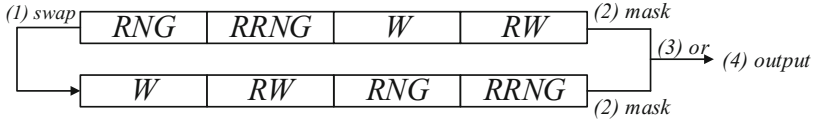


Fig. 1. Intra-instruction redundancy with random number generation and mask technique, where *RNG*: random number generator, *RRNG*: redundant random number generator, *W*: plaintext in word, *RW* redundant plaintext in word

Algorithm 1. Message shuffling in ARM-NEON, where *r4*: random number, *r5*: current state, *r8*: zero, *r9*: $2^{32} - 1$, *q0-q3*: plaintext

1: ror r4, r4, #1	14: vdup.32 q12, r6
2: lsl r6, r4, #31	15: vdup.32 q13, r7
3: sub r6, r8, r6, lsr #31	
4: eor r7, r9, r6	16: vand q0, q0, q12
5: eor r5, r5, r6	17: vand q1, q1, q12
	18: vand q2, q2, q12
6: vmov q4, q0	19: vand q3, q3, q12
7: vmov q5, q1	20: vand q4, q4, q13
8: vmov q6, q2	21: vand q5, q5, q13
9: vmov q7, q3	22: vand q6, q6, q13
	23: vand q7, q7, q13
10: vswp d0, d1	
11: vswp d2, d3	24: vorr q0, q0, q4
12: vswp d4, d5	25: vorr q1, q1, q5
13: vswp d6, d7	26: vorr q2, q2, q6
	27: vorr q3, q3, q7

current random bit is exclusive-ored with current state register (*r5*). The current state register is used to track the current state of shuffling in the message and perform the last message shuffling. In Step 6–9, the plaintext is duplicated from the NEON registers (*q0-q3*) to other registers (*q4-q7*). In Step 10–13, the lower and higher values in the register are exchanged. In Step 14–15, the mask bits (*r6*, *r7*) are duplicated to the NEON registers (*q12*, *q13*). The NEON registers can retain two states $[(0, 2^{128} - 1) \text{ or } (2^{128} - 1, 0)]$. In Step 16–23, the plaintext and shuffled plaintext are masked with mask registers. Lastly both plaintext registers are combined together in Step 24–27 to generate the output.

After 24 times of message shuffling and round function, the results are shuffled again to be aligned. In Step 1, the opposite state of current state (*r5*) is stored into register (*r7*). In Step 3–6, the plaintext is duplicated from the NEON registers (*q0-q3*) to the other registers (*q4-q7*). In Step 7–10, the lower and higher values in the register are exchanged. In Step 11–12, the mask bits (*r6*, *r7*) are duplicated to the NEON registers (*q12*, *q13*). In Step 13–20, the plaintext and swapped plaintext are masked with mask registers. Lastly both plaintext variables are added together in Step 21–24.

Algorithm 2. Last message shuffling in ARM–NEON, where $r5$: current state, $r9$: $2^{32} - 1$, $q0$ – $q3$: plaintext

```

1: eor r7, r9, r5
2: mov r6, r5
3: vmov q4, q0
4: vmov q5, q1
5: vmov q6, q2
6: vmov q7, q3
7: vswp d0, d1
8: vswp d2, d3
9: vswp d4, d5
10: vswp d6, d7
11: vdup.32 q12, r6
12: vdup.32 q13, r7
13: vand q0, q0, q12
14: vand q1, q1, q12
15: vand q2, q2, q12
16: vand q3, q3, q12
17: vand q4, q4, q13
18: vand q5, q5, q13
19: vand q6, q6, q13
20: vand q7, q7, q13
21: vorr q0, q0, q4
22: vorr q1, q1, q5
23: vorr q2, q2, q6
24: vorr q3, q3, q7

```



Fig. 2. Intra-instruction redundancy without random number generation, where W : plaintext in word, RW redundant plaintext in word

For one encryption, we need 24 random bits for 24 times of message shuffling. In our implementation, we generate 128-bit random number with 128-bit LEA implementation. One time of random number generation can cover 4 times of LEA encryption. To accelerate the performance, we perform 2-way encryption techniques. First, we generate the random number with message packet described in Fig. 1. This mode only performs one encryption per computation. Afterward, we perform the encryption only mode as described in Fig. 2. This mode performs four encryptions per computation. The message packet consists of plaintext and redundant plaintext.

The detailed descriptions of 2-way encryption are described in Algorithm 3. In Step 1, we calculate the number of messages. When the number of message is equal or larger than 4 128-bit messages, we perform one encryption with random number generator and four encryption without random number generator in Step 3–6. If the message size is not long enough, we only perform the encryption with random number generator as described in Step 7–9. Finally, in Step 10, we output the ciphertext.

4 Evaluation

For performance evaluation, we selected the ARM Cortex-A9 board, which is 32-bit processor with L1 cache and L2 cache. The board is widely used in mini computers such as PandaBoard/Odroid development platforms. Particularly, our target platform supports quad-core operated at 1.4 GHz and NEON engine. The program codes are written in assembly language and compiled with NDK android library. The performance is measured in system time function.

Algorithm 3. 2-way encryption, where ENC_1 : 1 encryption with random number generation, ENC_2 : 4 encryptions only

Require: Plaintext P , Length of plaintext L (in bytes)

Ensure: Ciphertext C

```

1:  $N \leftarrow \lceil L/16 \rceil, T \leftarrow 0$ 
2: while  $N > 0$  do
3:   if  $N \geq 5$  then
4:      $C[T] \leftarrow ENC_1(P[T])$ 
5:      $C[T + 1 : T + 4] \leftarrow ENC_2(P[T + 1 : T + 4])$ 
6:      $N \leftarrow N - 5, T \leftarrow T + 5$ 
7:   else
8:      $C[T] \leftarrow ENC_1(P[T])$ 
9:      $N \leftarrow N - 1, T \leftarrow T + 1$ 
10: return  $C$ 

```

The comparison results are drawn in Table 2. In [6], the LEA implementation on ARM instruction sets achieved the 20.06 cycle/byte. This implementation does not take advantages of NEON instruction sets. In [20], the performance is improved by taking advantages of barrel shifter instruction. The technique optimizes the rotation overheads and enhances the performance by 13.8% for ARM platform. In terms of NEON instruction, the parallel LEA implementation achieved the 10.06 cycle/byte in [19]. The works used several optimization techniques such as efficient ARX instructions in NEON and pipelined implementations. In [20], the author further improved the performance of LEA implementation. First, they suggested the compact round key access techniques. The method reduces the required registers for round keys and increases the number of plaintext for encryption in each round. Second, ordinary format is efficiently converted into SIMD format with `transpose` and `swap` instructions. With these techniques, NEON implementation is improved by 15.5%. Third, the ARM and NEON instruction sets are finely integrated together, which hides the latency for ARM instruction into NEON instruction. The results achieved the 20.4% performance enhancements than [19].

However, previous implementations only achieved the highest performance. For practical usages, the implementation should be secure against side channel attacks. In this paper, we presented new fault injection attack countermeasures

Table 2. Comparison of LEA implementations on ARM–NEON architectures in terms of execution timing (c/b: cycle/byte) and security against fault attack

Method	Speed(c/b)	Instruction	Fault Attack
Hong et al. [7]	20.1	ARM	Insecure
Seo et al. [20]	17.3	ARM	Insecure
Seo et al. [19]	10.1	NEON	Insecure
Seo et al. [20]	8.5	NEON	Insecure
Seo et al. [20]	8.0	ARM–NEON	Insecure
Proposed Method ver 1	99.0	NEON	Secure
Proposed Method ver 2	50.4	NEON	Secure

in NEON instruction set. The countermeasure adds some overheads to the implementation and we achieved 99.0 cycles/bytes for secure LEA implementations on ARM–NEON platform. For better performance, we used 2-way encryption technique, which improved the performance by 49.1 % and requires 50.4 clock/byte.

In terms of security model, we tested several different fault attack scenario studied in previous works as follows [12].

- **Random Word:** The adversary has no control on the number of faulty bits. The adversary can only create random faults in the target word (32-bit).
- **Random Byte:** The adversary can tune the fault injection to randomly affect a single byte of the 32-bit data.
- **Random Bit:** The fault injection can be tuned to affect single bit of the target word.
- **Chosen Bit Pair:** The adversary can inject faults into two chosen, adjacent bits of the target word.

In the unprotected LEA implementation, any computation or instruction fault injection attacks are easily exploited by the adversaries because this unprotected model doesn’t have any countermeasures to prevent the fault injection attacks. In the previous secure implementation by [12], they suggested the fault attack countermeasure with bit-slice technique. However, the bitslicing implementation has many limitations in practice. First, the ARX based block cipher (LEA, SIMON, and SPECK) cannot be implemented with bitslicing technique due to non-linear addition operation generating carry bits. Second, the bitslicing is efficient when the word size is small enough. In [12], they used very old 32-bit SPARC processor called LEON3. However, for modern 64-bit processors, bitslicing is not efficient because the 64 encryptions are required to perform bitslicing method and it requires a number of general purpose registers (at least 128 registers for 128-bit block size). Considering that 64-bit INTEL processors only support 16 64-bit general purpose registers, the bitslicing technique reduces the performance significantly. Instead of bitslicing, we can accelerate the performance significantly with powerful SIMD instruction set. Third, they suggested the random shuffling technique but they failed to describe how to make it work.

In the word, bit-wise shuffling imposes huge overheads without the dedicated bit-wise shuffling instruction but the target platform does not support it. On the other hand, we fully describe the efficient random shuffling technique using SIMD instruction set. The proposed technique is generic and can be used for any modern processors.

In terms of security, the bitslicing is vulnerable to fault injection attacks on chosen bit pair. If the attacker affects chosen bit pair such as plaintext and its redundant plaintext, the users cannot figure out the fault injection attacks. However, our technique is based on word wise computations so two chosen bits cannot manipulate the two different words and same bit at once.

Moreover, we support the computations from one encryption. This feature provides high scalability to utilize the encryption mode. For efficient random number generation and encryption, we also support 2-way encryption mode. For this reason, our approach is more secure and efficient than previous works on the modern computers. The detailed comparison results are given in Table 3¹.

Table 3. Security comparison of proposed method, where RNG: random number generator, RW: random word, RB: random byte, Rb: random bit, CbP: chosen bit pair, IS: instruction skip

Method	Instruction	RNG	RW	RB	Rb	CbP	IS
Seo et al. [20]	SIMD	-	-	-	-	-	-
Patrick et al. [12]	Normal	-	✓	✓	✓	-	✓
Proposed Method	SIMD	✓	✓	✓	✓	✓	✓

5 Conclusion

In this paper, we presented new secure fault attack countermeasures for LEA block cipher algorithm on representative SIMD architecture, namely ARM-NEON platform. We achieved the intra-instruction redundancy by using 128-bit wise NEON registers. This new technique successfully prevent several fault attack models that is feasible in previous works. Furthermore, we perform the 2-way encryption for high-speed encryption and random number generation. Finally, we achieved the 50.4 cycle/byte for secure LEA encryption.

The proposed methods improved the security of LEA block cipher on ARM-NEON platforms. For this reason, there are many future works remained. First, we can directly apply the fault attack countermeasures to the other ARX block ciphers such as SPECK and SIMON. Recent works by [11,20] do not consider the any secure measures proposed in this paper so we can enhance the security by applying the proposed method. Second, we only explore the ARM-NEON

¹ For reproduction of results, the source codes will be public domain in following address. (https://github.com/solowal/WISA2017_SCA). The source code is encrypted with the password (wisa2017).

platform in this paper. However, INTEL processors also provide SIMD instructions such as AVX and SSE. The functionality of SIMD instructions are very similar to NEON instructions so we can directly apply the proposed techniques to other SIMD instruction sets without difficulty.

References

1. Barengi, A., Breveglieri, L., Koren, I., Pelosi, G., Regazzoni, F.: Countermeasures against fault attacks on software implemented AES: effectiveness and cost. In: Proceedings of the 5th Workshop on Embedded Systems Security, p. 7. ACM (2010)
2. Bernstein, D.J., Chuengsatiansup, C., Lange, T., Schwabe, P.: Kummer strikes back: new DH speed records. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 317–337. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45611-8_17
3. Bernstein, D.J., Schwabe, P.: NEON crypto. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 320–339. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33027-8_19
4. Faz-Hernández, A., Longa, P., Sánchez, A.H.: Efficient and secure algorithms for GLV-based scalar multiplication and their implementation on GLV-GLS curves. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 1–27. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04852-9_1
5. Holzer-Graf, S., Krinninger, T., Pernull, M., Schläffer, M., Schwabe, P., Seywald, D., Wieser, W.: Efficient vector implementations of AES-based designs: a case study and new implementations for Grøstl. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 145–161. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36095-4_10
6. Hong, D., Lee, J.-K., Kim, D.-C., Kwon, D., Ryu, K.H., Lee, D.-G.: LEA: a 128-bit block cipher for fast encryption on common processors. In: Kim, Y., Lee, H., Perrig, A. (eds.) WISA 2013. LNCS, vol. 8267, pp. 3–27. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05149-9_1
7. Jap, D., Breier, J.: Differential fault attack on LEA. In: Khalil, I., Neuhold, E., Tjoa, A.M., Da Xu, L., You, I. (eds.) CONFENIS/ICT-EurAsia -2015. LNCS, vol. 9357, pp. 265–274. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24315-3_27
8. Kim, Y., Yoon, H.: First experimental result of power analysis attacks on a FPGA implementation of LEA. IACR Cryptology ePrint Archive 2014:999 (2014)
9. Liu, Z., Azarderakhsh, R., Kim, H., Seo, H.: Efficient software implementation of Ring-LWE encryption on IoT processors. IEEE Trans. Comput. (2017)
10. Osvik, D.A., Bos, J.W., Stefan, D., Canright, D.: Fast software AES encryption. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 75–93. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13858-4_5
11. Park, T., Seo, H., Kim, H.: Parallel implementations of SIMON and SPECK. In: 2016 International Conference on Platform Technology and Service (PlatCon), pp. 1–6. IEEE (2016)
12. Patrick, C., Yuce, B., Ghalaty, N.F., Schaumont, P.: Lightweight fault attack resistance in software using intra-instruction redundancy. In: Avanzi, R., Heys, H. (eds.) SAC 2016. LNCS, vol. 10532, pp. 231–244. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69453-5_13

13. Seo, H., Choi, J., Kim, H., Park, T., Kim, H.: Pseudo random number generator and hash function for embedded microprocessors. In: 2014 IEEE World Forum on Internet of Things (WF-IoT), pp. 37–40. IEEE (2014)
14. Seo, H., Kim, H.: Low-power encryption algorithm block cipher in JavaScript. *J. Inf. Commun. Converg. Eng.* **12**(4), 252–256 (2014)
15. Seo, H., Liu, Z., Choi, J., Park, T., Kim, H.: Compact implementations of LEA block cipher for low-end microprocessors. In: Kim, H., Choi, D. (eds.) WISA 2015. LNCS, vol. 9503, pp. 28–40. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31875-2_3
16. Seo, H., Liu, Z., Großschädl, J., Choi, J., Kim, H.: Montgomery modular multiplication on ARM-NEON revisited. In: Lee, J., Kim, J. (eds.) ICISC 2014. LNCS, vol. 8949, pp. 328–342. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15943-0_20
17. Seo, H., Liu, Z., Großschädl, J., Kim, H.: Efficient arithmetic on ARM-NEON and its application for high-speed RSA implementation. *IACR Cryptology ePrint Archive* 2015:465 (2015)
18. Seo, H., Liu, Z., Nogami, Y., Park, T., Choi, J., Zhou, L., Kim, H.: Faster ECC over $\mathbb{F}_{2^{521}-1}$ (feat. NEON). In: Kwon, S., Yun, A. (eds.) ICISC 2015. LNCS, vol. 9558, pp. 169–181. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30840-1_11
19. Seo, H., Liu, Z., Park, T., Kim, H., Lee, Y., Choi, J., Kim, H.: Parallel implementations of LEA. In: Lee, H.-S., Han, D.-G. (eds.) ICISC 2013. LNCS, vol. 8565, pp. 256–274. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12160-4_16
20. Seo, H., et al.: Parallel implementations of LEA, revisited. In: Choi, D., Guilley, S. (eds.) WISA 2016. LNCS, vol. 10144, pp. 318–330. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56549-1_27
21. Van Nguyen, H., Seo, H., Kim, H.: Prospective cryptography in NFC with the lightweight block encryption algorithm LEA. In: Dang, T.K., Wagner, R., Neuhold, E., Takizawa, M., Küng, J., Thoai, N. (eds.) FDSE 2014. LNCS, vol. 8860, pp. 191–203. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12778-1_15



Exposing Digital Forgeries by Detecting a Contextual Violation Using Deep Neural Networks

Jong-Uk Hou, Han-Ul Jang, Jin-Seok Park, and Heung-Kyu Lee(✉)

School of Computing, KAIST, Daejeon, Republic of Korea
heunglee@kaist.ac.kr

Abstract. Previous digital image forensics focused on the low-level features that include traces of the image modifying history. In this paper, we present a framework to detect the manipulation of images through a contextual violation. First, we proposed a context learning convolutional neural networks (CL-CNN) that detects the contextual violation in the image. In combination with a well-known object detector such as R-CNN, the proposed method can evaluate the contextual scores according to the combination of objects in the image. Through experiments, we showed that our method effectively detects the contextual violation in the target image.

1 Introduction

In the age of high performance digital camera and the Internet, digital images have become one of the most popular information sources. Unlike text, images remain an effective and natural communication medium for humans, for their visual aspect makes it easy to understand the content. Traditionally, there has been confidence in the integrity of visual data, such that a picture in a newspaper is commonly accepted as a certification of the news. Unfortunately, digital images are easily manipulated, especially since the advent of high-quality image-editing tools such as Adobe Photoshop and Paintshop Pro. Therefore, digital-image forensics, a practice aimed at identifying forgeries in digital images, has become an important field of research.

To cope with image forgeries, a number of forensic schemes were proposed recently. Most of the previous digital image forensics focused on the low level features that include traces of the image modifying history. They are based on detecting local inconsistencies such as resampling artifacts [14], color filter array interpolation artifacts [2], JPEG compression [3], and so on. The pixel photo response non-uniformity (PRNU) is also widely used for detecting digital image forgeries [1, 6, 7]. There are also some methods for detecting identical regions caused by copy-move forgery [8, 16].

Most of the local inconsistency based methods are vulnerable to the general image processing such as JPEG and GIF compression, white balancing, and noise addition. On the other hand, high-level features such as lighting condition [10],

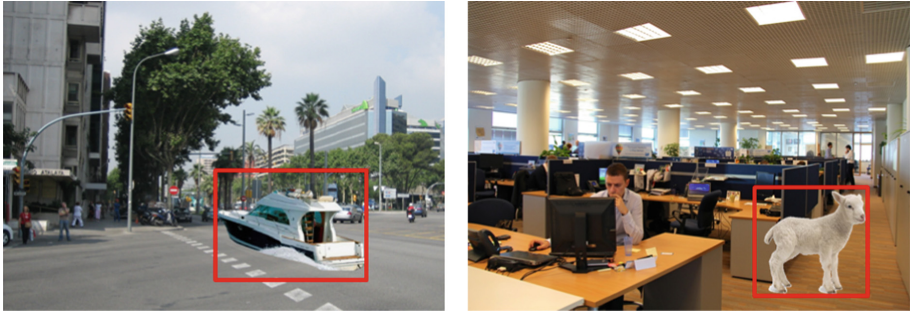


Fig. 1. Examples of image forgery detection based on contextual violation. (left) The boat was manipulated on the road, and (right) the lamb was found in the office. The manipulated parts (areas in the red box) can be detected because these objects caused contextual violations. (Color figure online)

shading and shadows [11] provide fairly robust clues for the forgery detection system against the general image processing. In this paper, we present a forensic scheme to detect the manipulation of images through contextual violation based on high-level objects in target image (See Fig. 1). Relationship information between objects in an image is used as a robust feature that is not affected by general image processing.

Our research proposes a model that can learn the context by learning the image label combination by spatial coordinate. First, we proposed a context learning convolutional neural networks (CL-CNN) that detects the contextual violation in the image. CL-CNN was trained using an annotated large image database (COCO 2014), and learned spatial context that was not provided by existing graph-based context models. In combination with a well-known object detector such as R-CNN, the proposed method can evaluate the contextual scores according to the combination of objects in the image. As a result, our method effectively detects the contextual violation in the target image.

2 Context-Learning CNN

In this research, we proposed Context-Learning convolutional neural networks (CL-CNN) to learn co-occurrence and spatial relationship between object categories. The proposed CL-CNN provides an explicit contextual model through deep learning with large image databases. CL-CNN is learned to return a high-value for the natural (or learned) combination of the object categories, while it returns low value for strange combinations or spatial contexts of the image labels.

2.1 Input Data Structure

The process of generating input data is as follows. We used an annotated image database for object location and category information. Because size of the image

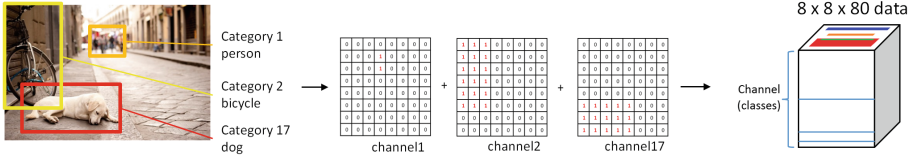


Fig. 2. Input data structure and encoding process

is too large to be used, the size is reduced to $N \times N$, where $N < \text{height and width}$ of the input image, to record the position information of the object. The channel size of the input data structure is the total number of categories to distinguish each category. The label defined part is padded with 1 value and the remaining blank area is filled with 0 values.

In this research, we reduced the size of the image to 8×8 and chose the category with 80. The final input data size is $8 \times 8 \times 80$ and the generation process is shown in Fig. 2.

2.2 CL-CNN Structure

The structure of CL-CNN is as follows (See Fig. 3). It receives input data of $8 \times 8 \times 80$, passes through two convolutional layers, and passes through three fully connected layer. Then the fully connected layer finally outputs 2×1 vector. The first value of the output is a score that evaluates how natural the label is in combination with the category and spatial context, and the second value is the score that evaluates how the category combination and spatial context of the label are awkward. The loss function used Euclidian loss L which is defined by

$$L = \frac{1}{2} \sum_i (y_i - a_i)^2, \quad (1)$$

where y is the output of the CL-CNN, and a is the label of the data sample.

2.3 Dataset Generation

In order to learn the proposed network, it is necessary to acquire a large amount of datasets. We need to have both a collection of natural labels and a collection of unnatural labels. Moreover, we also need data that shows both the location and type of the object. A dataset that meets these criteria is Microsoft COCO: Common Objects in Context [13]. Microsoft COCO 2014 provides 82,783 training image sets and label information, and 40,504 validation images and label information. This can aid in learning detailed object models capable of precise 2D localization and contextual information.

Before we use the dataset, we excluded single-category images since they are useless for learning contextual information. Thus, we used 65,268 multi-category images for learning of the CL-CNN. The categories are divided into 80 categories.

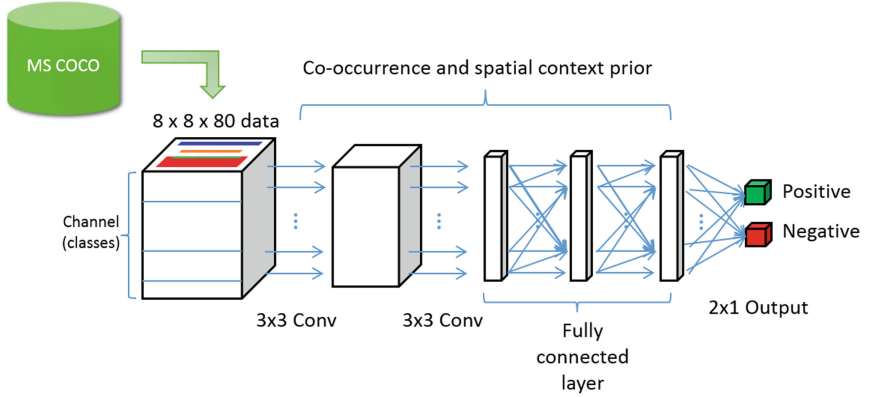


Fig. 3. Overall structure of the context-learning convolutional neural networks

A positive set was constructed using label information of t multi-category images, as the way described in Sect. 2.1.

We cannot use negative sets based on the existing databases because they need to learn combinations of unnatural labels that do not actually exist. For this reason, we generated the negative set in two ways. Negative set 1 was created by changing the size and position of the object while maintaining the category combination. Negative set 2 was created by selecting the combination of the less correlated categories. Figure 4 shows the histogram of the co-occurrences between object categories. Using the probability $P(c_1, c_2)$ from the co-occurrence histogram, combinations of classes c_1, c_2 , which has which has a low co-occurrence probability $P(c_1, c_2)$, was selected to generate a negative dataset. Next, the negative set 2 was modified by changing the size and position of the object while maintaining the category combination.

2.4 Network Training

With a simple learning approach, object combination and location shuffled dataset are trained at the same time. Next, we tested ‘combination and location shuffling’ and ‘location shuffling’ and obtained the accuracy of 0.97 and 0.53, respectively. When learning ‘combination and location shuffling’ at the same time, ‘combination change’ was strongly learned. As a result, learning of the spatial context was ignored by the over-fitted CL-CNN. Therefore, we need to improve the learning method so that the ‘location shuffling’ was learned enough.

Therefore, we trained CL-CNN by learning ‘location shuffling’ of object first, and then fine-tuning the ‘combination and location shuffling’ part sequentially. We set the learning rate to 0.001 for ‘location shuffling’ and set the learning rate to 0.00001 for ‘combination and location shuffling’ learning. We also tested ‘combination and location shuffling’ and ‘location shuffling’ and obtained an accuracy of 0.93 and 0.81, respectively. The test accuracy for ‘location shuffling’

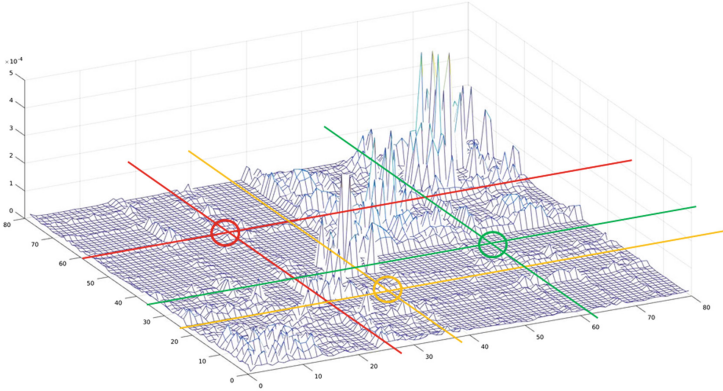


Fig. 4. Histogram of the co-occurrences between object categories. Negative set was generated by selecting the combinations of the less correlated categories (e.g. combinations shown in circled regions)

has been greatly improved from 0.53 to 0.81 when compared with the way of learning at the same time.

3 Detection of Contextual Violation of Target Image

We propose a method to detect the contextual violation of the target image using CL-CNN. The proposed method operates by combining with the output of the existing object detector such as [4, 5, 15]. Among the above methods, we solved the object detection task based on Faster R-CNN [15]. Using object detection results and probability values, we proposed a system that detects objects that are most inappropriate in the image context. The proposed method is described as follows.

Step 1. Extract objects from the suspicious image: Let I be a suspicious image. Using the image object detector, we extract the area of the object in the image and calculate the category score in each area as follows:

$$P_{r_i}(c) = F(I), \quad (2)$$

where the function $F(\cdot)$ is the region-based object detector such as Faster R-CNN [15] for a single input image I . $P \in [0, 1]^{\mathbf{R} \times \mathbf{C}}$ is the probability of the each object class c from the detected region r_i . Figure 5 shows sample of the object detection result and its details (Fig. 6).

Step 2. Generate input sets for CL-CNN: After extracting objects from the image, candidates for the contextual violation check were selected by:

$$\mathbb{P} = \{(r_i, c) : P_{r_i}(c) > \tau_i\} \quad (3)$$

where τ_i is selection threshold for the raw output. If $P_{r_i} < \tau_i$, the corresponding object region r_i is not used. For example, when $\tau_i = 0.7$, three candidates: lamb,

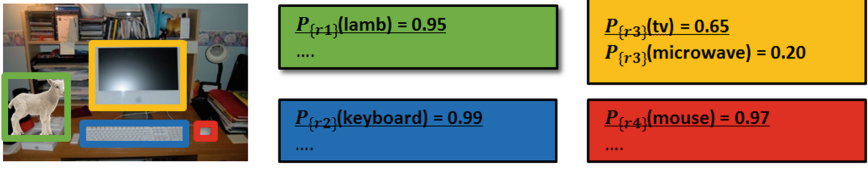


Fig. 5. Step 1. Extract object region and calculate category score using the object detector such as Faster R-CNN [15].

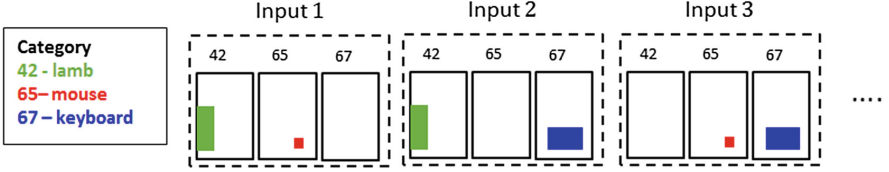


Fig. 6. Generate input sets for CL-CNN

keyboard, and mouse, are selected in the sample image in Fig. 5. Then, input sets \mathbb{S}_i for CL-CNN is generated by:

$$\mathbb{S}_i = \mathbb{P} \setminus \{(r_i, c)\} \quad (4)$$

where $\mathbb{P} \setminus \{x\}$ denotes the set \mathbb{P} excluding the element x .

Step 3. Evaluate context score of the inputs: Each input sets \mathbb{S}_i is passed to CL-CNN to generate a result value vector.

$$\hat{i} = \underset{i}{\operatorname{argmax}} [C(\mathbb{S}_i)], \quad (5)$$

where return value of the function $C(\cdot)$ denotes the positive output value of the CL-CNN. Before calculating $C(\cdot)$, the input \mathbb{S}_i is converted according to the input data structure described in Sect. 2.1. Since \mathbb{S}_i is the set \mathbb{P} excluding the element r_i , the object class c from the region $r_{\hat{i}}$ is the most unlikely object in the context of the target image I . Therefore, \hat{i} indicates the index value of the region that may cause the contextual violation.

In addition, we should consider the case where there is no contextual violation in the suspicious image. In order to reduce false positive error, we should check whether $C(\mathbb{S}_{\hat{i}})$ value is larger than user defined threshold τ_o .

$$\begin{cases} \text{Forgery detected:} & \text{if } C(\mathbb{S}_{\hat{i}}) > \tau_o, \\ \text{No detection:} & \text{otherwise.} \end{cases} \quad (6)$$

We use value 0.8 for τ_o . In addition, there may be cases where multiple objects are forged. In this case, we can solve the above problem by checking at the top n results of the Eq. (5).

4 Experimental Results

For the experiment, we used sample images collected from the Microsoft COCO: Common Objects in Context [13] as described in Sect. 2.3. The implementation of CL-CNN is based on *Caffe library* [9].

Experimental results for natural image (positive set) and forged image (negative set) are shown in Figs. 3 and 4. The natural images are from COCO 2014 test database. Since no manipulation was detected, we showed output value with the CL-CNN input contained all the object sets \mathcal{P} extracted by the detector. For natural image, the average output value was 0.98 or higher. For instance, a combination of vases, indoor tables and chairs is the frequently observed combination in the COCO dataset, so all of them are judged as natural objects as shown in Fig. 7(a). Note that, the appearance rate of people in the training dataset was high, so that an image with some people is tended to evaluate positively by CL-CNN. Therefore, we can see that the output values are somewhat large as shown in Fig. 7(b) and (c).

On the other hand, we made some forged images with arbitrary combination of object classes (See Fig. 4). In Fig. 8(a), a manipulated boat with surrounding cars and trees causes contextual violation. We confirm that the ‘naturalness of the image’ is better when the boat is removed.

However, our framework has some limitations. For example, a cow is manipulated beside a kite in Fig. 8(c). However, in our method, the information that

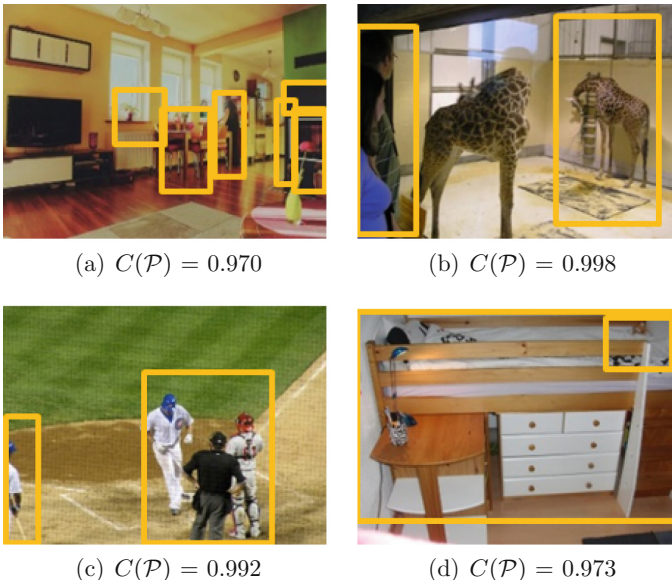


Fig. 7. CL-CNN results with natural image input. The average output value was 0.98 or higher.

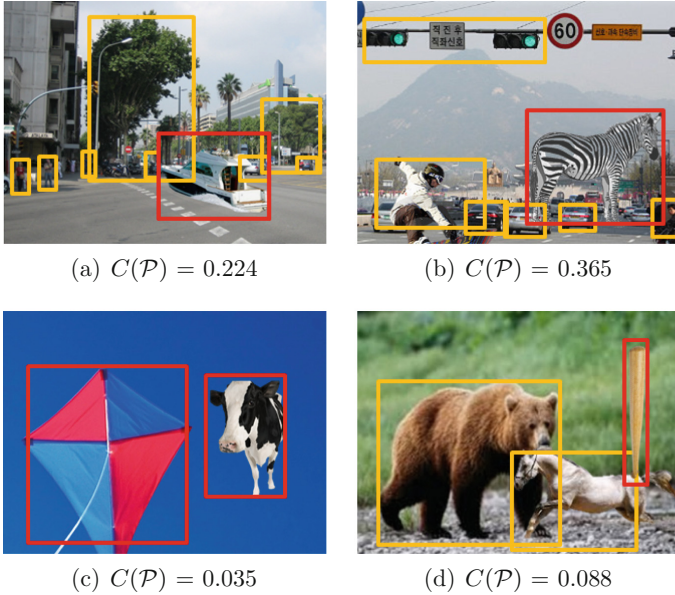


Fig. 8. CL-CNN and forgery detection results with manipulated images. The object in the red box was detected as a manipulated object. (Color figure online)

a cow is in the sky is lost during the object detection step. Therefore, both cow and kite are judged to be inappropriate for the image. In Fig. 8(d), horse and baseball bat were manipulated, but only the baseball bat was selected as the awkward object. In this case, other forgery detector such as [2, 3, 14] combined with our method to improve the detection accuracy.

5 Conclusion

In this study, we proposed a model (CL-CNN) that can provide a contextual prior by directly learning the combination of image labels. The trained model provides contextual prior based on convolutional neural networks. In combination with a well-known object detector such as R-CNN [5], the proposed method can evaluate contextual scores according to the combination of objects in the image.

However, the region-based object detector used in this study ignores the background parts, so the context between the object and background cannot be directly evaluated. Therefore, we plan to enhance the accuracy of the study by combining the deep learning based on scene classification such as [12, 17]. In addition, we will improve the model to give more robust and accurate results, with a bit more attention to the generation of negative sets.

Acknowledgement. This work was supported by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korean government (MSIP) (2017-0-01671, Development of high reliability elementary image authentication technology for smart media environment).

References

1. Chen, M., Fridrich, J., Goljan, M., Lukas, J.: Determining image origin and integrity using sensor noise. *IEEE Trans. Inf. Forensics Secur.* **3**(1), 74–90 (2008)
2. Choi, C.H., Lee, H.Y., Lee, H.K.: Estimation of color modification in digital images by CFA pattern change. *Forensic Sci. Int.* **226**, 94–105 (2013)
3. Farid, H.: Exposing digital forgeries from JPEG ghosts. *IEEE Trans. Inf. Forensics Secur.* **4**(1), 154–160 (2009)
4. Girshick, R.: Fast R-CNN. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448 (2015)
5. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587 (2014)
6. Hou, J.U., Jang, H.U., Lee, H.K.: Hue modification estimation using sensor pattern noise. In: *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 5287–5291, October 2014
7. Hou, J.U., Lee, H.K.: Detection of hue modification using photo response non-uniformity. *IEEE Transactions on Circuits and Systems for Video Technology* (2016)
8. Huang, H., Guo, W., Zhang, Y.: Detection of copy-move forgery in digital images using SIFT algorithm. In: *Pacific-Asia Workshop on Computational Intelligence and Industrial Application, PACIIA 2008*, vol. 2, pp. 272–276, December 2008
9. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093* (2014)
10. Johnson, M.K., Farid, H.: Exposing digital forgeries in complex lighting environments. *IEEE Trans. Inf. Forensics Secur.* **2**(3), 450–461 (2007)
11. Kee, E., O’Brien, J.F., Farid, H.: Exposing photo manipulation from shading and shadows. *ACM Trans. Graph.* **33**(5), Article No. 165 (2014)
12. Lin, D., Lu, C., Liao, R., Jia, J.: Learning important spatial pooling regions for scene classification. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3726–3733 (2014)
13. Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) *ECCV 2014*. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
14. Popescu, A., Farid, H.: Exposing digital forgeries by detecting traces of resampling. *IEEE Trans. Sig. Process.* **53**(2), 758–767 (2005)
15. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: *Advances in Neural Information Processing Systems*, pp. 91–99 (2015)
16. Ryu, S.J., Kirchner, M., Lee, M.J., Lee, H.K.: Rotation invariant localization of duplicated image regions based on Zernike moments. *IEEE Trans. Inf. Forensics Secur.* **8**(8), 1355–1370 (2013)
17. Zhang, F., Du, B., Zhang, L.: Saliency-guided unsupervised feature learning for scene classification. *IEEE Trans. Geosci. Remote Sens.* **53**(4), 2175–2184 (2015)



Robust 3D Mesh Watermarking Scheme for an Anti-Collusion Fingerprint Code

Jong-Uk Hou, In-Jae Yu, Hyun-Ji Song, and Heung-Kyu Lee(✉)

School of Computing, KAIST, Daejeon, Republic of Korea
heunglee@kaist.ac.kr

Abstract. Collusion attack is one of the techniques used for unauthorized removal of embedded marks. In this paper, we propose a robust 3D mesh fingerprinting scheme for an anti-collusion code. In contrast to the existing robust mesh watermarking which provides unsuitable primitives for anti-collusion code, the proposed method has well-operated capacity to carry the anti-collusion fingerprint code. In order to minimize the detection error, we also modeled the response of the detector and herein present optimized thresholds for our method. Based on the experiments, the proposed method outperformed conventional robust mesh watermarking against collusion attack in all cases.

1 Introduction

Recently, the demand for 3D mesh watermarking has increased due to the emergence of low-cost 3D printers. On the other hand, copyright issues will inevitably occur with the expansion of 3D model sharing platforms such as Thingiverse, Pinshape, and Sketchfab. To cope with the current situation, digital watermark can be used to determine authorship when a copyright dispute occurs [8], and can be used as a fingerprint to track a distribution path when a prototype in the hands of only a few people is leaked. Digital watermarking is the process of hiding digital information in a noise-tolerant signal such as multimedia data. Furthermore, digital watermarking could be utilized as an active component of an automatic system to regulate unauthorized users in a content sharing environment. For this purpose, the 3D model watermark should be covertly embedded into the 3D model content before distribution. In addition, the embedded watermark has to resist possible attempts to infringe the copyright.

Collusion attack is one of the main techniques used for unauthorized removal of embedded marks [3]. Attackers can exploit different versions of the fingerprinted content to estimate the original content. Moreover, a simple averaging process of some different protected contents can be used to disable detection of the embedded pattern. To cope with collusion attacks, two categories of countermeasure techniques were previously researched. The first technique works by pre-warping at the signal-processing level, so that each copy of the multimedia content is varied in a slightly different way [4, 17]. However, in the case that the method fails to protect against collusion attack, it cannot provide information

useful for tracking the attackers. The second countermeasures are based on the averaging-resilient fingerprint, which theoretically guarantees robustness against a number of attackers [3, 9, 14, 16, 20]. In contrast to pre-warping based methods, the anti-collusion based techniques work passively by providing explicit clues for tracking the attackers.

In this paper, we report our design of a mesh watermarking algorithm based on a mesh Laplacian matrix [2]. We used two anti-collusion fingerprint code system: one is based on the group-divisible partially balanced incomplete block design (GD-PBIBD) [9], and another one is based on Tardos’s fingerprint code [15]. By modeling the detector response of the proposed method, we theoretically determined a threshold to minimize the detection error. As a result of our tests, we experimentally demonstrate that the proposed method does not lose recipient information during collusion attack.

The rest of this paper is organized as follows. Section 2 explains the details of the proposed method. To demonstrate the performance of the proposed scheme, we tested our method with mesh benchmark sets, and report the results in Sect. 3. In Sect. 4, the conclusions are presented.

2 Proposed Method

The main assumption for designing the anti-collusion fingerprint codes is that the averaging attack of the anti-collusion code has the same effect as a bitwise AND operation of every bit position. Therefore, the embedded bit information has to be zero for the averaging attack when each bit from each colluded work is different. However, conventional robust mesh watermarking methods such as [1, 5, 8, 12, 19] did not consider this aspect. Moreover, anti-collusion fingerprint code requires relatively a number of payloads in many cases, so that the existing mesh watermarks are not suitable to carry anti-collusion fingerprint codes. In contrast to the existing robust mesh watermarking, our proposed method has sufficient capacity to carry the anti-collusion fingerprint code.

Unlike the image and video watermarking described in [10, 16], each vertex of a 3D mesh model is non-uniformly sampled, so that the spatial structure of the mesh is not suitable for utilizing fingerprint codes. Alternatively, we can divide the mesh into disjoint bands in the frequency domain and embed a watermark in them. To perform the frequency analysis, we use a technique called *mesh spectral analysis* [11] that can be considered a principal component analysis of the shape. We now describe each step of watermark embedding and detection in detail.

2.1 Watermark Embedding Algorithm

The spectrum of a 3D mesh is computed from connectivity and coordinates of vertices of the mesh. Eigenvalue decomposition of a mesh Laplacian matrix is required to produce spectral coefficients. For the Laplacian matrix, we employed a mesh Laplacian matrix proposed by Bollobás [2], referred to as the Kirchhoff matrix. The Kirchhoff matrix \mathbf{K} is defined as follows:

$$\mathbf{K} = \mathbf{D} - \mathbf{A}, \quad (1)$$

where \mathbf{D} is a diagonal matrix of which an element is defined by a degree of each vertex, and \mathbf{A} is an adjacent matrix of the mesh of which elements $a_{i,j}$ are equal to '1' if vertices i and j are adjacent, and otherwise '0'. A mesh \mathbb{M} with m vertices produces a Kirchhoff matrix \mathbf{K} of size $m \times m$, and its eigenvalue decomposition may be described as follows:

$$\mathbf{K} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T, \quad (2)$$

where $\mathbf{\Lambda}$ is a diagonal matrix composed of the m eigenvalues λ_i in ascending order, and \mathbf{U} is a unitary matrix composed of the m -dimensional normalized eigenvectors.

The Laplacian eigenvector basis is used to perform an orthogonal decomposition of a function. The spectra of the mesh are obtained by projection of the vector according to the Laplacian basis, as indicated in the following equation.

$$\mathbf{c} = \mathbf{U}^T \mathbf{x}, \quad (3)$$

where $\mathbf{c} = (c_1, c_2, \dots, c_m)^T$ is a spectral coefficient of input mesh, and \mathbf{x} is a vector of the vertices of the mesh.

We generate a watermark pattern using a 'group-divisible partially balanced incomplete block design' (GD-PBIBD) [7]. Using a generator proposed by InKoo et al. [9], the generated code set is defined as an $N \times R$ binary matrix \mathbf{N} in which a column represents a unique fingerprint code for each user, and where N is the length of each fingerprint code. The generated fingerprint code set \mathbf{N} for $R = s^{2(p-1)}$ recipients can protect cover work from n attackers ($=s - 1$). Detailed information can be found in [7].

For the u -th recipient, we obtain u -th column vector \mathbf{y}^u from the generated fingerprint code set N . Each binary value '0, 1' from \mathbf{y}^u is encoded into two symbol vectors $\mathbf{s}_0, \mathbf{s}_1$ with same length L , then we obtain a watermark sequence \mathbf{w} with length $N \cdot L$. We use zero value vectors for symbol vector \mathbf{s}_0 . For symbol vector \mathbf{s}_1 , we use a pseudo-random sequences with positive values generated by a watermark key.

We embed the generated pattern \mathbf{w} by changing the amplitude of the spectral coefficient vector \mathbf{c} . First, we determine a frequency band for watermark embedding by selecting a start index i_0 of the coefficient vector. The watermarking process is computed using the following equation:

$$c'_{(i+i_0)} = c_{(i+i_0)} + \alpha \cdot w_i \cdot |c_{(i+i_0)}|, \quad (4)$$

where $1 \leq i \leq (N \cdot L)$, and α is the strength of the watermark ($\alpha > 0$). Here, c_i and w_i denote the i -th index of \mathbf{c} and \mathbf{w} , respectively. The watermarked mesh is obtained by combining the Laplacian basis \mathbf{U} with coefficient \mathbf{c}' as in the following equation.

$$\mathbf{x}' = \mathbf{U}\mathbf{c}', \quad (5)$$

where \mathbf{x}' is a vector of the reconstructed vertices for the watermarked mesh model.

2.2 Watermark Extraction

Detection of the Embedded Watermark. We designed the proposed collusion-resilient watermarking based on informed-detection watermarking that requires a reference mesh. For the watermark extraction, we first obtained a spectral coefficient vector $\mathbf{c}^* = (c_1^*, c_2^*, \dots, c_n^*)^T$ and \mathbf{c} from the suspicious mesh \mathbb{M}^* , and reference mesh \mathbb{M} using the method described in the embedding section. Then, we calculated the residuals vector \mathbf{r}^* by subtracting \mathbf{c}^* from \mathbf{c} .

We obtain two-dimensional vector $\mathbf{s}^* = (\mathbf{s}_1^*, \mathbf{s}_2^*, \dots, \mathbf{s}_N^*)$ from \mathbf{r}^* , where

$$\mathbf{s}_j^* = (r_{i_0+Nj+1}^*, \dots, r_{i_0+N(j+1)}^*). \quad (6)$$

Then, we determined whether the suspicious model was watermarked or not as follows:

$$\begin{cases} \text{Watermarked : if } \sum_j \text{corr}(\mathbf{s}_j^*, \mathbf{s}_1)/N > \frac{\tau_w}{n}, \\ \text{No watermark : otherwise.} \end{cases} \quad (7)$$

where $1 \leq j \leq N$, and \mathbf{s}_1 is the symbol vector used in the embedding step, and n is the number of attackers that can be covered by the code set \mathbf{N} . Here, τ_w is a threshold to minimize a false-positive rate ($<10^{-5}$) that was determined by experiment without any collusion attack. Because the embedded signal could be attenuated by $\frac{1}{n}$ (in the worst case), τ_w must be adjusted by the maximum number of attackers to avoid false-negative error. The function term $\text{corr}(\mathbf{r}, \mathbf{s})$ denotes the normalized correlation calculated as in the following equation:

$$\text{corr}(\mathbf{r}, \mathbf{s}) = \frac{(\mathbf{r} - \bar{\mathbf{r}}) \cdot (\mathbf{s} - \bar{\mathbf{s}})}{\|\mathbf{r} - \bar{\mathbf{r}}\| \cdot \|\mathbf{s} - \bar{\mathbf{s}}\|}, \quad (8)$$

where the bar above a symbol denotes the mean value. When the sum of correlation values is smaller than $\frac{\tau_w}{n}$, we reject the suspicious model as a watermarked model whether the extracted code is attacked or not.

Colluder Accusation. To detect bit information ‘1’, detector response ψ was derived using the embedding signal (Eq.4),

$$\psi_j = \frac{\alpha(c_j^* \cdot \mathbf{s}_1)}{|\mathbf{s}_j^*|}, \quad (9)$$

where $1 \leq i \leq n_c$, and c_j^* is the corresponded vector of \mathbf{s}_j^* . Based on ψ , we reconstruct the fingerprint \mathbf{y}^* based on the embedding equation as follows:

$$y_j^* = \begin{cases} 1 : \text{if } \psi_j > \tau_b, \\ 0 : \text{otherwise.} \end{cases} \quad (10)$$

where y_j^* denote the j -th index of \mathbf{y} , and τ_b is a threshold to minimize the error rate (discussed in Sect. 2.3).

When we suspect that the extracted fingerprint has been attacked, we examine the extracted fingerprint \mathbf{y}^* using the code set \mathbf{N} . The averaging attack on the binary sequences can be considered the logical AND of the fingerprint codes [16]. Using the GD-PBIBD code, we can find the pirates by comparing the bit position of a result sequence whose value is ‘1’ with all columns in the code set \mathbf{N} . Therefore, we designed an examination process:

$$\mathbf{v}^* = \mathbf{N}^T \mathbf{y}^*, \quad (11)$$

where \mathbf{v}^* is a vector of which the index indicates each recipient. By normalizing and applying a floor function to each value of \mathbf{v}^* , we obtain vector \mathbf{v}' as follows:

$$\mathbf{v}' = \begin{cases} 1 : \text{if } \frac{\mathbf{v}^*}{|\mathbf{y}^*|} = 1, \\ 0 : \text{otherwise.} \end{cases} \quad (12)$$

Using the vector \mathbf{v}' , we decided whether the extracted code \mathbf{w}^* was attacked or not. If $|\mathbf{v}'| = 1$ then \mathbf{w}^* was not attacked by pirates. In contrast, if $|\mathbf{v}'| > 1$ then $|\mathbf{y}^*|$ indicates the number of attackers, and values from \mathbf{v}' demonstrate explicit clues for tracking the attackers.

2.3 Detector Response Modeling and Error Minimization

To reflect the statistical characteristics of the actual noise and interference, the threshold τ_b has to be controlled to minimize the bit error ratio. In this section, we model a distribution of the responses from the extracted bit, and determine τ_b by analyzing the proposed model.

The averaging attack of the anti-collusion code has the same effect as a bitwise AND operation of every bit position. Therefore, bit information ‘1’ can survive during the averaging when ‘0’ was not exist in the same bit position of the attacker’s code. The probability of ‘1’ from the code set \mathbf{N} is $\frac{n-1}{n}$ [7], where n denotes the number of attackers covered by \mathbf{N} . After the collusion with n attackers, the expected value of bit information ‘1’ can be modeled as a binomial distribution as $E(n) = \sum_{k=0}^n \binom{n}{k} \frac{1}{n}^k (1 - \frac{1}{n})^{n-k}$, where k is the number of the ‘1’ bit in the same position of the attacker’s code.

To reflect the statistical characteristic of the actual noise and interference, the detector response ψ was modeled as a mixture of the Gaussian distribution. Similar to $E(n)$, the embedded signal is attenuated by the number of attackers (n). Therefore, the distribution of the attenuated signal of bit ‘1’ can be modeled as

$$\Psi(x|n, \sigma) = \sum_{k=0}^n \left(\binom{n}{k} \frac{1}{n}^k (1 - \frac{1}{n})^{n-k} \cdot f(x|\frac{k}{n}, \sigma^2) \right), \quad (13)$$

where $f(x|\mu, \sigma^2)$ denotes the probability density function of the Gaussian distribution, and $\sigma = \|\alpha\|^2 / \sigma_d^2$ is determined by the watermark-to-noise-ratio (WNR) where σ_d^2 is the strength of the additive noise. Figure 1 shows simulated plots of $\Psi(x|n, \sigma)$, where $n = 8$ and $\sigma = 0.1$. To verify the proposed model in Eq. 13, we

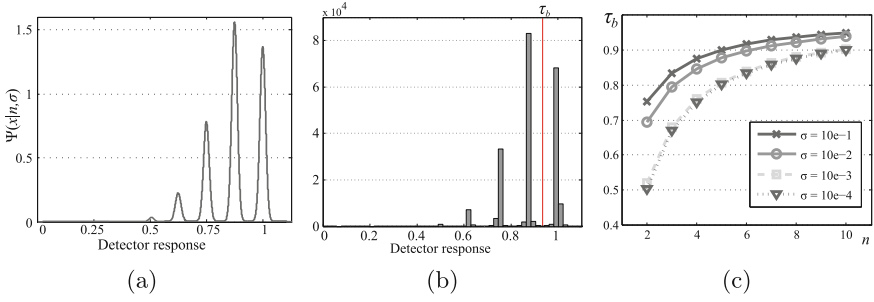


Fig. 1. (a) Plot of the proposed model $\Psi'(x|n, \sigma)$ where $n = 8$, and $\sigma = 0.1$, (b) Histogram of the detector responses obtained from the 1000 sample models ($n = 8$, $\sigma = 0.1$), and (c) obtained τ_b using Eq. 14 with various σ, n values. (Color figure online)

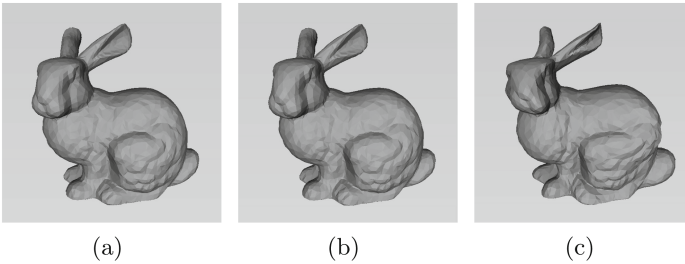


Fig. 2. Comparison between the original and the watermarked model: (a) original Bunny model, (b) watermarked model, (c) strongly watermarked model

measured the detector response from 1000 sample models with the same parameters ($n = 8, \sigma = 0.1$). As a result of the tests, Fig. 1(b) shows a histogram of the measured responses that has structure very similar to that of the proposed model (Fig. 1(a)).

Using the model Ψ , we determine the threshold τ_b by minimizing the code extraction error as follows:

$$\tau_b = \underset{t}{\text{minimize}} (e_1 + e_2), \tag{14}$$

where e_1 and e_2 are false positive error and false negative error, respectively. Each error probability was derived from Ψ by $e_1 = \int_0^t \left(\frac{1}{n}\right)^n \cdot f(x|1, \sigma^2) dx$, and $e_2 = \int_t^\infty \sum_{k=0}^{n-1} \left(\binom{n}{k} \frac{1}{n}^k \left(1 - \frac{1}{n}\right)^{n-k} \cdot f(x|\frac{k}{n}, \sigma^2)\right) dx$.

Figure 1(c) shows τ_b values for various σ and n by solving Eq. 14. For instance, $\tau_b (=0.9378)$ for the test environment $n = 8, \sigma = 0.1$ is presented as a red line in Fig. 1(b).

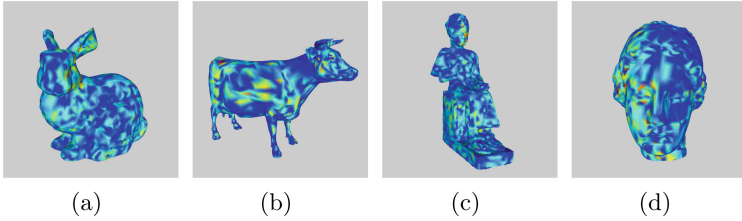


Fig. 3. Sample distortion maps of the watermarked meshes with $MRMS = 0.02\%$: (a) Bunny, (b) Cow, (c) Ramesses, and (d) Venus.

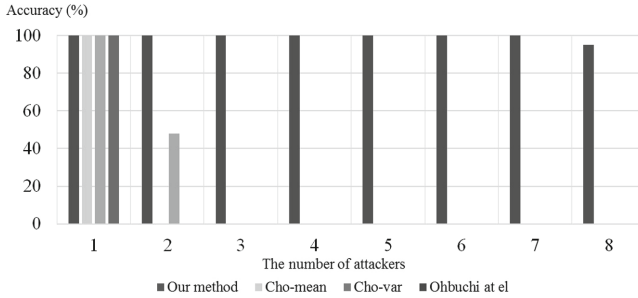


Fig. 4. Comparison test results with collusion attack

3 Experimental Results

To assess the proposed scheme, we tested its robustness against collusion attacks. Experiments were carried out on six triangular mesh models from 3D mesh watermarking benchmark project [18]: Bunny, Rabbit, Venus, Horse, Cow, and Ramesses. We used the fingerprint code set constructed by [9] with parameters (72, 81, 9, 8, 0, 1), which could protect the model against $n = 8$ attackers. The watermarking algorithm was implemented in MATLAB 2012b using the mesh toolbox [13]. The geometric distortion was measured using maximum root mean square error (MRMS) [6].

Figure 2(b) shows the watermark with $MRMS = 0.02\%$ ($\alpha = 0.05$), and no visual difference can be perceived between Fig. 2(a) and (b). In addition, we embedded the watermark with high strength factor ($\alpha = 0.5$) to show the visual shape of the watermark ($MRMS = 2\%$). The embedded watermark appears as noise strongly represented on certain areas of the model's surface. In addition, Fig. 3 illustrates the distortion maps which represents the HSV maps of the geometric objective distortions between the original and the watermarked meshes based on the hausdorff distance [6].

The experiments were divided into two test environments: (1) collusion attack test with a variety of compared methods, and (2) robustness test with collusion and noise addition attacks. To compare our method against the performance of a conventional scheme, we used a variety of robust watermarking methods

as follows: Ohbuchi et al. [12], and Cho et al. [5]. Similar to our method, the method of Ohbuchi et al. [12] is based on the mesh spectral decomposition. The method of Cho et al. [5] embedded a watermark pattern by modifying the statistical features of the vertex norm. Anti-collusion codes were adopted not only our method but also the compared methods [5, 12].

In the experiments, we generated 81 watermarked models by embedding a fingerprint code for each recipient of the models. Then collusion attacks were performed 100 times using a set of randomly selected watermarked models for each iteration. For the collusion attack for our tests, we averaged a set of selected models and obtained one attacked model. We also conducted experiments by varying the number of selected models k ($1 \leq k \leq 8$). Robustness for collusion attack was evaluated by the number of times that tracing all attackers was successful.

Figure 4 shows the experimental results of collusion attack of the proposed method, Ohbuchi et al. [12], and Cho et al. [5]. In this figure, *Cho-mean* and *Cho-var* indicate the first and the second method of Cho et al., respectively. The proposed method perfectly found attackers for every attack when the number of attackers was 1–7, and it also performed well when the number of attackers was ‘8’. In contrast, *Cho-mean* and Ohbuchi et al. did not trace attackers for all attacks when the number of attackers was two or more. *Cho-var* also did not work when the number of attackers was three or more. The results showed that proposed scheme was much more robust against collusion attack compared to the conventional schemes.

Figure 5 shows robustness to the combination of noise addition and collusion attack. We added Gaussian noise $\mathcal{N}(0, \sigma)$ to the averaged model, where $\sigma = 0.1, 0.01, 0.001$, or 0.0001 , and traced the attackers using the proposed detector. The performance of the proposed method for the noise strength $\sigma < 0.001$ was as good as the result for $\sigma = 0$, and the results in cases with $\sigma = 0.001$ was also acceptable. On the other hand, in the experiments with strong noise ($\sigma \geq 0.01$), the accuracy of our method decreased. Therefore, the result shows that the proposed method was robust to the combination of a small amount of added noise and collusion attack.

In addition, we also conducted tests for the proposed watermarking based on the Tardos code. First, we used a parameter set ($\epsilon = 0.1, c = 3, \text{length} = 2072$) to generate the fingerprint code matrix. We added Gaussian noise $\mathcal{N}(0, \sigma)$ to the averaged model, where $\sigma = 0.01, 0.001$, or 0.0001 , and traced the attackers using the proposed detector. The accuracy of the proposed method for the small noise $\sigma < 0.001$ was 95% accuracy with $n = 2$. However, in the tests with strong noise ($\sigma \geq 0.01$), the accuracy of our method was not acceptable (under 30% accuracy with $n = 3$). Compared to the GD-PBIBD based scheme, Tardos’s code based scheme requires more bit capacity to carry the fingerprint code. This aspect results reduction in the ratio between imperceptibility and robustness for the fingerprinting scheme. Although Tardos’s fingerprint code outperforms the other anti-collusion codes in many applications, it does not work well in some cases of the mesh fingerprinting such as the high number of accused users.

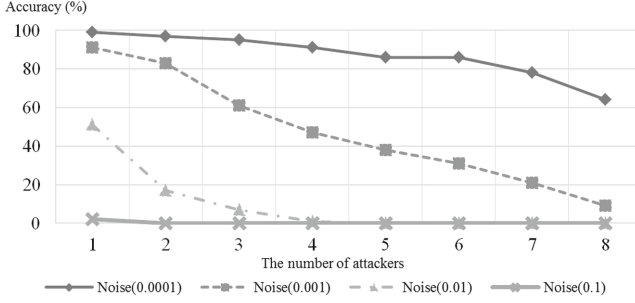


Fig. 5. Accuracy of the proposed method with collusion attack and noise addition

4 Discussion and Conclusion

The main contribution of our work is a collusion resilient watermark based on the anti-collusion fingerprint code. In contrast to the existing robust mesh watermarking, our proposed method has sufficient capacity to carry the anti-collusion fingerprint code. In order to minimize the detection error, we also modeled the response of the detector and presented optimized thresholds for our method. Based on experiments with a public benchmark, the proposed method outperformed the conventional robust mesh watermarking in all cases.

Nowadays, the demand for 3D mesh watermarking has increased due to the emergence of low-cost 3D printers. On the other hand, copyright issues will inevitably occur with the expansion of 3D model sharing platforms such as Thingiverse, Pinshape, and Sketchfab. In spite of various watermarking methods, mesh watermarking techniques resilient to collusion attack were rarely considered. Therefore, we believe that the proposed method can give advantages to our industry as well as the academic society.

Acknowledgement. This work was supported by Samsung Research Funding Center of Samsung Electronics under Project Number SRFCIT1402-05. The work of Jong-Uk Hou was supported by a Global Ph.D Fellowship Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2015H1A2A1030715).

6 Appendix

In this appendix, we focus on mathematical derivation, in order to provide the threshold value τ_b illustrated in Fig. 1(c). From Eq. 14, τ_b is value that makes the derivative of $e_1 + e_2$ by t to be zero.

$$\frac{de_1}{dt} = \left(\frac{1}{n}\right)^n \cdot f\left(t\left|1, \sigma^2\right.\right) \quad (15)$$

$$\frac{de_2}{dt} = - \sum_{k=0}^{n-1} \left(\binom{n}{k} \frac{1}{n}^k \left(1 - \frac{1}{n}\right)^{n-k} \cdot f\left(t\left|\frac{k}{n}, \sigma^2\right.\right) \right) \quad (16)$$

Substitute $\binom{n}{k} \frac{1}{n} (1 - \frac{1}{n})^{n-k}$ to α_k for $k = 1$ to n .

$$\frac{de_1 + de_2}{dt} = \alpha_n \cdot f(t|1, \sigma^2) - \left(\sum_{k=0}^{n-1} \alpha_k \cdot f(t|\frac{k}{n}, \sigma^2) \right) \quad (17)$$

$$= \frac{1}{\sqrt{2\pi\sigma}} \cdot \left(\alpha_n \cdot e^{-\frac{(t-1)^2}{2\sigma^2}} - \left(\sum_{k=0}^{n-1} \alpha_k \cdot e^{-\frac{(t-\frac{k}{n})^2}{2\sigma^2}} \right) \right) \quad (18)$$

$$= \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{t^2}{2\sigma^2}} \cdot \left(\alpha_n e^{\frac{2t-1}{2\sigma^2}} - \left(\sum_{k=0}^{n-1} \alpha_k e^{\frac{2k \cdot t - \frac{k^2}{n^2}}{2\sigma^2}} \right) \right) \quad (19)$$

Substitute $e^{\frac{t}{n\sigma^2}}$ to x and $\alpha_k e^{-\frac{k^2}{2n^2\sigma^2}}$ to β_k for $k = 1$ to n .

$$= \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{t^2}{2\sigma^2}} \cdot \left(\beta_n x^n - \sum_{k=0}^{n-1} \beta_k x^k \right) \quad (20)$$

Therefore, the value τ_b that minimizes $e_1 + e_2$ can be derived from solving following polynomial of degree n .

$$\beta_n x^n - \sum_{k=0}^{n-1} \beta_k x^k = 0 \quad (21)$$

Finally, we can get τ_b by using solution of (Eq.21)

$$\tau_b = n \cdot \sigma^2 \cdot \ln(x) \quad (22)$$

References

1. Alface, P.R., Macq, B., Cayre, F.: Blind and robust watermarking of 3D models: how to withstand the cropping attack? In: 2007 IEEE International Conference on Image Processing, ICIP 2007, vol. 5, p. V-465. IEEE (2007)
2. Bollobás, B.: Modern Graph Theory, vol. 184. Springer, New York (1998). <https://doi.org/10.1007/978-1-4612-0619-4>
3. Boneh, D., Shaw, J.: Collusion-secure fingerprinting for digital data. IEEE Trans. Inf. Theor. **44**(5), 1897–1905 (1998)
4. Celik, M.U., Sharma, G., et al.: Collusion-resilient fingerprinting using random pre-warping. In: 2003 International Conference on Image Processing, Proceedings, ICIP 2003, vol. 1, p. I-509. IEEE (2003)
5. Cho, J.W., Prost, R., Jung, H.Y.: An oblivious watermarking for 3-D polygonal meshes using distribution of vertex norms. IEEE Trans. Sig. Process. **55**(1), 142–155 (2007)
6. Cignoni, P., Rocchini, C., Scopigno, R.: Metro: measuring error on simplified surfaces. Technical report, Paris, France (1996)
7. Clatworthy, W.H., Cameron, J.M., Speckman, J.A.: Tables of Two-Associate-Class Partially Balanced Designs, vol. 63. US Government Printing Office, Washington (1973)

8. Hou, J.U., Kim, D.G., Choi, S., Lee, H.K.: 3D print-scan resilient watermarking using a histogram-based circular shift coding structure. In: Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security, pp. 115–121. ACM (2015)
9. InKoo, K., Sinha, K., Lee, H.K.: New digital fingerprint code construction scheme using group-divisible design. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **89**(12), 3732–3735 (2006)
10. Kang, I.K., Lee, C.H., Lee, H.Y., Kim, J.T., Lee, H.K.: Averaging attack resilient video fingerprinting. In: 2005 IEEE International Symposium on Circuits and Systems, ISCAS 2005, pp. 5529–5532. IEEE (2005)
11. Karni, Z., Gotsman, C.: Spectral compression of mesh geometry. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, pp. 279–286. ACM Press/Addison-Wesley Publishing Co. (2000)
12. Ohbuchi, R., Takahashi, S., Miyazawa, T., Mukaiyama, A.: Watermarking 3D polygonal meshes in the mesh spectral domain. In: Graphics Interface. vol. 2001, pp. 9–17. Citeseer (2001)
13. Peyré, G.: The numerical tours of signal processing - advanced computational signal and image processing. *IEEE Computing in Science and Engineering* **13**(4), 94–97 (2011). <http://hal.archives-ouvertes.fr/hal-00519521/>
14. Stinson, D.R., Wei, R.: Combinatorial properties and constructions of traceability schemes and frameproof codes. *SIAM J. Discrete Math.* **11**(1), 41–53 (1998)
15. Tardos, G.: Optimal probabilistic fingerprint codes. *J. ACM (JACM)* **55**(2), 10 (2008)
16. Trappe, W., Wu, M., Wang, Z., Liu, K.: Anti-collusion fingerprinting for multimedia. *IEEE Trans. Sig. Process.* **51**(4), 1069–1087 (2003)
17. Uccheddu, F., Kuo, C.C.J., Barni, M.: Anticollusion watermarking of 3D meshes by pre-warping. In: Electronic Imaging 2008, p. 68190S. International Society for Optics and Photonics (2008)
18. Wang, K., Lavoué, G., Denis, F., Baskurt, A., He, X.: A benchmark for 3D mesh watermarking. In: Proceeding of the IEEE International Conference on Shape Modeling and Applications, pp. 231–235 (2010)
19. Wang, K., Lavoué, G., Denis, F., Baskurt, A.: Technical section: robust and blind mesh watermarking based on volume moments. *Comput. Graph.* **35**(1), 1–19 (2011)
20. Zhu, X., Chen, C.W.: A collusion resilient key management scheme for multi-dimensional scalable media access control. In: 2011 18th IEEE International Conference on Image Processing (ICIP), pp. 2769–2772, September 2011

Theory in Security



The Search Successive Minima Problem Is Equivalent to Its Optimization Version

Haoyu Li^{1,2,3} and Yanbin Pan^{1,2}(✉)

¹ Key Laboratory of Mathematics Mechanization, NCMIS,
Academy of Mathematics and Systems Science, Chinese Academy of Sciences,
Beijing 100190, China

panyanbin@amss.ac.cn

² Science and Technology on Communication Security Laboratory,
Chengdu 610041, China

³ School of Mathematical Sciences, University of Chinese Academy of Sciences,
Beijing 100049, China

lihaoyu14@mailsucas.ac.cn

Abstract. The shortest vector problem (SVP) and the shortest independent vectors problem (SIVP) are two famous problems in lattices, which are usually used to evaluate the hardness of some computational problems related to lattices. It is well known that the search-SVP is equivalent to its optimization version. However, it seems very difficult to prove the equivalence between search-SIVP and optimization-SIVP. In this paper, we revisit the Successive Minima Problem (SMP), which is proved the equivalence relation with SIVP. Naturally we will consider its optimization version as to find all successive minima of a given lattice, and finally we will prove that it is equivalent to its search version.

Keywords: Lattice · Successive minima · SMP · SVP · SIVP

1 Introduction

Since Ajtai's seminal work [1] in 1996, lattice-based cryptosystems become more and more popular due to their potential ability to resist the quantum computer attack and successful applications in constructing important cryptographic primitives: such as the hash functions [1, 19, 20, 23], the digital signature schemes [4, 9, 13], the encryption schemes [3, 11, 14, 25], and the fully homomorphic encryption schemes [7, 10].

Another attractive feature of lattice-based cryptosystems is their average-case security can be based on the worst-case hardness of some lattice problems,

This work was supported in part by the NNSF of China (No. 61572490, and No. 11471314), the National Center for Mathematics and Interdisciplinary Sciences, CAS, and Science and Technology on Communication Security Laboratory (No. 9140C110301150C11051).

which are typically some approximation variants of the shortest vector problem (SVP) and the shortest independent vectors problem (SIVP).

SVP refers to the problem of finding a shortest non-zero vector in a given lattice, and its hardness has been studied widely [2, 6, 8, 12]. Interestingly, there are three variants of SVP: search-SVP, optimization-SVP, and decisional-SVP, which aim to find a shortest nonzero vector, find the length of the shortest vector, and decide whether the shortest vector is shorter than some given number respectively. It is well known that the three variants of SVP are equivalent to each other (see [22]). In fact, it is obvious that if we could solve search-SVP then we can solve the other two problems. Moreover, it is easy to show the equivalence between decisional-SVP and optimization-SVP. However, reducing search-SVP to optimization-SVP is not an easy task.

The first efficient reduction from search-SVP to optimization-SVP was presented by Kannan [18] in 1987. However, the reduction is not rank-preserving, since it needs the optimization-SVP oracle to deal with some lower rank lattices, besides the lattices with the same rank as the original lattice. Moreover, the reduction invokes the optimization-SVP oracle for polynomial times. In 2013, Hu and Pan [16] revisited the reduction and presented a rank-preserving reduction which can solve search-SVP with only one call to the optimization-SVP oracle.

When considering the relations between search-SIVP and optimization-SIVP, it becomes a bit more complicated. Search-SIVP refers to the question of finding n linearly independent vectors in a given n -rank lattice \mathcal{L} such that the maximal length of the vectors is as small as possible. In fact, denote by $\lambda_i(\mathcal{L}) (1 \leq i \leq n)$ the successive minima, that is, the minimum length of a set of i linearly independent vectors in \mathcal{L} , where the length of a set is defined as the length of the longest vector in it. Then the target of search-SIVP is to find n linearly independent vectors with length at most $\lambda_n(\mathcal{L})$, whereas optimization-SIVP should be defined as the problem to find λ_n . It is obvious that optimization-SIVP can be reduced to its search version. However, it seems hard to give a reduction from the search version to the optimization version since λ_n is only an upper bound of the length of these independent vectors.

In this paper, we consider a lattice problem called the successive minima problem (SMP), which is introduced in [5]. In fact, the original SMP in [5] which aims to find n linearly independent vectors achieving the successive minima respectively is a search version of this problem. We will naturally consider its optimization version as to find all the values of successive minima. Therefore, the relation between these two variants will be considered. Obviously, a reduction from optimization-SMP to its search version is trivial, but the inverse reduction seems difficult.

By perturbing the original lattice basis carefully as in [16], we can transform it to another basis of a new lattice, and we consider the relation between this pair of lattice bases. Then we find that the components of all successive minimal vectors do not change. Moreover, the successive minima of the new lattice are all different, which lead to an algorithm to recover all components for successive

minimal vectors of the original lattice. Similar to [16], the reduction from search-SMP to optimization-SMP is also rank-preserving. But by using some results of matrix analysis, we find that our reduction holds for every lattice, no matter whether it is full-rank or not.

Roadmap. The remainder of the paper is organized as follows. In Sect. 2, we give some preliminaries needed. In Sect. 3, we describe the reduction from search successively minimal vectors to its optimization version. Finally, we give a short conclusion in Sect. 4.

2 Preliminaries

We denote by $\mathbb{Z}, \mathbb{R}, \mathbb{C}, \mathbb{Z}^+,$ and \mathbb{R}^+ the integer ring, the real field, the complex field, the set of positive integers, and the set of positive real numbers respectively.

For any vector $v = (v_1, v_2, \dots, v_m)^T \in \mathbb{R}^m$ and $m \in \mathbb{Z}^+,$ we denote by $\|v\| = \sqrt{\sum_{i=1}^m v_i^2}$ its length.

2.1 Lattice and the Successively Minima Problem

Given a matrix $B = (b_{ij}) \in \mathbb{R}^{m \times n}$ with rank $n,$ the lattice $\mathcal{L}(B)$ spanned by the columns of B is

$$\mathcal{L}(B) = \{Bx = \sum_{i=1}^n x_i b_i \mid x_i \in \mathbb{Z}\},$$

where b_i is the i th column of $B.$ We call m, n the dimension and the rank of $\mathcal{L}(B)$ respectively. The determinant of $\mathcal{L}(B),$ say $\det(\mathcal{L}(B)),$ is defined as $\sqrt{|\det(B^T B)|}.$ It is easy to see when B is full-rank ($n = m$), its determinant becomes $|\det(B)|.$

Definition 1 (Successive Minima). For a n -rank lattice $\mathcal{L}(B)$ with $B \in \mathbb{R}^{m \times n},$ and $i \in \{1, 2, \dots, n\},$ we define the i th successive minimum as

$$\lambda_i(\mathcal{L}(B)) = \inf \left\{ r \in \mathbb{R}^+ \mid \dim(\overline{\text{span}(B(0, r))} \cap \mathcal{L}(B)) \geq i \right\},$$

where $\overline{B(0, r)}$ is the closed ball centered at 0 with radius $r \in \mathbb{R}^+,$ i.e., $\overline{B(0, r)} = \{x \in \mathbb{R}^m \mid \|x\| \leq r\}.$

Simply speaking, $\lambda_i(\mathcal{L}(B))$ means the infimum of the maximal length of i linearly independent vectors in $\mathcal{L}(B).$

It is well-known that the successive minima can be achieved, that is, there exist n linearly independent lattice vectors $v_1, v_2, \dots, v_n \in \mathcal{L}(B)$ such that $\|v_i\| = \lambda_i(\mathcal{L}(B)).$ Therefore, we can define

Definition 2 (Successively Minimal Vectors). Given a lattice basis $B \in \mathbb{R}^{m \times n}$ with rank $n,$ any n linearly independent lattice vectors $v_1, v_2, \dots, v_n \in \mathcal{L}(B)$ satisfying $\|v_i\| = \lambda_i(\mathcal{L}(B))$ are called the successively minimal vectors of $\mathcal{L}(B).$

In [5,21], the Successive Minima Problem is defined as below:

Definition 3 (SMP $_{\gamma}$). *Given a lattice $\mathcal{L}(B)$ and a constant $\gamma \geq 1$, output n linearly independent vectors v_1, v_2, \dots, v_n in $\mathcal{L}(B)$ such that $\|v_i\| \leq \gamma \lambda_i(\mathcal{L}(B))$.*

When $\gamma = 1$, it becomes Search-SMP:

Definition 4 (Search-SMP). *Given a lattice $\mathcal{L}(B)$, find a set of the successively minimal vectors in $\mathcal{L}(B)$.*

It is proved that SMP is equivalent to SIVP and the closest vector problem (CVP) in [21]. We can define its optimization version similar to SVP as following:

Definition 5 (Optimization-SMP). *Given a lattice $\mathcal{L}(B)$, find the successive minima of $\mathcal{L}(B)$.*

2.2 Linear Algebra

For a matrix $A \in \mathbb{C}^{m \times n}$, we denote by A^* its conjugate transpose and A^T its transpose. The singular values of A is defined to be the nonnegative square root of the eigenvalues of A^*A .

Using the singular values or the eigenvalues of matrices, we can obtain the following Lemma stated in [15]:

Lemma 1 (Rayleigh quotient). *Let $A \in \mathbb{C}^{m \times n}$ and $0 \leq \mu_1 \leq \mu_2 \leq \dots \leq \mu_n$ be all eigenvalues of A^*A , then for any $x = (x_i) \in \mathbb{C}^n \setminus \{0\}$, we have*

$$\mu_1 \leq \frac{x^* A^* A x}{x^* x} \leq \mu_n.$$

We present a lower bound for the smallest singular value of a matrix in the following lemma, whose proof can be found in [24].

Lemma 2. *Given a matrix $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ with $\det(A) \neq 0$, we let $0 \leq \sigma_1 \leq \sigma_2 \leq \dots \leq \sigma_n$ be all singular values of A , then the smallest singular value σ_1 satisfies the inequality:*

$$\sigma_1 \geq \left(\frac{n-1}{\|A\|_F^2} \right)^{\frac{n-1}{2}} |\det(A)|,$$

where $\|A\|_F = (\sum_{i,j=1}^n |a_{ij}|^2)^{\frac{1}{2}}$ is the Frobenius norm of A .

Finally, we give a lemma to illustrate the perturbation bound for the determinant of a matrix [17]:

Lemma 3. *Let $B, C \in \mathbb{C}^{n \times n}$, then*

$$|\det(B+C) - \det(B)| \leq n \|C\|_F \max\{\|B\|_F, \|B+C\|_F\}^{n-1}.$$

3 The Search-SMP Is Equivalent to Optimization-SMP

It is obvious that if we could solve search-SMP, then we can solve optimization-SMP easily. To show the equivalence between the two problems, what we really need is a reduction from search-SMP to its optimization version.

In this section, we give such a reduction, which consists of three main steps. Suppose we want to find the successively minimal vectors in a given lattice $\mathcal{L}(B)$, we first construct a new lattice basis B_ϵ by perturbing the original basis B . By the optimization-SMP oracle, we can then get the successive minima of $\mathcal{L}(B_\epsilon)$. In fact, using the successive minima, we can efficiently recover the coefficients of the successively minimal vectors under the basis B_ϵ . With the recovered coefficients, we can get the successively minimal vectors in $\mathcal{L}(B)$ finally.

First we present a lemma to show that the coefficients of the successively minimal vectors under the basis can be well bounded.

Lemma 4. *Given a lattice basis $B = (b_{ij}) \in \mathbb{Z}^{m \times n}$ with rank n , let $M = \max\{|b_{ij}|\}$. For any $x = (x_i) \in \mathbb{Z}^n$ such that $\|Bx\| \leq \lambda_n(\mathcal{L}(B))$, we have*

$$x_i^2 \leq 2^{\frac{n+1}{2}} n^{\frac{n-1}{2}} (mM^2)^n.$$

Proof. Note that when $n = 1$, the result is trivial, so we assume $n \geq 2$.

It is easy to check that $\lambda_i(\mathcal{L}(B))^2 \leq \max\{\|b_i\|^2\} \leq mM^2$, so for any $x = (x_i) \in \mathbb{Z}^n$ such that $\|Bx\| \leq \lambda_n(\mathcal{L}(B))$, we have

$$\|Bx\|^2 \leq mM^2.$$

Considering the Gram matrix $A = B^T B$, that is,

$$A = \begin{pmatrix} b_{11}^2 + b_{21}^2 + \cdots + b_{m1}^2 & \cdots & b_{11}b_{1n} + b_{21}b_{2n} + \cdots + b_{m1}b_{mn} \\ \vdots & \ddots & \vdots \\ b_{1n}b_{11} + b_{2n}b_{21} + \cdots + b_{mn}b_{m1} & \cdots & b_{1n}^2 + b_{2n}^2 + \cdots + b_{mn}^2 \end{pmatrix},$$

by Lemma 1, we know that

$$0 < \mu_1(A) = \mu_1(B^T B) \leq \frac{x^T B^T B x}{x^T x} = \frac{\|Bx\|^2}{\|x\|^2}.$$

Together with $\|Bx\|^2 \leq mM^2$, we have

$$\|x\|^2 \leq \frac{mM^2}{\mu_1(A)}.$$

So for each $i(1 \leq i \leq n)$, we have

$$|x_i| \leq \frac{\sqrt{m}M}{\sqrt{\mu_1(A)}}. \tag{1}$$

Note that the singular values of $A = B^T B$ are in fact their eigenvalues. By the lower bound of the smallest singular value in Lemma 2, we know

$$\mu_1(A) \geq \left(\frac{n-1}{\|A\|_F^2} \right)^{\frac{n-1}{2}} |\det(A)|. \quad (2)$$

Since the entries of B are integers, then

$$|\det(A)| \geq 1 > \frac{1}{2}.$$

Notice that the absolute values of entries of B are bounded by M , then the absolute values of entries of A are bounded by mM^2 , which implies that

$$\|A\|_F^2 \leq n^2 (mM^2)^2 = (nmM^2)^2.$$

Since $n \geq 2$, we have $n-1 \geq \frac{n}{2}$. Hence, we have:

$$\mu_1 \geq \frac{1}{2} \left(\frac{1}{2nm^2M^4} \right)^{\frac{n-1}{2}}$$

By (1),

$$x_i^2 \leq 2mM^2 (2nm^2M^4)^{\frac{n-1}{2}} = 2^{\frac{n+1}{2}} n^{\frac{n-1}{2}} (mM^2)^n.$$

Remark 1. We can also use $B^T Bx$ to evaluate the upper bound of each component of x by the Cramer's Rule and Hadamard inequality, and the upper bound will be $n^{n/2} m^{n+1/2} M^{2n}$. This bound is not so tight as in Lemma 4.

In the following, we describe our reduction in detail.

Theorem 1. *Given an oracle \mathcal{O} that can solve the optimization SMP for any lattice $\mathcal{L}(B')$ with basis $B' \in \mathbb{Z}^{m \times n}$, there is a deterministic polynomial time algorithm that can solve the search SMP for $\mathcal{L}(B)$ with the input basis $B \in \mathbb{Z}^{m \times n}$.*

Proof. We will complete the proof in the following 4 steps:

(1) First we construct a matrix $B_\epsilon \in \mathbb{Z}^{m \times n}$:

$$B_\epsilon = \epsilon_{n+1} B + \begin{pmatrix} \epsilon_1 & \epsilon_2 & \dots & \epsilon_n \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

where ϵ_i 's are determined as below.

Let $M_1 = 2^{\frac{n+1}{4}} n^{\frac{n-1}{4}} m^{\frac{n}{2}} (M+1)^n$ and $M_2 = \sqrt{m}(M+1)$ where $M = \max\{|b_{ij}|\}$, then we choose

$$p = 2 \max\{M_2^2, 2M_1 M_2, 2M_1^2\} + 1.$$

Note that $\log p = \text{poly}(n, m, \log M)$, where $\text{poly}(n, m, \log M)$ stands for a polynomial of n , m and $\log M$.

Then we choose $n + 1$ positive integers $a_1 < a_2 < \dots < a_n < a_{n+1}$ such that all $a_i + a_j$ ($1 \leq i \leq j \leq n + 1$)'s are distinct and a_{n+1} is bounded by $\text{poly}(n)$. As in Lemma 1 of [16], we can first choose

$$a_i = i^2 + (2(n + 1)^2)i + 4(n + 1)^4,$$

for $i = 1, 2, \dots, n$. Then we let

$$a_{n+1} = 3a_n.$$

By Lemma 1 in [16], all $a_i + a_j$ ($1 \leq i \leq j \leq n$)'s are distinct. Together with the fact that $a_{n+1} > 2a_n$, it is easy to see that $a_i + a_j$ ($1 \leq i \leq j \leq n + 1$)'s are distinct and a_{n+1} is bounded by $\text{poly}(n)$. Finally we let

$$\epsilon_i = p^{a_i}.$$

Notice that for every entry $b_{\epsilon ij}$ in B_ϵ , $\log |b_{\epsilon ij}| = \text{poly}(n, m, \log M)$. Hence B_ϵ can be constructed efficiently.

- (2) Next we claim that the columns of B_ϵ are linearly independent, so B_ϵ forms a lattice basis of $\mathcal{L}(B_\epsilon)$. In fact we can prove the claim by showing that $\det(B_\epsilon^T B_\epsilon) \neq 0$. In the following, we prove that

$$\left| \det\left(\left(\frac{1}{\epsilon_{n+1}} B_\epsilon\right)^T \left(\frac{1}{\epsilon_{n+1}} B_\epsilon\right)\right) \right| = \left| \det\left(\frac{1}{\epsilon_{n+1}^2} B_\epsilon^T B_\epsilon\right) \right| > \frac{1}{2}.$$

Notice that the absolute values of entries of $\frac{1}{\epsilon_{n+1}} B_\epsilon$ can be bounded by $M + 1$, then the absolute values of entries of $\left(\frac{1}{\epsilon_{n+1}} B_\epsilon\right)^T \left(\frac{1}{\epsilon_{n+1}} B_\epsilon\right)$ are bounded by $m(M + 1)^2$. Note that

$$\begin{aligned} \left(\frac{1}{\epsilon_{n+1}} B_\epsilon\right)^T \left(\frac{1}{\epsilon_{n+1}} B_\epsilon\right) &= B^T B + \frac{1}{\epsilon_{n+1}} B^T \begin{pmatrix} \epsilon_1 & \epsilon_2 & \dots & \epsilon_n \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix} + \\ &\frac{1}{\epsilon_{n+1}} \begin{pmatrix} \epsilon_1 & 0 & \dots & 0 \\ \epsilon_2 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \epsilon_n & 0 & \dots & 0 \end{pmatrix} B + \frac{1}{\epsilon_{n+1}^2} \begin{pmatrix} \epsilon_1^2 & \epsilon_1 \epsilon_2 & \dots & \epsilon_1 \epsilon_n \\ \epsilon_2 \epsilon_1 & \epsilon_2^2 & \dots & \epsilon_2 \epsilon_n \\ \vdots & \vdots & \ddots & \vdots \\ \epsilon_n \epsilon_1 & \epsilon_n \epsilon_2 & \dots & \epsilon_n^2 \end{pmatrix} \end{aligned}$$

Let $A = B^T B$ and $C = \left(\frac{1}{\epsilon_{n+1}} B_\epsilon\right)^T \left(\frac{1}{\epsilon_{n+1}} B_\epsilon\right) - A$, then each entry of C can be bounded by $2M \frac{\epsilon_n}{\epsilon_{n+1}} + \frac{\epsilon_n^2}{\epsilon_{n+1}^2} \leq (2M + 1) \frac{\epsilon_n}{\epsilon_{n+1}} \leq 2(M + 1) \frac{\epsilon_n}{\epsilon_{n+1}}$, which implies

$$\|C\|_F \leq 2n(M + 1) \frac{\epsilon_n}{\epsilon_{n+1}}.$$

By Lemma 3, we have

$$\begin{aligned} |\det(A + C) - \det(A)| &\leq n\|C\|_F \max\{\|A\|_F, \|A + C\|_F\}^{n-1} \\ &\leq 2n^2(M + 1) \frac{\epsilon_n}{\epsilon_{n+1}} (nm(M + 1)^2)^{n-1} \\ &= 2n^{n+1}m^{n-1}(M + 1)^{2n-1} \frac{\epsilon_n}{\epsilon_{n+1}}. \end{aligned}$$

By the choice of $a_n \geq n$ and $p > M_2^2$, we have

$$\begin{aligned} p^{a_{n+1}-a_n} &\geq p^{2n} \\ &> (m(M + 1)^2)^{2n} \\ &\geq 4m^{2n}(M + 1)^{2n} \\ &> 4m^{n-1}n^{n+1}(M + 1)^{2n-1}. \end{aligned}$$

That is

$$\frac{\epsilon_n}{\epsilon_{n+1}} < \frac{1}{4n^{n+1}m^{n-1}(M + 1)^{2n-1}}.$$

Then we immediately have $|\det(A + C) - \det(A)| < \frac{1}{2}$, which is in fact

$$\left| \det\left(\left(\frac{1}{\epsilon_{n+1}}B\right)^T \left(\frac{1}{\epsilon_{n+1}}B\right) - \det(B^T B)\right) \right| < \frac{1}{2}.$$

Note that $\det(B^T B)$ is a nonzero integer, then we finally have

$$\left| \det\left(\left(\frac{1}{\epsilon_{n+1}}B\right)^T \left(\frac{1}{\epsilon_{n+1}}B\right)\right) \right| > \frac{1}{2}.$$

For a vector satisfying $\|B_\epsilon x\| \leq \lambda_n(\mathcal{L}(B_\epsilon))$, all the components $|x_i|$ of x can also be bounded by M_1 .

- (3) Moreover, we claim that if n linearly independent vectors $B_\epsilon x_1, B_\epsilon x_2, \dots, B_\epsilon x_n \in \mathcal{L}(B_\epsilon)$ form a set of the successively minimal vectors in $\mathcal{L}(B_\epsilon)$ where $x_1, x_2, \dots, x_n \in \mathbb{Z}^n$, that is, $\|B_\epsilon x_i\| = \lambda_i(\mathcal{L}(B_\epsilon)), 1 \leq i \leq n$, then $Bx_1, Bx_2, \dots, Bx_n \in \mathcal{L}(B)$ also form a set of the successively minimal vectors in $\mathcal{L}(B)$.

First note that since $B_\epsilon x_1, B_\epsilon x_2, \dots, B_\epsilon x_n \in \mathcal{L}(B_\epsilon)$ are linearly independent and B_ϵ is a basis, then x_1, x_2, \dots, x_n are linearly independent, which implies that $Bx_1, Bx_2, \dots, Bx_n \in \mathcal{L}(B)$ are linearly independent.

Second we will prove that $\|Bx_i\| = \lambda_i(\mathcal{L}(B))$, for $1 \leq i \leq n$. For contradiction, let l be the smallest index such that for $1 \leq i < l$, $\|Bx_i\| = \lambda_i(\mathcal{L}(B))$, whereas $\|Bx_l\| > \lambda_l(\mathcal{L}(B))$. We have

$$\|Bx_l\|^2 \geq \lambda_l(\mathcal{L}(B))^2 + 1. \quad (3)$$

By the definition of successively minimal vectors, there must exist vectors $y_i = (y_{i1}, y_{i2}, \dots, y_{in})^T \in \mathbb{Z}^n$, $1 \leq i \leq l$ such that $\|By_i\| = \lambda_i(\mathcal{L}(B))$ and By_1, By_2, \dots, By_l are linearly independent.

Considering $B_\epsilon y_i$, note that

$$\|B_\epsilon y_i\|^2 = \epsilon_{n+1}^2 \|By_i\|^2 + \sum_{j=1}^n y_{ij}^2 \epsilon_j^2 + \sum_{j=1}^n 2c(y_i) y_{ij} \epsilon_j \epsilon_{n+1} + \sum_{1 \leq j < k \leq n} 2y_{ij} y_{ik} \epsilon_j \epsilon_k, \quad (4)$$

where $c(y_i) = \sum_{j=1}^n b_{1j} y_{ij}$ for any $y_i \in \mathbb{Z}^n$. Since $\|By_i\| = \lambda_i(\mathcal{L}(B))$, we know that

$$\|By_i\| \leq M_2.$$

By Lemma 4, we have for $1 \leq j \leq n$

$$|y_{ij}| \leq M_1.$$

Note that $|c(y_i)| \leq \|By_i\|$, we have also

$$|c(y_i)| \leq M_2.$$

By the choice of p , we know that all coefficients $\|By_i\|^2, y_{ij}^2, 2c(y_i)y_{ij}, 2y_{ij}y_{ik}$ of $\epsilon_j \epsilon_k$ in Eq. (4) are in the interval $(-\lfloor \frac{p}{2} \rfloor, \lfloor \frac{p}{2} \rfloor)$. Since $\epsilon_j \epsilon_k$'s are different powers of p , when we take $\|B_\epsilon y_i\|^2$ as a number with base p , it is easy to check that

$$\begin{aligned} \|B_\epsilon y_i\|^2 &< \epsilon_{n+1}^2 \|By_i\|^2 + \frac{1}{2} \epsilon_{n+1}^2 \\ &\leq \epsilon_{n+1}^2 (\lambda_l(\mathcal{L}(B))^2 + \frac{1}{2}). \end{aligned}$$

However, by Eq. (3), we know that

$$\begin{aligned} \|B_\epsilon x_l\|^2 &> \epsilon_{n+1}^2 \|Bx_l\|^2 - \frac{1}{2} \epsilon_{n+1}^2 \\ &\geq \epsilon_{n+1}^2 (\lambda_l(\mathcal{L}(B))^2 + 1 - \frac{1}{2}) \\ &= \epsilon_{n+1}^2 (\lambda_l(\mathcal{L}(B))^2 + \frac{1}{2}). \end{aligned}$$

Note that $B_\epsilon y_1, B_\epsilon y_2, \dots, B_\epsilon y_l$ are linearly independent, and $\lambda_l(\mathcal{L}(B_\epsilon)) = \|B_\epsilon x_l\| > \|B_\epsilon y_i\|$, $1 \leq i \leq l$, which leads to a contradiction to the definition of the successive minima. Hence, for $1 \leq i \leq n$, we have

$$\|Bx_i\| = \lambda_i(\mathcal{L}(B)).$$

- (4) Finally, we recover all successively minimal vectors as following. Querying the oracle \mathcal{O} with B_ϵ , we obtain $\lambda_i(\mathcal{L}(B_\epsilon)), 1 \leq i \leq n$. We next show we can efficiently find $x_i \in \mathbb{Z}^n$, such that $\|B_\epsilon x_i\| = \lambda_i(\mathcal{L}(B_\epsilon))$ by the value of $\lambda_i(\mathcal{L}(B_\epsilon))$ for $1 \leq i \leq n$.

Let $x_i = (x_{i1}, x_{i2}, \dots, x_{in})^T \in \mathbb{Z}^n$ satisfy

$$\lambda_i(\mathcal{L}(B_\epsilon))^2 = \|B_\epsilon x_i\|^2.$$

First note that $\log(\lambda_n(\mathcal{L}(B_\epsilon)))$ is bounded by $\text{poly}(m, n, \log M)$, and by Lemma 4, we know that $\log|x_{ij}|$ can also be bounded by $\text{poly}(m, n, \log M)$.

Second we expand $\|B_\epsilon x_i\|^2$ as follows:

$$\|B_\epsilon x_i\|^2 = \epsilon_{n+1}^2 \|Bx_i\|^2 + \sum_{j=1}^n x_{ij}^2 \epsilon_j^2 + \sum_{j=1}^n 2c(x_i)x_{ij}\epsilon_j\epsilon_{n+1} + \sum_{1 \leq j < k \leq n} 2x_{ij}x_{ik}\epsilon_j\epsilon_k. \quad (5)$$

Similarly, since $\|B_\epsilon x_i\| = \lambda_i(\mathcal{L}(B_\epsilon))$, we know that $\|Bx_i\| = \lambda_i(\mathcal{L}(B))$. As discussed in Step (3), all the coefficients $\|Bx_i\|^2, x_{ij}^2, 2c(x_i)x_{ij}, 2x_{ij}x_{ik}$ of $\epsilon_i\epsilon_j$ in Eq. (5) are in the interval $(-\lfloor \frac{p}{2} \rfloor, \lfloor \frac{p}{2} \rfloor]$. It is easy to recover all the coefficients in $\text{poly}(m, n, \log M)$ time by Lemma 2 in [16]. More precisely, we can recover all x_{ij}^2 and $x_{ij}x_{il}, j \neq l$ for each x_i . In fact for x_i , let $k_i = \min\{j|x_{ij} \neq 0\}$, and we can fix x_{ik_i} positive, that is $x_{ik_i} = \sqrt{x_{ik_i}^2}$. For the remaining x_{ij} , we can recover their absolute values according to x_{ij}^2 , and their signs according to the signs of $x_{ik_i}x_{ij}$. This can be done in $\text{poly}(m, n, \log M)$ time.

After recovering $x_1, x_2, \dots, x_n \in \mathbb{Z}^n$ such that $\|B_\epsilon x_i\| = \lambda_i(\mathcal{L}(B_\epsilon)), 1 \leq i \leq n$, we compute $Bx_1, Bx_2, \dots, Bx_n \in \mathcal{L}(B)$. Then they form a set of the successively minimal vectors in $\mathcal{L}(B)$.

All the reduction above is in $\text{poly}(m, n, \log M)$ time. The proof is completed. Hence, we finally have:

Corollary 1. *Search-SMP is equivalent to optimization-SMP.*

Remark 2. In our proof of the main theorem, we use the expansion of base p to recover all the successive minimal vectors. An obvious observation is that all the values $\|B_\epsilon x_i\|$ must be different since the same value must have the same expansion for base p in the interval $(-\lfloor \frac{p}{2} \rfloor, \lfloor \frac{p}{2} \rfloor]$. That is, when we add these errors to a given lattice basis B to transform it to be B_ϵ , all the successive minima $\lambda_i(\mathcal{L}(B_\epsilon))$ will be different.

4 Conclusions

In this paper, we revisit the problem SMP in lattices, and propose a rank-preserving reduction in polynomial time from search-SMP to optimization-SMP with only one call to the optimization-SMP oracle, which leads to the equivalence between search-SMP and its optimization version.

Acknowledgement. We very thank the anonymous referees for their valuable suggestions on how to improve the presentation of this paper.

References

1. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC 1996, pp. 99–108. ACM, New York (1996). <https://doi.org/10.1145/237814.237838>
2. Ajtai, M.: The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract). In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, STOC 1998, pp. 10–19. ACM, New York (1998). <https://doi.org/10.1145/276698.276705>
3. Ajtai, M., Dwork, C.: A public-key cryptosystem with worst-case/average-case equivalence. In: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, STOC 1997, pp. 284–293. ACM, New York (1997). <https://doi.org/10.1145/258533.258604>
4. Alkim, E., Bindel, N., Buchmann, J.A., Dagdelen, Ö., Schwabe, P.: TESLA: Tightly-Secure Efficient Signatures from Standard Lattices. IACR Cryptology ePrint Archive 2015, 755 (2015). <https://eprint.iacr.org/2015/755.pdf>
5. Blömer, J., Naewe, S.: Sampling methods for shortest vectors, closest vectors and successive minima. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 65–77. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73420-8_8
6. Boas, P.V.E.: Another NP-complete problem and the complexity of computing short vectors in lattices. Mathematics Department Report 81-04. University of Amsterdam (1981)
7. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. *SIAM J. Comput.* **43**(2), 831–871 (2014). <https://doi.org/10.1137/120868669>
8. Cai, J.Y., Nerurkar, A.: Approximating the SVP to within a factor $(1-1/\dim^\epsilon)$ is NP-hard under randomized conditions. In: Proceedings of the Thirteenth Annual IEEE Conference on Computational Complexity (Formerly: Structure in Complexity Theory Conference) (Cat. No.98CB36247), pp. 46–55, June 1998
9. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal Gaussians. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 40–56. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_3
10. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, STOC 2009, pp. 169–178. ACM, New York (2009). <https://doi.org/10.1145/1536414.1536440>
11. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, STOC 2008, pp. 197–206. ACM, New York (2008). <https://doi.org/10.1145/1374376.1374407>
12. Goldreich, O., Micciancio, D., Safra, S., Seifert, J.P.: Approximating shortest lattice vectors is not harder than approximating closest lattice vectors **71**, 55–61 (1999). <http://www.sciencedirect.com/science/article/pii/S0020019099000836>

13. Hoffstein, J., Howgrave-Graham, N., Pipher, J., Silverman, J.H., Whyte, W.: NTRUSign: digital signatures using the NTRU lattice. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 122–140. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36563-X_9
14. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: a ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0054868>
15. Horn, R.A., Johnson, C.R.: Matrix Analysis. Cambridge University Press, Cambridge (2012)
16. Hu, G., Pan, Y.: Improvements on reductions among different variants of SVP and CVP. In: Kim, Y., Lee, H., Perrig, A. (eds.) WISA 2013. LNCS, vol. 8267, pp. 39–51. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05149-9_3
17. Ipsen, I.C.F., Rehman, R.: Perturbation bounds for determinants and characteristic polynomials. SIAM J. Matrix Anal. Appl. **30**(2), 762–776 (2008). <https://doi.org/10.1137/070704770>
18. Kannan, R.: Minkowski’s convex body theorem and integer programming. Math. Oper. Res. **12**(3), 415–440 (1987). <https://doi.org/10.1287/moor.12.3.415>
19. Lyubashevsky, V., Micciancio, D.: Generalized compact knapsacks are collision resistant. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 144–155. Springer, Heidelberg (2006). https://doi.org/10.1007/11787006_13
20. Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: SWIFFT: a modest proposal for FFT hashing. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 54–72. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71039-4_4
21. Micciancio, D.: Efficient reductions among lattice problems. In: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, pp. 84–93. Society for Industrial and Applied Mathematics, Philadelphia (2008). <http://dl.acm.org/citation.cfm?id=1347082.1347092>
22. Micciancio, D., Goldwasser, S.: Complexity of Lattice Problems: A Cryptographic Perspective. The Kluwer International Series in Engineering and Computer Science, vol. 671. Kluwer Academic Publishers, Boston (2002)
23. Peikert, C., Rosen, A.: Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 145–166. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_8
24. Piazza, G., Politi, T.: An upper bound for the condition number of a matrix in spectral norm. J. Comput. Appl. Math. **143**(1), 141–144 (2002). <http://www.sciencedirect.com/science/article/pii/S0377042702003965>
25. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, STOC 2005, pp. 84–93. ACM, New York (2005). <https://doi.org/10.1145/1060590.1060603>



An Improved Algorithm to Solve the Systems of Univariate Modular Equations

Jingguo Bi^{1,2}, Mingqiang Wang^{3(✉)}, and Wei Wei⁴

¹ Institute for Advanced Study, Tsinghua University, Beijing 100084, China
jingguobi@mail.tsinghua.edu.cn

² State Key Laboratory of Cryptology, P. O. Box 5159, Beijing 100878, China

³ School of Mathematics, Shandong University,
Jinan 250100, People's Republic of China

wangmingqiang@sdu.edu.cn

⁴ China Information Technology Security Evaluation Center, Beijing 10085, China
weiweiunique@163.com

Abstract. In this paper, we propose an improved algorithm to solve the univariate modular equations with mutually co-prime moduli problem. This problem was first proposed in Håstad's original RSA broadcast attack. At PKC 2008, May and Ritzenhofen improved Håstad's result by using a slightly different transformation from polynomial systems to a single polynomial. In this work, we propose a new construction method to combine all the k equations into a single equation $f(x) \equiv 0 \pmod{\prod_{i=1}^k N_i}$. Our improved algorithm possesses two advantages compared with the two previous ones. Compared with Håstad's approach, our algorithm only needs fewer number of equations which suffice for a recovery of all common roots. Compared with May and Ritzenhofen's, our method obtains the single equation $f(x)$ with a smaller degree. The benefit is that this new algorithm will find the small solution x_0 more efficiently when we invoke Coppersmith's algorithm.

Keywords: RSA · Coppersmith's algorithm
Solving univariate modular equations

1 Introduction

The RSA cryptosystem [16] is the most widely used public key cryptosystem in practice, which was proposed by Rivest, Shamir and Adleman in 1978. Since its invention, it has attracted the interest of many cryptanalysts. In this paper, let us denote by $N = pq$ to be the RSA modulus with prime factors p and q . Usually, in the original RSA, $\gcd(p-1, q-1) = 2$. Let \mathbb{Z}_N denote the ring of integers modulo N . The public and private exponents are chosen to be inverses of each other modulo $\varphi(N) = (p-1)(q-1)$. Let e be the public exponent, and let $d = e^{-1} \pmod{\varphi(N)}$ be the private key.

Given an RSA modulus N , a public exponent e and a ciphertext $c \equiv m^e \pmod{N}$, find the corresponding plaintext m . This problem of computing e -th

roots modulo N is a well-known RSA problem. Obviously, if $m^e < N$, the equation does not only hold in \mathbb{Z}_N but over the integers, and then one can calculate m easily. This implies that encrypting small messages with small public exponents is insecure. The problem of computing e -th roots modulo N is directly related to the hardness of decrypting ciphertexts. How difficult of this problem is still open, partial results about the relationship (equivalence or inequivalence) in restricted models between the computing e -th roots modulo N problem and factorization problem were proposed in paper [4, 5, 11].

For the inhomogeneous case, i.e. given the equation $(\bar{m} + x)^e \equiv c \pmod{N}$, with \bar{m} denoting the known and x is the unknown part of the message, the question is how to find the unknown part of the plaintext x . Coppersmith [7] showed that given a composite integer N and a univariate polynomial $f(x) \in \mathbb{Z}_N[x]$ of degree δ , one can determine all zeros smaller than $N^{1/\delta}$ efficiently. Hence, $(\bar{m} + x)^e \equiv c \pmod{N}$ can be solved if $|x| < N^{1/e}$. It means that this inhomogeneous case can be solved efficiently under the same condition $x^e < N$, which is powerful if the unknowns are small enough on the partial key exposure attack on the RSA cryptosystem.

In the specific RSA broadcast scenario, the attacker can easily collect additional polynomials. Intuitionally, one can obtain further information with these additional polynomials. Specific to the problem of computing e -th roots modulo N , there are two variants of systems of polynomial modular equations, the same modulus or all moduli are different. In the former case, Coppersmith et al. [8] considered the situations that given two equations $f_1(x) \equiv 0 \pmod{N}$ and $f_2(x) \equiv 0 \pmod{N}$ in order to recover the common roots. Let a be the common solution of the two equations. Then, $f_1(x)$ and $f_2(x)$ share a factor $(x - a)$. Computing the greatest common divisor $\gcd(f_1(x), f_2(x)) \pmod{N}$ reveals this factor if it is the only common factor. The running time of this method is $O(\delta \log^2 \delta)$ where δ is the degree of the given polynomials. When the attacker obtains two RSA encryptions, under coprime public exponents (e_1, e_2) and a common modulus N is a special case of this setting. Namely, the attacker has to find the common root m of $f_1(x) = x^{e_1} - m^{e_1} \pmod{N}$ and $f_2(x) = x^{e_2} - m^{e_2} \pmod{N}$. Simmons [18] presented a neat attack for this special setting with running time polynomial in the bitlength of (e_1, e_2) . Precisely, one computes integers u_1, u_2 such that $u_1 e_1 + u_2 e_2 = 1$ with the help of the Extended Euclidean Algorithm. This gives us $m = (m^{e_1})^{u_1} (m^{e_2})^{u_2} \pmod{N}$.

Let us consider an analogue of Simmons attack in the setting of different RSA moduli. A user wishes to send the same message m to several participants having different moduli and using plain RSA encryption without padding techniques. In this paper, we focus on this attack under this motivating cryptographic application. Without loss of generality, we assume that all moduli are composite as modular equations over finite fields can be solved efficiently. We further assume that the $N_i, i = 1, \dots, k$, are relatively prime. Otherwise, we can compute prime factors of the N_i by computing the greatest common divisor.

For the arbitrary number of polynomial equations, May and Ritzenhofen [14] defined the problem of solving systems of modular univariate polynomial equations (SMUPE-problem), we recall this problem as follows.

Definition 1 (SMUPE-problem). Let $k \in \mathbb{N}$, and let $N_1, \dots, N_k \in \mathbb{N}$ be mutually co-prime composite numbers of unknown factorization. Suppose $N_1 < N_2 < \dots < N_k$. Let $f_1(x), f_2(x), \dots, f_k(x)$ be the polynomials with degree $\delta_1, \delta_2, \dots, \delta_k$ in $\mathbb{Z}_{N_1}[x], \mathbb{Z}_{N_2}[x], \dots, \mathbb{Z}_{N_k}[x]$, respectively. Let

$$\begin{cases} f_1(x) \equiv 0 \pmod{N_1} \\ f_2(x) \equiv 0 \pmod{N_2} \\ \vdots \\ f_k(x) \equiv 0 \pmod{N_k} \end{cases} \quad (1)$$

be a system of univariate polynomial equations. Let $X \leq N_1, X \in \mathbb{R}$. Find all the common roots x_0 of (1) with size $|x_0| < X$.

Our goal is to compute an upper bound X for which the SMUPE-problem is solvable in time polynomial in $\prod_{i=1}^k \delta_i$ and in the bitlength of $\prod_{i=1}^k N_i$. This upper bound will give us a condition on the number of equations k in terms of δ_i and N_i . This will enable us to compute the minimal k such that the SMUPE-problem can be computed up to the bound $X = N_1$, i.e. system (1) can be solved efficiently. The basic idea of solving the SMUPE-problem is how to find a transformation that reduced the SMUPE-problem to solving a single univariate polynomial equation with the same solutions.

Håstad [9, 10] gave the following algorithm for solving the SMUPE-problem. Let $\delta \in \mathbb{N}$ be the maximum degree of all polynomials occurring in the system, i. e. $\delta = \max_{i=1}^k \{\delta_i\}$. One first multiplies the given polynomials with $x^{\delta-\delta_i}$ to adjust their degrees. Then one combines the resulting polynomials using the Chinese Remainder Theorem to a univariate polynomial $f(x)$ with the same roots modulo $\prod_{i=1}^k N_i$. One only need to collect $k \geq \delta$ as a lower bound on the number of polynomials for efficiently finding all roots x_0 with $|x_0| < N_1$ by directly applying Coppersmiths lattice techniques [6, 7] to $f(x)$ (see e.g. [3]).

At PKC 08, May and Ritzenhofen [14] proposed a different construction to combine all the k polynomial equations into a single modular equation. Instead of multiplying the polynomials by powers of x like in Håstad’s approach, they took powers of $f_i(x)$ to solve the SMUPE-problem for all x_0 with $|x_0| < N_1$. More precisely, let $\delta = \text{lcm}_{i=1}^k \{\delta_i\}$, one first calculate the power of the given polynomials with $f_i(x)^{\delta/\delta_i}$ to adjust their degrees. After that, one can combine the resulting polynomials using the Chinese Remainder Theorem to a univariate polynomial $f(x)$ with the same roots modulo $\prod_{i=1}^k N_i^{\delta/\delta_i}$. Note that the degree of $f(x)$ is δ . One can obtain the solutions $|x_0| \leq N_1$ under the condition $\sum_{i=1}^k \frac{1}{\delta_i} \geq 1$ by invoking the Coppersmiths lattice techniques [6, 7] to $f(x)$. When all polynomials share the same degree δ , this corresponds to Håstad’s condition $k \geq \delta$. For polynomials of different degrees, the new condition is superior. In another word, a fewer polynomials of low degree suffice.

In this paper, we propose a new transformation method to combine all the k equations into a single modular equation $f(x)$ with a tighter condition. More specifically, in order to obtain the single modular equation, we not only take the powers of polynomials, but also multiply the power of x to adjust their

degrees. Let the degree of $f(x)$ is δ , one can obtain the solutions $|x_0| \leq N_1$ under the condition $\sum_{i=1}^k \lfloor \frac{\delta}{\delta_i} \rfloor \geq \delta$ by invoking the Coppersmiths lattice techniques [6, 7] to $f(x)$. Note that the modulus is $\prod_{i=1}^k N_i^{\lfloor \frac{\delta}{\delta_i} \rfloor}$. Hence, the new modular equation has much smaller degree and much less modulus compared with the single modular equation obtained by May and Ritzenhofen’s method. The benefit is that our method will spend much less time to find the small solution x_0 when we invoke Coppersmith’s algorithm. This speedup will become much more obvious when the degrees of equations are bigger. In Sect. 4, we propose a toy example to illustrate this improvement. Furthermore, we stress that our method also maintains the advantage of May and Ritzenhofen’s method. Compared with Håstad’s approach, our method only needs fewer number of equations which suffice for a recovery of all common roots.

The remainder of this paper is organized as follows. In Sect. 2, we review Coppersmiths result from [7] and the Chinese Remainder Theorem for polynomials. In Sect. 3, we show the new sufficient condition on the number of polynomials that is needed to recover all common roots efficiently. In Sect. 4, as an application, we show the improved RSA broadcast attack. Conclusions are finally drawn in Sect. 5.

2 Preliminary

In this paper, the main tool we will use is the polynomial algorithm to solve the small roots of a given modular univariate equations. In [7], Coppersmith showed how to provably determine zeros of modular univariate equations with sufficiently small size. More precisely, May’s recent survey [13] gives the time complexity upper bound $O(\delta^5 \log^9 N)^1$ for Coppersmith’s lattice-based algorithm by using the Nguyen-Stehlé L_2 algorithm [15] as the lattice reduction algorithm. At PKC 2014, Bi et.al. proposed a significant speedups over Coppersmiths algorithm by using the “rounding” and “chaining” technique [1].

Theorem 1 (Coppersmith [1]). *Let $f(x)$ be a monic polynomial of degree $\delta \in \mathbb{N}$ in one variable modulo an integer N of unknown factorization. Let X be a bound on the desired solution x_0 . If $X \leq N^{1/\delta}$, then we can find all integers x_0 such that $f(x_0) \equiv 0 \pmod{N}$ and $|x_0| \leq X$ in time $O(\log^7 N)$.*

The time complexity $O(\log^7 N)$ is obtained by utilizing the Nguyen-Stehlé L_2 algorithm [15] as the reduction algorithm. For the LLL lattice basis reduction step, please refer to [12, 13].

The basic idea of solving the SMUPE-problem is how to find a transformation that reduced the SMUPE-problem to solving a single univariate polynomial equation with the same solutions. A possible way to combine these equations is by Chinese Remaindering which is described e.g. in [10, 14, 17]. For the sake of completeness, we restate here.

¹ It is worth mentioning that the time complexity is not correct in the Theorem 1 of Sect. 2 in [14], neither is the time complexity estimation of solving the SMUPE-problem in [14].

Theorem 2 (*Chinese Remainder Theorem*). *Let $k \in \mathbb{Z}$. Let $\delta \in \mathbb{N}, \delta > 1$. For $i = 1, \dots, k$, let $N_i \in \mathbb{N}$ be pairwise relatively prime numbers, and let $f_i(x) \in \mathbb{Z}[x]$ be polynomials of degree δ . Then there exists a unique polynomial $f(x)$ modulo $M = \prod_{i=1}^k N_i$ such that $f(x) \equiv f_i(x) \pmod{N_i}$. The polynomial $f(x)$ can be determined in time $O(\delta \log^2 M)$.*

Proof. For $i = 1, \dots, k$, let $M = \prod_{i=1}^k N_i$, $M_i = \frac{M}{N_i}$ and M'_i be the inverse of M_i modulo N_i . The existence of such an inverse is guaranteed by $\gcd(M_i, N_i) = 1$. Then define

$$f(x) = \sum_{i=1}^k M_i M'_i f_i(x).$$

is the desired solution. If we look at $f(x)$ modulo N_j for $j \in \{1, \dots, k\}$, all summands with index $i \neq j$ cancel out (as N_j divides M_i) and $M_j M'_j f_j(x) \equiv f_j(x) \pmod{N_j}$.

Now suppose that $g(x)$ is another solution fulfilling the required conditions. Then, $f(x) - g(x) \equiv 0 \pmod{N_i}$ for all $i = 1, \dots, k$, and therefore also $f(x) \equiv g(x) \pmod{M}$.

Multiplication modulo M and calculating the inverses by the Extended Euclidean Algorithm can be performed in time $O(\log^2 M)$. Determining all coefficients of f then gives us $O(\delta \log^2 M)$ for the complete algorithm.

3 Main Result

For notational convenience, let us briefly recall the SMUPE-problem, let $k \in \mathbb{N}$, and let $N_1, \dots, N_k \in \mathbb{N}$ be mutually co-prime composite numbers of unknown factorization. Suppose $N_1 < N_2 < \dots < N_k$. Let $f_1(x), f_2(x), \dots, f_k(x)$ be the polynomials with degree $\delta_1, \delta_2, \dots, \delta_k$ in $\mathbb{Z}_{N_1}[x], \mathbb{Z}_{N_2}[x], \dots, \mathbb{Z}_{N_k}[x]$, respectively. Let

$$\begin{cases} f_1(x) \equiv 0 \pmod{N_1} \\ f_2(x) \equiv 0 \pmod{N_2} \\ \vdots \\ f_k(x) \equiv 0 \pmod{N_k} \end{cases} \tag{1}$$

be a system of univariate polynomial equations. Let $X \leq N_1, X \in \mathbb{R}$. Find all common roots x_0 of (1) with size $|x_0| < X$.

If we directly apply Coppersmiths method (Theorem 1) for the first equation in System (1), the small roots x_0 with $|x_0| < N_1^{1/\delta_1}$ can be found in polynomial time. By considering further equations this bound can be improved until all solutions can be found eventually.

Clearly, by Håstad's algorithm in combination with Theorem 1, the condition $k \geq \delta$ with $\delta = \max_{i=1}^k \{\delta_i\}$ is sufficient to solve a system of equations efficiently. However, this condition is clearly not optimal. Please see the trivial example shows in [14].

At PKC 08, May and Ritzenhofen [14] proposed a different construction to combine all k polynomial equations into a single equation $f(x) \equiv 0$

mod $\prod_{i=1}^k N_i$. Instead of multiplying the polynomials by powers of x like in Håstad's approach, they took powers of solving the SMUPE-problem for all x_0 with $|x_0| < N_1$. Let (f_i, δ_i, N_i) , $i = 1, 2, \dots, k$ be an instance of the SMUPE-problem with monic $f_i(x)$. Define $M = \prod_{i=1}^k N_i^{\delta/\delta_i}$ with $\delta = \text{lcm}(\delta_i, i = 1, 2, \dots, k)$. If one choose the smallest integer k such that $\sum_{i=1}^k \frac{1}{\delta_i} \geq 1$, then the SMUPE-problem can be solved only needs k equations in polynomial time.

Let $f(x)$ be a polynomial of degree δ and $f(x) \equiv 0 \pmod N$ for $N \in \mathbb{N}$. Choose an integer $m \in \mathbb{N}$, and then $g(x) = f^m(x) \equiv 0 \pmod{N^m}$. The solutions x with $|x| < N$ of the two equations remain unchanged. Moreover, with Coppersmiths algorithm (Theorem 1), one can determine those solutions for which the condition $|x| < N^{1/\delta} \iff |x| < (N^m)^{1/m\delta}$ holds. Thus, Coppersmiths bound is invariant under taking powers of the polynomial $f(x)$. As opposed to May and Ritzenhofen's approach, Håstad's algorithm does not take powers of the polynomials but only multiplications of polynomials with powers of x . This increases the degree of the polynomial but leaves the modulus unchanged. Let $f(x)$ be a polynomial of degree δ with $f(x) \equiv 0 \pmod N$ for $N \in \mathbb{N}$. Then with $\gamma > \delta$ the equation $g(x) = x^{\gamma-\delta}f(x) \equiv 0 \pmod N$ contains all the solutions x of $f(x)$ with $|x| < N$. However, applying Coppersmiths method to determine roots of $g(x)$, one only get roots x with $|x| < N^{1/\gamma} < N^{1/\delta}$. So obviously, Coppersmiths bound is not invariant under multiplication with powers of x . This explains why May and Ritzenhofen's work obtains a superior bound on the size of the roots.

In this paper, we propose a novel construction method to combine all the k equations into a single equation $f(x) \equiv 0 \pmod{\prod_{i=1}^k N_i}$ based on the crucial observation above. Now we state our main theorem and give a proof below. We will apply the similar technique that is used in Chap. 3 in [14].

Theorem 3. *Let $(f_i, \delta_i, N_i), i = 1, 2, \dots, k$ be an instance of the SMUPE-problem with monic $f_i(x)$. Define $\delta_{lcm} = \text{lcm}_{i=1}^k(\delta_i)$, and choose the integer*

$$\delta = \min\{\delta \in \{1, 2, \dots, \delta_{lcm}\}, \text{ s.t. } \sum_{i=1}^k \lfloor \delta/\delta_i \rfloor \geq \delta\}.$$

Let $M = \prod_{i=1}^k N_i^{\lfloor \delta/\delta_i \rfloor}$, then the SMUPE-problem can be solved in time $O(\log^7 M)$.

Proof. Let x_0 be a solution of the system of polynomial Eq. (1). Then x_0 is a solution of $f_i^{\lfloor \delta/\delta_i \rfloor}(x) \equiv 0 \pmod{N^{\lfloor \delta/\delta_i \rfloor}}$ for all $i = 1, \dots, k$.

Define

$$g_i(x) = x^{\delta \pmod{\delta_i}} \cdot f_i^{\lfloor \delta/\delta_i \rfloor}(x).$$

Then each of these equations $g_i(x)$ has a common degree δ and monic. Combining them by the Chinese Remaindered Theorem (Theorem 2), one can obtain a polynomial $f(x)$ of degree δ such that x_0 is a solution of $f(x) \equiv 0 \pmod M$ with $M = \prod_{i=1}^k N_i^{\lfloor \delta/\delta_i \rfloor}$.

Note that, $f(x)$ is still monic. For the coefficient a_δ of the monomial x^δ in $f(x)$ it holds that $a_\delta \equiv 1 \pmod{N_i^{\lfloor \delta/\delta_i \rfloor}}$ for all $i = 1, \dots, k$ and therefore $a_\delta \equiv 1 \pmod M$.

The above step can be performed in time $O(\delta \log^2 M)$ by the Chinese Remainder Theorem (Theorem 2).

By invoking Coppersmith's algorithm, one can find all the solutions x_0 of the above equation which fulfill

$$|x_0| \leq M^{1/\delta} = \left(\prod_{i=1}^k N_i^{\lfloor \delta/\delta_i \rfloor} \right)^{1/\delta}.$$

Under the condition $\sum_{i=1}^k \lfloor \delta/\delta_i \rfloor \geq \delta$, we have

$$N_1 \leq N_1^{(\sum_{i=1}^k \lfloor \frac{\delta}{\delta_i} \rfloor)/\delta} = \left(\prod_{i=1}^k N_1^{\lfloor \delta/\delta_i \rfloor} \right)^{1/\delta} \leq \left(\prod_{i=1}^k N_i^{\lfloor \delta/\delta_i \rfloor} \right)^{1/\delta}.$$

From Theorem 1, all the solutions $x_0, |x_0| \leq N_1$ can be found in time $O(\log^7 M)$.

Therefore, the SMUPE-problem can be solved in time $O(\log^7 (\prod_{i=1}^k N_i^{\lfloor \delta/\delta_i \rfloor}))$, with δ satisfies

$$\delta = \min\{\delta \in \{1, 2, \dots, \delta_{lcm}\}, s.t. \sum_{i=1}^k \lfloor \delta/\delta_i \rfloor \geq \delta\}.$$

A Toy Example

To illustrate the improvement of our new method, we propose a toy example in the following.

Let $N_1, \dots, N_4 \in \mathbb{N}$ be mutually co-prime composite numbers of unknown factorization and $N_1 < N_2 < N_3 < N_4$, take the following equations.

$$\begin{aligned} x^3 &\equiv c_1 \pmod{N_1} \\ x^3 &\equiv c_2 \pmod{N_2} \\ x^5 &\equiv c_3 \pmod{N_3} \\ x^7 &\equiv c_4 \pmod{N_4} \end{aligned}$$

The question is how can we determine all solutions smaller than N_1 ?

In Håstad's method, one should gather $k = \max\{3, 3, 5, 7\} = 7$ equations, and then utilize Chinese Remainder Theorem and Coppersmith's method to find the solutions x_0 satisfying $|x_0| \leq N_1$. Obviously, the only four equations given above is not enough.

Let consider May and Ritzenhofen's method. Firstly, one calculates $\delta = \text{lcm}\{3, 3, 5, 7\} = 105$, and then one can obtain

$$\begin{aligned} (x^3 - c_1)^{35} &\equiv 0 \pmod{N_1^{35}} \\ (x^3 - c_2)^{35} &\equiv 0 \pmod{N_2^{35}} \\ (x^5 - c_3)^{21} &\equiv 0 \pmod{N_3^{21}} \\ (x^7 - c_4)^{15} &\equiv 0 \pmod{N_4^{15}} \end{aligned}$$

Utilizing the Chinese Remainder Theorem, one can get an equation

$$x^{105} \equiv m_1C_1 + m_2C_2 + m_3C_3 + m_4C_4 \pmod{N}.$$

where $N = N_1^{35}N_2^{35}N_3^{21}N_4^{15}$, and the m_i are the coefficients from the Chinese Remainder Theorem, i.e. $m_i \equiv 1 \pmod{N_i}$, $m_i \equiv 0 \pmod{N_j}, j \neq i$. The C_i are the remainder algebraic terms except x^{105} in each equation. The above equation can be solved for x with $|x| \leq (N_1^{35}N_2^{35}N_3^{21}N_4^{15})^{1/105}$ by invoking Coppersmith's algorithm. This condition is fulfilled for any x with $|x| \leq N_1 = (N_1^{105})^{1/105} \leq (N_1^{35}N_2^{35}N_3^{21}N_4^{15})^{1/105}$. Therefore, from Theorem 1, one can determine all solutions of the above system of equations with the time complexity $O(\log^7(N_1^{35}N_2^{35}N_3^{21}N_4^{15}))$.

In our method, we firstly find the smallest integer $\delta' = 15$ satisfies the conditions of Theorem 3. Then we obtain

$$\begin{aligned} (x^3 - c_1)^5 &\equiv 0 \pmod{N_1^5} \\ (x^3 - c_2)^5 &\equiv 0 \pmod{N_2^5} \\ (x^5 - c_3)^3 &\equiv 0 \pmod{N_3^3} \\ x^{15} &\equiv x \cdot (2c_4x^7 - c_4^2) \pmod{N_4^2} \end{aligned}$$

Utilizing the Chinese Remainder Theorem, one can get an equation

$$x^{15} \equiv m_1C_1 + m_2C_2 + m_3C_3 + m_4x(2c_4x^7 - c_4^2) \pmod{N'}.$$

where $N' = N_1^5N_2^5N_3^3N_4^2$, and the m_i are the coefficients from the Chinese Remainder Theorem, i. e. $m_i \equiv 1 \pmod{N_i}$, $m_i \equiv 0 \pmod{N_j}, j \neq i$. where the C_1, C_2, C_3 are the remainder algebraic term except x^{15} in each equations. From Theorem 1, the above equation can be solved for x with $|x| \leq (N_1^5N_2^5N_3^3N_4^2)^{1/15}$ by invoking Coppersmith's algorithm. This condition is fulfilled for any x with $|x| \leq N_1 = (N_1^{15})^{1/15} \leq (N_1^5N_2^5N_3^3N_4^2)^{1/15}$. Therefore, one can determine all solutions of the above system of equations with time complexity $O(\log^7 N') = O(\log^7(N_1^5N_2^5N_3^3N_4^2))$. The speedup is at least $7^7 \approx 2^{20}$ compared with May and Ritzenhofen's method.

In order to obtain the single modular equation, we not only take the powers of polynomials, but also multiply the power of x to adjust their degrees. In practice, to determine the power of x in the final polynomial $f(x)$, one should traverse all the integer from $\min_{i=1}^k \{\delta_i\}$ to $\text{lcm}_{i=1}^k \{\delta_i\}$ to find the smallest integer satisfies $\sum_{i=1}^k \lfloor \delta/\delta_i \rfloor \geq \delta$. Compared with Håstad's approach, like May and Ritzenhofen's, our method also needs fewer number of equations which suffice for a recovery of all common roots, which means one can intercept and capture fewer ciphertxts in the practical attacks. Compared with May and Ritzenhofen's, our method obtains the single equation $f(x)$ with a much tighter condition. Hence, the new modular equation has much smaller degree and much less modulus compared with the single modular equation obtained by May and Ritzenhofen's method. The benefit is that our method will find the small solution more easily when we invoke Coppersmith's method to find the small solution x_0 . Note that this speedup will become much more obvious when the degrees of equations are bigger.

4 Application: RSA with Polynomially Related Messages

Take the public exponents are e_1, \dots, e_k and coprime public moduli are $N_1 < N_2 < \dots < N_k$. In order to avoid sending various encryptions of the same message, a user might add some randomness r_i and then encrypt the linearly related messages $(m + r_i), i = 1, 2, \dots, k$, instead of m . However, if the attacker gets to know the randomness, this setting is not secure either.

Theorem 4. *Let $k \in \mathbb{N}$, $(e_i, N_i), i = 1, 2, \dots, k$ be RSA public keys with $N_1 < N_2 < \dots < N_k$ and co-prime N_i . Furthermore, let $m \in \mathbb{Z}_{N_1}$ and let $g_i(x) \in \mathbb{Z}[x]$ be polynomials of degree $i \in \mathbb{N}$ with $a_{i\gamma_i}$ the coefficient of x^{γ_i} for $i = 1, \dots, k$. Let c_1, \dots, c_k be the RSA-encryptions of $g_i(m)$ under the public key (e_i, N_i) . Define $\delta_i = e_i\gamma_i$ and $M = \prod_{i=1}^k N_i^{\lceil \delta/\delta_i \rceil}$ with $\delta = \min\{\delta \in \{1, 2, \dots, \delta_{lcm}\}, s.t. \sum_{i=1}^k \lceil \delta/\delta_i \rceil \geq \delta\}$. Then an adversary can recover the message m in time $O(\log^t M)$ provided that*

$$\sum_{i=1}^k \lceil \delta/\delta_i \rceil \geq \delta.$$

Proof. Without loss of generality we assume that all $a_{i\gamma_i}$ are invertible modulo N_i . Otherwise, we can calculate $\gcd(a_{i\gamma_i}, N_i)$ to obtain the factorization of N_i for at least one $i \in \{1, 2, \dots, k\}$. Then we can calculate m by solving the roots of the univariate polynomial modulo the prime factors. This can be done efficiently (see [2]).

We are looking for a solution m of $f_i(x) = g_i(x)^{e_i} - c_i \equiv 0 \pmod{N_i}, i = 1, 2, \dots, k$. However, the polynomials $f_i(x)$ are not necessarily monic. Let $F_i(x) = a_{i,\gamma_i}^{-e_i}(g_i(x)^{e_i} - c_i) \pmod{N_i}, i = 1, 2, \dots, k$. Hence, $F_i(x)$ is a monic polynomial of degree $\delta_i = e_i\gamma_i$. Therefore, the theorem directly follows as an application of Theorem 3.

5 Conclusion

In this paper, we revisit the problem of polynomial time solving univariate modular equations with mutually co-prime moduli. Specifically, we propose a new construction to combine all the k equations into a single equation with a tighter condition. Our method possesses two advantages compared with the two previous methods. Compared with Håstad’s approach, our improved method only needs fewer number of equations which suffice for a recovery of all common roots. Compared with May and Ritzenhofen’s, our method obtain the single modular equation $f(x)$ with a much smaller degree and a much less modulus. The benefit is that our new method will find the small solution x_0 much more easily when we invoke Coppersmith’s algorithm.

Acknowledgments. This paper is partially supported by: NSF of China under grants No. 61502269 & 61672019, 973 Program grant 2013CB834205 and the national key research and development program of China grant No. 2017YFA0303903.

References

1. Bi, J., Coron, J.-S., Faugère, J.-C., Nguyen, P.Q., Renault, G., Zeitoun, R.: Rounding and chaining LLL: finding faster small roots of univariate polynomial congruences. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 185–202. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_11
2. Bach, E., Shallit, J.: Algorithmic Number Theory: Efficient Algorithms - Volume 1. The MIT Press, Cambridge (1996)
3. Boneh, D.: Twenty years of attacks on the RSA cryptosystem. Not. AMS **46**, 203–213 (1999)
4. Boneh, D., Venkatesan, R.: Breaking RSA may not be equivalent to factoring. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 59–71. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0054117>
5. Brown, D.: Breaking RSA may be as difficult as factoring. Cryptology ePrint Archive Report 2005/380 (2005)
6. Coppersmith, D.: Finding small solutions to small degree polynomials. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 20–31. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44670-2_3
7. Coppersmith, D.: Small solutions to polynomial equations and low exponent vulnerabilities. J. Cryptol. **10**(4), 223–260 (1997)
8. Coppersmith, D., Franklin, M., Patarin, J., Reiter, M.: Low-exponent rsa with related messages. In: Maurer, U. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 1–9. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68339-9_1
9. Hastad, J.: N using RSA with low exponent in a public key network. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 403–408. Springer, Heidelberg (1986). https://doi.org/10.1007/3-540-39799-X_29
10. Håstad, J.: Solving simultaneous modular equations of low degree. SIAM J. Comput. **17**(2), 336–341 (1988)
11. Leander, G., Rupp, A.: On the equivalence of RSA and factoring regarding generic ring algorithms. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 241–251. Springer, Heidelberg (2006). https://doi.org/10.1007/11935230_16
12. Lenstra, A.K., Lenstra, H.W., Lovsz, L.: Factoring polynomials with rational coefficients. Math. Ann. **261**, 513–534 (1982)
13. May, A.: Using LLL-reduction for solving RSA and factorization problems: a survey. In: LLL+25 Conference in Honour of the 25th Birthday of the LLL Algorithm (2007)
14. May, A., Ritzenhofen, M.: Solving systems of modular equations in one variable: how many RSA-encrypted messages does eve need to know? In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 37–46. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78440-1_3
15. Nguên, P.Q., Stehlé, D.: Floating-point LLL revisited. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 215–233. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_13
16. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978)
17. Shoup, V.: A Computational Introduction to Number Theory and Algebra. Cambridge University Press, Cambridge (2005)
18. Simmons, G.: A weak privacy protocol using the RSA crypto algorithm. Cryptologia **7**(2), 180–182 (1983)



O²TR: Offline Off-the-Record (OTR) Messaging

Mahdi Daghmehchi Firoozjaei, Sang Min Lee, and Hyounghshick Kim^(✉)

Electrical and Computer Engineering Department, Sungkyunkwan University,
Suwon 440-746, Republic of Korea
{mdaghmechi, sangmin91, hyoung}@skku.edu
<http://seclab.skku.edu/>

Abstract. Off-the-record (OTR) is a security protocol that can be used in privacy preserving instant messaging (IM) systems. However, the conventional OTR is not applicable in some practical scenarios (e.g., when communication network became disconnected) because OTR requires both parties to be online at the same time. To address this limitation, we extend the conventional OTR into a new protocol named offline OTR (O²TR). O²TR makes the conversation parties be able to handle an offline message even when a session connection is lost. To show the feasibility of the proposed protocol, we implemented a prototype to support O²TR based on the Gajim XMPP instant messaging platform. Our experiments showed that O²TR can reliably be used when a network party is broken down. Moreover, O²TR provides an efficient session refreshment which is about 34% faster than the original OTR.

Keywords: Key exchange · OTR · Instant messaging

1 Introduction

Off-the-record (OTR) is a security protocol to make a private session over an instant messaging (IM) system [1]. OTR provides end-to-end confidentiality and integrity of the transmitted messages by using encryption, authentication, perfect forward secrecy (PFS), and deniability services. In OTR systems, ephemeral keys are newly established for every session by updating key parameters continuously. These features make the OTR protocol a suitable solution for privacy preserving in the online services. By the deniability feature, no one, including the sender, can prove who authored a particular message in the transcript [2, 3]. On the other hand, PFS makes it impossible to compromise the past session keys even when long-term keys are compromised. Basically, OTR mimics the features of an actual face-to-face conversation into the virtual environment for deniability service [2]. Server-based security solutions establish private sessions for IM clients through a central server while OTR establishes secure sessions without relying on a central server; after establishing secure sessions, OTR provides a secure environment for pairwise encryption and authentication.

In general, asynchronous communication, where the sending and receiving of a message do not need to happen simultaneously, is a usual phenomenon in the most IM systems. In the central server-based security protocols, by providing the confidentiality between client and server, there is no issue to handle offline messages. For other security mechanisms, such as OTR, which are not server-based, there is a different condition. To start a private session, OTR requires both conversation parties to be online in order to exchange their Diffie-Hellman (DH) parameters for the key exchange. Furthermore, the encryption and authentication keys in each OTR data packet are securely chained to the previous keys. Based on the spirit of OTR, this protocol is only applicable to synchronous communications and does not support the offline messages. Although this condition does not challenge OTR privacy provision, its practicality can be limited by this constraint.

To address the offline message limitation, we extend the original OTR into a new protocol named offline OTR (O²TR). Basically, the original OTR is based on a mechanism of advertising keys and receiving confirmations for those keys in subsequent messages [4]. After establishing a private session, OTR parties exchange their next ephemeral keys in each data packet. In fact, the keys in OTR are chained together. The proposed O²TR utilizes this feature to handle offline messages. Our experiments showed the feasibility of O²TR to handle offline messages. Although processing time to handle O²TR messages is similar to OTR's, O²TR provides faster private session refreshment. The main contributions of this paper are summarized as follows:

- In O²TR, we introduce a solution to detect and store the latest keys of the private session to handle offline messages. We develop a mechanism to renew and replace the stored keys based on the incoming and outgoing messages. By detecting an offline message, the stored keys of the previous session are used to handle the message.
- To evaluate the performance of our model in a real network environment, we implemented O²TR on Gajim IM system launched on an XMPP/Jabber platform.

The rest of this paper is organized as follows. In Sect. 2, an overview of OTR and the offline message issue are presented. Section 3 provides a model explanation of O²TR. The implementation of O²TR and the evaluation results are respectively explained in Sects. 4 and 5. In Sect. 6, we analyze the security of O²TR and our conclusions are summarized in Sect. 7.

2 Overview of OTR

The protocol of OTR was initially introduced by Borisov et al. [1] in 2004. OTR is built on a high-level cryptographic abstraction, such that AES (128-bit), SHA-256 and SHA-1 hash functions are used in the last version of OTR (Ver. 3). SHA-256 is used in the handshake process, and to calculate the encryption and authentication keys based on a DH share. On the other hand, SHA-1 is used as

an HMAC function to authenticate each encrypted message [5]. Therefore, all encrypted messages are authenticated by the HMAC function. Using encrypted DH key exchange and keys derived from a shared secret to encrypt each message and to authenticate each ciphertext lead to building public key authenticated encryption on top of lower-level primitives [6].

To establish a private session, the conversation parties share their DH keys by OTR handshake protocol. A version of SIGMA is used as the authenticated key exchange (AKE) to exchange the generated keys. To prevent the man-in-the-middle (MitM) attack, no clear DH encryption key is revealed in the first AKE message. After exchanging the DH keys in the first two messages, the conversation parties authenticate each other by their public keys in an encrypted channel. These keys are long-lived DSA public keys and are used only for authentication [5, 7]. After the authentication process, the private session is established and the OTR parties are ready to exchange data packets.

Figure 1 shows the format of an OTR data packet. In this packet, the header field shows the version of OTR protocol, message type, the sender and the receiver of the message, and the required flags. As mentioned before, OTR exploits the ephemeral keys for encryption to provide PFS service. Since the keys are changed per message, *keyids* are used to select the latest DH keys and to make sure that a unique set of keys is being used [7]. In fact, a *keyid* is an integer number that is increased by generating a new DH key.

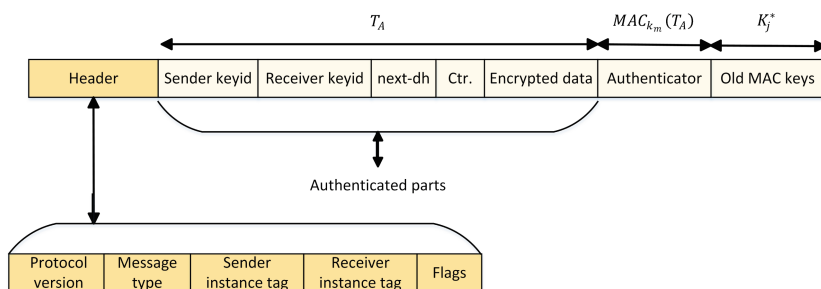


Fig. 1. OTR data packet format

Since the encryption key is pairwise between two OTR parties, in each data packet a new DH public key is suggested by the sender for the next data message. The suggested key in the *next.dh* field will be used for the next message after receiving its confirmation. DH share is computed based on both parties' DH keys to generate the encryption and authentication keys. AES algorithm in CTR mode is used to encrypt/decrypt the messages. The *Encrypted data* field consists of the encrypted message. In the rest fields, the MAC value and the used MAC keys are revealed. By revealing the MAC keys one round later to the public, OTR provides message deniability [8]. Therefore, once the MAC keys are revealed, anyone can modify the message and the receiver cannot prove the sender's authorship.

2.1 OTR and Offline Messages

The offline message problem occurs when a conversation party in an OTR private session goes offline or leaves the IM chat room. Due to the structure of OTR to exploit the pairwise encryption, it is only applicable in the synchronous communications where both the communicating parties are online [9]. During an OTR conversation, if the receiver party gets offline any undelivered message remains in the IM server side as an offline message. The offline message will be delivered whenever the receiver shows up. Since the offline message is encrypted based on the last lost private session, it is unreadable for the receiver.

As shown in Fig. 1, in OTR data packet, the next ephemeral DH share (in the *next_dh* field) is protected with a MAC computed with the current key. Until being acknowledged by the recipient this new DH share cannot be used by the sender [9]. Therefore, to pursue the key generation chain in OTR, both conversation parties require to be online. This requirement makes OTR impracticable in the asynchronous communication.

In OTR, the authentication and the message states indicate the current state of each OTR conversation. The authentication state shows the authentication progress, while the message state indicates the ciphering state of the outgoing messages [5]. To prevent any flaw and inconsistency for sending messages, the message state controls what happens during the OTR conversation. For instance, if during a private session a party logs out his OTR client the correspondent will be notified. This notification prevents to unwillingly send any message in plaintext mode [5].

Since the liveness of an OTR private session depends on the running IM chat room, the message state mismatching occurs when a client leaves the chat room while its OTR session is not finished. In this case, the current OTR private session is not terminated on the other side. Therefore, any message sent by the correspondent is based on the previous unfinished private session. The unreadability error will happen when the client receives an offline message from the previous unfinished private session.

As an example of this mismatching, Fig. 2 shows the related OTR error message in a Pidgin-OTR messaging client. During an OTR conversation one party, *mdf3@localhost*, gets offline and leaves the chat room. After showing up and connecting back to the Pidgin IM server, an offline message that was sent by *mdf2@localhost* and has remained on the server is delivered to *mdf3@localhost*. Since there is no private session and this OTR message was encrypted with the last session, it is not readable.

To prevent any OTR parameter mismatching in the asynchronous communication, we introduce O²TR to handle offline messages. As stated earlier in the Sect. 2, by sending or receiving an OTR message, all keys are regenerated, based on the new DH shared key. Therefore, each party knows the required keys of the next message. By retaining these keys in O²TR, we are able to retrieve any offline OTR message's required keys. In this case, an O²TR party can authenticate and decrypt the offline message that was created in the last session.

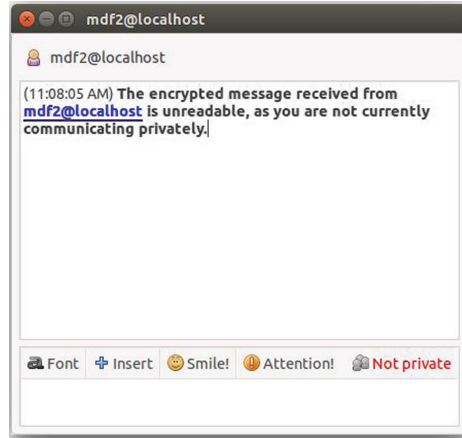


Fig. 2. The error message for receiving an offline message in the Pidgin-OTR IM client

3 Offline OTR- O²TR

In order to achieve PFS, OTR exploits the ephemeral keys to renew the encryption and authentication keys for every data packet. To this end, each OTR party generates a new DH key and shares it in the *next_dh* field of the data packet. Therefore, each party knows the next DH key of the correspondent. Based on this fact, saving the next DH key for every packet is the basic concept of O²TR.

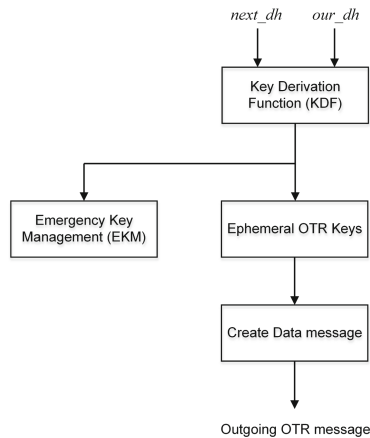


Fig. 3. Storing the latest keys in EKM

Basically, the O²TR model consists of two extra parts in comparison to the original OTR; emergency key management (EKM) and offline message detection

(OMD). As shown in Fig. 3, all ephemeral keys for encryption and authentication generated by the key derivation function (KDF) are saved in EKM too. These keys are generated based on the DH keys of both parties. Since new keys are generated for each message, the registered keys in EKM are replaced with the new keys. Therefore, in each conversation EKM consists of the latest keys in both O²TR parties. Unlike the ephemeral keys, EKM retains the registered keys even if the client leaves the chat room.

To handle offline messages, in O²TR model, OMD checks the state of the incoming message. In the case of offline message, an encrypted message is delivered to a client with no private session. Therefore, the state of an offline message does not match with the self state of the receiver that is in the plaintext mode. As shown in Fig. 4, by detecting this mismatching by OMD, the message is considered as an offline message, and is handled by EKM stored keys instead of rejecting.

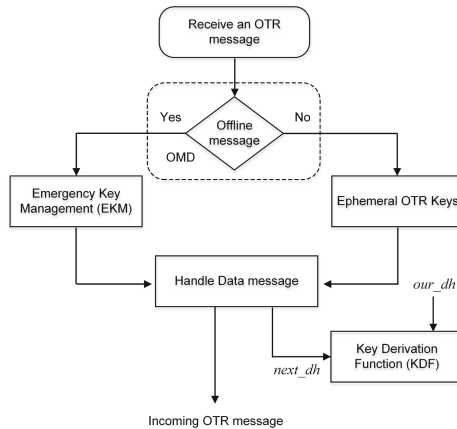


Fig. 4. Handling the offline message in O²TR

Since the offline message is created by the last unfinished O²TR session, no MAC keys are released and they are valid yet. Therefore, it is impossible to compromise this offline message's validity before releasing its MAC key in the next O²TR data message [5].

4 Implementation

To analyze the performance of O²TR, Gajim¹ messaging application was selected as an IM system. Furthermore, eJabberd XMPP² server, implemented in Ubuntu 14.04, was used to provide an XMPP/Jabber platform to launch the Gajim IM.

¹ <https://gajim.org/index.php?lang=en>.

² <https://www.process-one.net/en/ejabberd/>.

Due to the abilities of eJabberd to provide secure client-to-server (C2S) connections (i.e., SSL/TLS connection) and a storage for undelivered messages, this server was selected for our implementation. Figure 5 shows a typical IM connection through an eJabberd server.

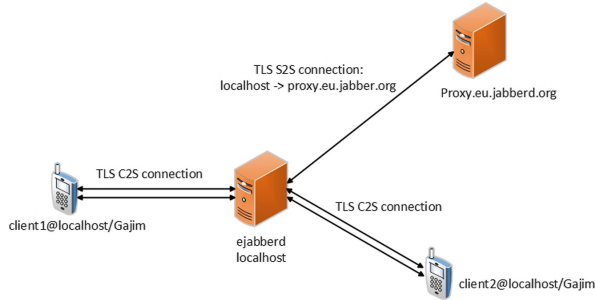


Fig. 5. IM connection through an eJabberd XMPP sever

To enable OTR for the Gajim XMPP/Jabber client, we used OTR plugin provided by K. Braden available on GitHub³. Furthermore, the Pure-Python-OTR, a Python OTR implementation package provided by K. Braden, available on GitHub⁴ was installed to enable this OTR plugin. To implement O²TR in the Gajim XMPP/Jabber client, we optimized the Pure-Python-OTR package and OTR plugin to add EKM and OMD modules.

5 Experiments

To evaluate and scrutinize the performance of O²TR model, the concepts of offline message recovery and processing time for message handling were considered. These items were scrutinized on the IM communication between Gajim XMPP/Jabber clients in two models: OTR and O²TR.

5.1 Offline Message Recovery

To evaluate the ability of O²TR to recover the offline message, we considered a condition that led to disconnect a normal Gajim-OTR conversation. That event was carried out for more than 200 separate Gajim IM conversations. As the results, all offline messages (100%) were recovered and decrypted completely with Gajim-O²TR while it was not possible for the normal Gajim-OTR model. Figure 6 shows the offline message delivery capability in Gajim-O²TR IM conversation.

³ <https://github.com/python-otr/gajim-otr>.

⁴ <https://github.com/python-otr/pure-python-otr>.

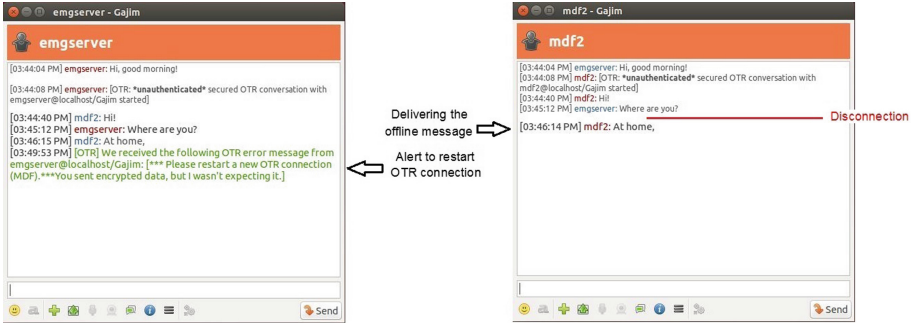


Fig. 6. Retrieving offline message in Gajim-O²TR

5.2 Processing Time

To provide a practical comparison, we evaluated the processing time of message exchange, handshake, and session refreshment in both protocols (O²TR and OTR). As shown in Fig. 7, there are no meaningful differences between processing time to handle the incoming and outgoing messages in OTR and O²TR protocols. Unlike sending outgoing messages, handling incoming messages takes more time due to performing some additional tasks, such as scrutinizing the current policies (accepting encrypted or non-encrypted messages), message type (data packet or AKE), OTR’s parameters (version and validity), key IDs, and authenticate the received message.

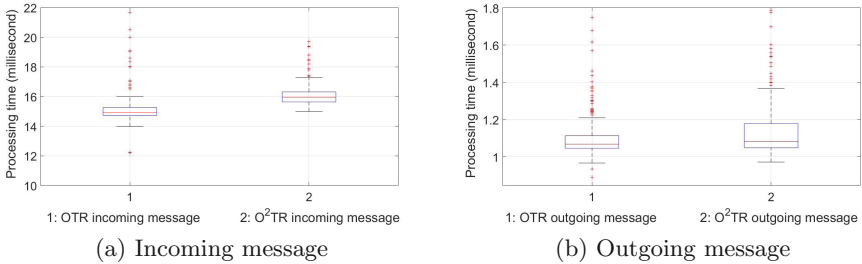


Fig. 7. Processing time to handle the incoming message (a) and outgoing message (b) in OTR and O²TR

Table 1 compares the average processing time to handle all messages in OTR and O²TR protocols. The average processing time to handle incoming and outgoing messages is almost same in both protocols. Based on these results, O²TR does not impose significant processing time to IM clients. To handle offline messages, which is only possible in O²TR, averagely 29.95 ms is needed for packet processing. In comparison to incoming message handling in OTR, this amount

is almost two times longer. It should be noted that this time is only spent for the first offline message in which the required keys are extracted from the local memory.

Table 1. Average processing time to handle incoming, outgoing, and offline message in OTR and O²TR

Message type	OTR	O ² TR
Incoming message	15.15 ms	16.07 ms
Outgoing message	1.11 ms	1.14 ms
Offline message-first	—	29.95 ms

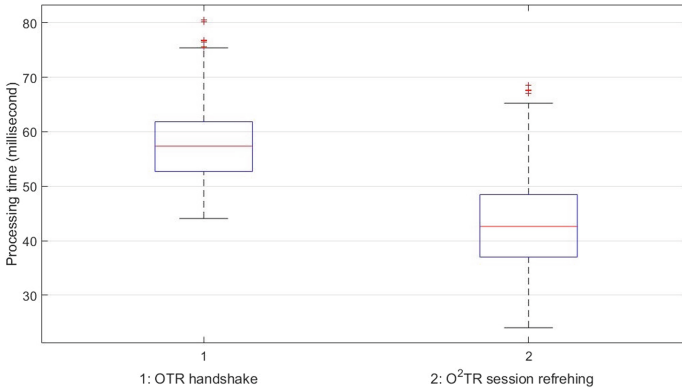


Fig. 8. Processing time for handshaking in OTR and session refreshment in O²TR

On the other hand, O²TR provides a condition to refresh the private session when an offline message is received. Therefore, the private session is refreshed with no interrupt instead of starting a new OTR handshake which is longer. Figure 8 shows the processing time for OTR handshaking and to refresh the private session in O²TR. Averagely, an OTR handshake takes 58.29ms while the refreshment of a private session takes 43.49ms in O²TR. Based on these results, in O²TR a private session is refreshed 34% faster than creating a new private session in OTR. Consequently, in a private session disconnection, O²TR not only retrieves offline messages but also refreshes the interrupted private session faster.

6 Security Analysis

As one of the basic security properties of OTR, PFS feature assures to protect session keys in which case leads to compromising server’s private key. Based on

the chain of key generation in OTR, each data packet consists of a new DH key suggested by the sender, which will be used for the next message. It's worth noting that, until the receiver has acknowledged this advertised next key, it can't be used. If an OTR party needs to send multiple messages before receiving any replies, he/she will need to keep using the current key and advertising the same next key [4]. This condition leads to a possible flaw when a curious IM server as an active attacker filters the packets and forces the client to send all messages with the same one key.

Since all packets are encrypted, the success of the attacker to detect the ciphering keys depends on the possibility of breaking AES-CTR protocol. To prevent any key leakage possibility, we define a threshold (e.g., $k = 5$) to use an encryption key to do multiple encryption rounds. Therefore, no O²TR party can use the same encryption key more than k times for multiple messages. Otherwise, the sending is paused until receiving a reply or refreshing to a new private session.

On the other hand, since no ephemeral keys are reused, the property of PFS is not compromised. Based on this fact, the attacker, the curious server in our adversary model, has no chance to detect the private session key even by logging the revealed keys. Consequently, any kind of the replay attack is not practical to compromise O²TR protocol as well as OTR. Finally, after retrieving the offline message the private session will be refreshed by the receiver. In this state, the same security level as OTR's is provided for the conversation parties.

7 Conclusions

We proposed O²TR, an offline OTR protocol, that makes the conversation parties be able to handle offline messages. O²TR utilizes EKM and OMD parts to save the latest keys, detect and retrieve the offline message even after leaving the chat room. To prevent any key leakage flaw, we set a threshold to use an encryption key to do multiple encryption rounds. Our experiments showed the complete capability to handle offline messages under different conditions. It's worth noting that, O²TR does not impose significant processing time to IM clients. Furthermore, O²TR provides an efficient session refreshment which is about 34% faster than creating a new private session in OTR.

Acknowledgments. This work was supported in part by the IITP (No. B0717-16-0116), the ITRC (IITP-2017-2015-0-00403) and the MISP (R2215-16-1005). The authors would like to thank all the anonymous reviewers for their valuable feedback.

References

1. Borisov, N., Goldberg, I., Brewer, E.: Off-the-record communication, or, why not to use PGP. In: Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society, WPES 2004, pp. 77–84. ACM (2004)
2. Liu, H., Vasserman, E.Y., Hopper, N.: Improved group off-the-record messaging. In: Proceedings of the 12th ACM Workshop on Privacy in the Electronic Society, WPES 2013, pp. 249–254. ACM (2013)

3. Di Raimondo, M., Gennaro, R., Krawczyk, H.: Secure off-the-record messaging. In: Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, WPES 2005, pp. 81–89. ACM (2005)
4. Marlinspike, M.: Advanced cryptographic ratcheting. Open Whisper Systems (2013). <https://whispersystems.org/blog/advanced-ratcheting/>
5. Goldberg, I., Goulet, D., Bergen, J.V.: Off-the-record messaging protocol version 3. <https://otr.cypherpunks.ca/Protocol-v3-4.1.1.html>
6. Petullo, W.M., Zhang, X., Solworth, J.A., Bernstein, D.J., Lange, T.: MinimalLT: minimal-latency networking through better security. In: Proceedings of the 2013 ACM SIGSAC Conference on Compute, Communications Security, CCS 2013, pp. 425–438. ACM (2013)
7. Alexander, C., Goldberg, I.: Improved user authentication in off-the-record messaging. In: Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society, WPES 2007, pp. 41–47. ACM (2007)
8. Goldberg, I., Ustaoglu, B., Van Gundy, M.D., Chen, H.: Multi-party off-the-record messaging. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS 2009, pp. 358–368. ACM (2009)
9. Frosch, T., Mainka, C., Bader, C., Bergsma, F., Schwenk, J., Holz, T.: How secure is textsecure? Cryptology ePrint Archive, Report 2014/904 (2014). <http://eprint.iacr.org/2014/904>



ARM/NEON Co-design of Multiplication/Squaring

Hwajeong Seo¹, Taehwan Park², Janghyun Ji², Zhi Hu³, and Howon Kim²(✉)

¹ Hansung University, Seoul, Republic of Korea
hwajeong84@gmail.com

² Pusan National University, Busan, Republic of Korea
{pth5804, jjh0819, howonkim}@pusan.ac.kr

³ Central South University, Changsha, China
huzhi_math@csu.edu.cn

Abstract. Many modern mobile processors support new SIMD extensions (e.g. NEON engine) and previous applications (e.g. image processing, cryptography) written in SISD are accelerated by re-writing the previous implementations in SIMD instruction sets. Particularly, integer multiplication and squaring operations are the most expensive in Public Key Cryptography (PKC). Many works have been conducted to reduce the execution timing in NEON instruction set. However, ARM-NEON processor also supports powerful ARM instruction set as well. By exploiting the ARM instruction together with NEON engine, we can achieve further improved performance. After this observation, we introduce new parallel approach for integer multiplication and squaring operations on ARM-NEON processors. Unlike previous implementations, we mix-use both ARM and NEON instructions to hide computation latency for ARM into NEON. Since ARM and NEON modules are separated units, the assignments are successfully issued independently. The integer multiplication and squaring are finely divided into several sub-tasks and the sub-tasks are properly assigned to ARM and NEON in order to balance the workloads. Finally, the proposed implementations outperform the best-known results on the identical ARM-NEON processors by 22.4% and 18.3% for 2048-bit integer multiplication and squaring, respectively.

Keywords: Parallel implementation · ARM · NEON · Co-design
Multiplication · Squaring · Public key cryptography

This work was supported by the Energy Efficiency & Resources Core Technology Program of the Korea Institute of Energy Technology Evaluation and Planning (KETEP), granted financial resource from the Ministry of Trade, Industry & Energy, Republic of Korea. (No. 20152000000170). Hwajeong Seo was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2014-0-00743) supervised by the IITP (Institute for Information & communications Technology Promotion). Zhi Hu was partially supported by the Natural Science Foundation of China (Grant No. 61602526).

1 Introduction

Modern high-end embedded processors have started to employ advanced Single Instruction Multiple Data (SIMD) architecture, which can efficiently perform multiple data in parallel way. When it comes to modern ARM processors, they provide new SIMD extensions (e.g. NEON engine), which support 128-bit registers and vectorized instructions. This nice property introduces new direction to enhance the performance of cryptography applications by rewriting the traditional ARM instruction sets into NEON instruction sets. The first NEON based cryptography implementation belongs to Bernstein and Schwabe in CHES'12 [1]. They introduced implementations of ARX based stream cipher (Salsa) and elliptic curve cryptography (Curve25519) by taking advantages of specific NEON instruction sets and general purpose registers. Afterward, a number of papers have introduced new vectorized implementation techniques to enhance the speed factor by multiple times from traditional implementations [2, 4–6, 8–10]. For high-speed Public Key Cryptography (PKC), previous works mainly focused on optimizations of integer multiplication and squaring operations since both operations are the largest overheads in both RSA and ECC implementations. Particularly, over SIMD instruction sets, two different approaches are usually considered to perform integer multiplication and squaring operations. First approach is the reduced-radix representation based computation. Unlike traditional Arithmetic Logic Unit (ALU), SIMD architecture does not provide status flag registers, which represent current status after certain instruction. For this reason, SIMD instruction cannot reserve overflow/underflow conditions due to the absence of status flag registers. In order to avoid these shortcomings, the reduced-radix representation leaves part of register bits to retain carry and borrow bits. Since these carry and borrow bits are only handled once at the last round of computation, the middle round can perform computations without considerations on carry/borrow bits. Furthermore, this does not require modular reduction after addition and subtraction operations. The optimal approach accelerates many ECC implementations on ARM–NEON processor [1, 6, 10]. However, the reduced-radix representation requires to compute more number of partial products than the non-reduced-radix representation, because the reduced-radix representation only utilizes small bits (e.g. radix 2^{16} – 2^{28}) in 32-bit register. These representations require more number of words to store the same length of variables than non-reduced-radix representation. For this reason, the long integer multiplication/squaring operations (e.g. RSA, SIDH) usually adopt non-reduced-radix representation [2–5, 8, 9]. In ICISC'14, ARM–NEON specialized multiplication technique, 2-way Cascade Operand Scanning (COS), was presented [8]. This method performs the partial products in a non-conventional order to reduce the number of data-dependencies in the carry propagations. The COS method operates on 32-bit words in a row-wise fashion (similar to the operand-scanning method) and does not require a “non-canonical” representation of operands with a reduced radix. For modular multiplication, two COS computations can be “coarsely” integrated into an efficient vectorized variant of Montgomery multiplication, namely Coarsely Integrated Cascade Operand Scan-

ning (CICOS) method. Recently, Seo et al. suggested Double Operand Scanning (DOS) for squaring operation, which doubles the operand instead of the intermediate result to get the doubled intermediate results [9]. Afterward, they applied the Karatsuba techniques to both integer multiplication and squaring operations to enhance the performance for long integer cases. For modular multiplication/squaring, separated Montgomery algorithm is selected since Karatsuba technique is hard to be integrated together with multiplication/squaring and reduction. Finally, proposed modular multiplication/squaring operations based RSA implementations outperform the best-known results on the ARM-NEON platforms.

In this paper, we further improve the previous integer multiplication and squaring techniques (COS and DOS) by mix-using ARM and NEON instruction sets to hide latencies for ARM instructions into NEON instructions. The proposed mixed approach efficiently optimizes the execution timing since ARM and NEON instruction sets are performed in parallel way. The proposed methods outperform the best-known results on the identical ARM-NEON processors by 22.4% and 18.3% for 2048-bit integer multiplication and squaring, respectively [9].

Summary of Research Contributions

The contributions of our work are summarized as follows.

1. *ARM/NEON co-design of Cascade Operand Scanning based multiplication.* Novel co-design of COS multiplication technique requires 10,657 clock cycles for 2,048-bit integer multiplication on Cortex-A9 processor. The optimized implementation is faster than state-of-art results by 22.4% [9]. The detailed descriptions can be found in Sect. 2.1 and performance comparison is drawn in Table 2.
2. *ARM/NEON co-design of Double Operand Scanning based squaring.* Similarly, we proposed novel co-design of DOS squaring technique on ARM-NEON processor. The high performance enhancements are also observed in the integer squaring by finely combining ARM and NEON instruction sets. The detailed descriptions and performance evaluations can be found in Sect. 2.2 and Table 2, respectively.

The remainder of this paper is organized as follows. In Sect. 2, we introduce new ARM/NEON co-design of integer multiplication and squaring operations for ARM-NEON processors. In Sect. 3, we summarize our experimental results and compare with the state-of-art works. In Sect. 4, we conclude the paper.

2 Proposed Methods

In this section, we introduce the efficient implementation techniques for integer multiplication and squaring operations on ARM-NEON processors. In order

to improve the performance of both implementations, we present optimized ARM/NEON co-design to perform the sub-routines of multiplication/squaring in parallel way. Unlike traditional integer multiplication/squaring implementations on NEON platforms, we mix-use both ARM and NEON instructions to hide latencies of ARM instruction into NEON instruction. This finely scheduled instruction sets allow 22.4% and 18.3% enhancements for 2048-bit integer multiplication and squaring operations than the state of art work, respectively. The source codes for proposed methods will be available in public domain for any public and private purposes.

2.1 Multiplication

Previous implementations of integer multiplication only utilize NEON instruction sets to achieve high performance on the ARM-NEON processor since NEON engine has powerful vectorized features including SIMD instruction sets (e.g. 2 32-bit integer multiplication and 2 64-bit integer addition) to perform multiple operations and long general purpose registers ($2048 = 16 \times 128$ -bit) to retain operands and intermediate results. Furthermore, the data transfer between ARM and NEON registers causes huge overheads from pipeline stalls. For this reason, ARM or NEON instruction is solely utilized in previous works. However, Seo et al. proves that both ARM and NEON instruction sets can be mix-used to reduce the execution timing for LEA block cipher encryption in WISA'16 [11]. Even though ARM module provides relatively less powerful instruction sets than that of NEON, ARM has enough functionality to compute small tasks in reasonable timing. The author shows that independent tasks can be performed in both ARM and NEON, simultaneously.

Inspired by previous parallel approach, we introduce new multiplication techniques to use full functionalities of ARM-NEON processors. In the previous works, the author performs completely independent tasks (encryptions) in data-parallelism. This is easy to adopt parallelism since the tasks do not interfere each other and do not cause pipeline stalls. On the other hand, our multiplication implementation uses task-parallelism, since unlike block cipher encryption, long integer multiplication requires computations on long length operands. However, straight-forward task-parallelism is not available in integer multiplication, since the intermediate results should be accumulated from the least significant word to the most significant word to optimize the carry propagation.

In order to achieve the task-parallelism for integer multiplication, we divide the whole operations into four sub-sections. Afterward, we assigned one sub-section to ARM module and three sub-sections to NEON engine. Both ARM and NEON routines are performed in parallel way and the workloads for ARM instruction are hidden into NEON computations¹. Particularly, for integer multiplication in ARM module, total 9 registers (4 registers for two 64-bit operands,

¹ When m -bit multiplication is required, one m -bit multiplication is evenly divided into 4 $m/2$ -bit multiplication operations. Among them 3 $m/2$ -bit multiplication is performed in COS method on NEON engine. On ARM processor, $m/2$ -bit multiplication is performed in hybrid-scanning method (width: 64-bit).

4 registers for 128-bit intermediate results and 1 register for carry value) are used to retain the variables. For inner loop of hybrid scanning method, 4 partial products (32-bit wise) are performed as follows. Given 2-word operands $\{A(A[1], A[0]), B(B[1], B[0])\}$, the four partial products ($A[0] \cdot B[0]$, $A[0] \cdot B[1]$, $A[1] \cdot B[0]$ and $A[1] \cdot B[1]$) are performed from the least significant word to the most significant word. In order to optimize the hybrid scanning in ARM instruction sets, we used `umlal` (multiplication and accumulation) instruction to update the intermediate results² The `umlal` instruction optimizes one addition instruction in each accumulation round since the lower 32-bit register (`r0`) is updated directly. The higher 32-bit is retained in temporal register (`r3`). Afterward the temporal register is added to the intermediate results (`r1-r2`). The detailed accumulation part is as follows.

```
umlal r0, r3, r4, r5 → adds r1, r1, r3 → adcs r2, r2, #0
```

The optimized inner loop (64-bit multiplication) only requires 4 `umull/umlal`, 12 `adds/adcs` and 1 `mov` operations. This inner loop is used 4 times to perform 256-bit multiplication.

The detailed mixed computation order is described in Fig. 1. In order to improve the readability, we used following notations in Fig. 1. Let A and B be operands with a length of m -bit. Each operand is written as follows: $A = (A[n-1], \dots, A[2], A[1], A[0])$, $B = (B[n-1], \dots, B[2], B[1], B[0])$, whereby $n = \lceil m/w \rceil$, and w is the word size. The result of multiplication $C = A \cdot B$ is twice length of operand, and represented by $C = (C[2n-1], \dots, C[2], C[1], C[0])$. For better understanding, we describe the method using both multiplication structure and rhombus form. The multiplication structure describes order of partial products from top to bottom and each point in rhombus form represents a partial product. The rightmost corner of the rhombus represents the lowest indices ($i, j = 0$), whereas the leftmost represents corner the highest indices ($i, j = n-1$). A black arrow over the point indicates the processing of the partial products. The lowermost side represents result indices $C[k]$, which ranges from the rightmost corner ($k = 0$) to the leftmost corner ($k = 2n-1$). In ARM instruction set (SISD architecture), we perform one partial product in a row. On the other hand, in NEON instruction set (SIMD architecture), we perform two partial products in a row. The differences of partial products are described in multiplication structure.

In order to perform ARM/NEON co-design of multiplication the multiplication is evenly divided into four sub-sections (①②③④). The three parts (①②③) are performed in NEON instruction using COS method and the other part (④) is performed in ARM instruction using Hybrid Scanning method. Since both ARM and NEON computation units are separated modules, ARM and NEON instructions are performed in parallel way without interference. The comparisons of execution timing between NEON and ARM/NEON approaches are drawn in Fig. 3(a). In previous approach, 4 sub-multiplication operations are sequentially performed in NEON engine. The NEON-only implementation achieved high

² `umlal a, b, c, d : {b, a} ← {b, a} + c × d.`

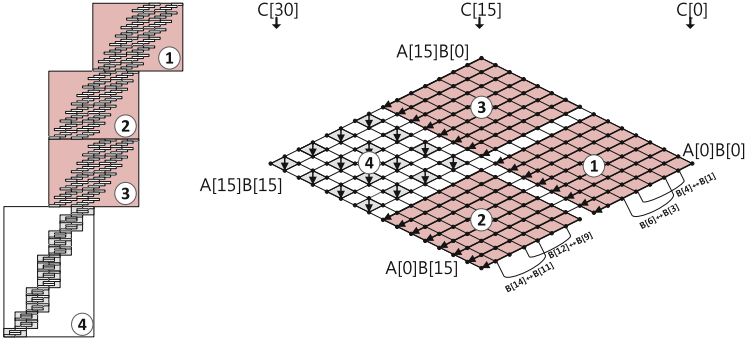


Fig. 1. ARM/NEON co-design for multiplication on ARM–NEON ①②③:NEON, ④:ARM

Algorithm 1. ARM/NEON Co-design of integer multiplication

Require: n -word operands $\{A(A_H A_L)$	3: $C_3 \leftarrow A_H \cdot B_L$	{NEON}
and $B(B_H B_L)\}$	4: $C \leftarrow C_1 + (C_2 + C_3) \cdot 2^{\frac{n}{2}}$	{NEON}
Ensure: $2n$ -word result (C)	5: $C_4 \leftarrow A_H \cdot B_H$	{ARM}
1: $C_1 \leftarrow A_L \cdot B_L$	{NEON}	
2: $C_2 \leftarrow A_L \cdot B_H$	{NEON}	
	6: $C \leftarrow C + C_4 \cdot 2^n$	{ARM}

performance through data parallelism in SIMD instruction sets. However, the approach does not use ARM module, which has enough functionality to perform the multiplication. In order to fully utilize both ARM and NEON modules, we use the mixed ARM/NEON instruction sets. The proposed technique performs 3 sub-multiplication operations on NEON engine (Algorithm 1 Step 1, 2, 3, 4) and 1 sub-multiplication operation on ARM module (Algorithm 1 Step 5), simultaneously. Afterward, both intermediate results from ARM and NEON are added together to output the final result. The accumulation step is performed in ARM instruction (Algorithm 1 Step 6). This co-design approach optimizes the execution time for one sub-multiplication (ARM) since the latency for ARM instruction is hidden into NEON instruction. The comparison in source code level is described in Table 1. In sequential approach, NEON or ARM instruction set is solely executed in the time line. On the other hand, parallel approach executes ARM and NEON instruction sets alternatively. This program style ensures the computations of both ARM and NEON in the same time line. With this approach, a number of clock cycles for ARM are optimized away.

We also investigated the Karatsuba technique for proposed co-design approach. However, we found that the Karatsuba technique is inefficient for co-design approach since the Karatsuba technique replaces 1 $m/2$ -bit multiplication into several $m/2$ -bit addition and subtraction operations. The addition and subtraction operations generate carry/borrow propagation, which cause a number of pipeline stalls on NEON engine. Alternatively, we can perform the operations

Table 1. Sequential and parallel implementations for integer multiplication in source codes

Sequential	Parallel
<NEON>	<ARM/NEON>
vmull.u32 q13, d0, d4[0]	vmull.u32 q13, d0, d4[0]
vmull.u32 q12, d2, d4[0]	ldr r3, [r1, #4*8]
vmull.u32 q11, d1, d4[0]	vmull.u32 q12, d2, d4[0]
vmull.u32 q10, d3, d4[0]	ldr r4, [r1, #4*9]
veor q5, q5, q5	vmull.u32 q11, d1, d4[0]
veor q6, q6, q6	ldr r5, [r2, #4*8]
veor q7, q7, q7	vmull.u32 q10, d3, d4[0]
veor q8, q8, q8	ldr r6, [r2, #4*9]
vtrn.32 q13, q5	veor q5, q5, q5
vtrn.32 q12, q6	umull r7, r8, r3, r5
vtrn.32 q11, q7	veor q6, q6, q6
vtrn.32 q10, q8	umull r9, r10, r4, r6
vadd.i64 q12, q12, q5	veor q7, q7, q7
vadd.i64 q11, q11, q6	mov r14, #0
vadd.i64 q10, q10, q7	veor q8, q8, q8
vqadd.u64 d16, d16, d27	umlal r8, r14, r3, r6
vext.8 d28, d28, d26, #4	vtrn.32 q13, q5
...	adds r9, r9, r14
<ARM>	vtrn.32 q12, q6
ldr r3, [r1, #4*8]	adcs r10, r10, #0
ldr r4, [r1, #4*9]	vtrn.32 q11, q7
ldr r5, [r2, #4*8]	mov r14, #0
ldr r6, [r2, #4*9]	vtrn.32 q10, q8
umull r7, r8, r3, r5	umlal r8, r14, r4, r5
umull r9, r10, r4, r6	vadd.i64 q12, q12, q5
mov r14, #0	adds r9, r9, r14
umlal r8, r14, r3, r6	vadd.i64 q11, q11, q6
adds r9, r9, r14	adcs r10, r10, #0
adcs r10, r10, #0	vadd.i64 q10, q10, q7
mov r14, #0	str r7, [r0, #4*16]
umlal r8, r14, r4, r5	vqadd.u64 d16, d16, d27
adds r9, r9, r14	str r8, [r0, #4*17]
adcs r10, r10, #0	vext.8 d28, d28, d26, #4
str r7, [r0, #4*16]	
str r8, [r0, #4*17]	

on ARM module but this imposes more tasks on ARM module than proposed method and the approach leads to the unbalanced overheads between ARM and NEON. The detailed complexities are as follows. In m -bit Karatsuba multiplication, ARM module performs 1 $m/2$ -bit multiplication, 3 $m/2$ -bit addition, 3 m -bit addition and 2 $m/2$ -bit subtraction operations. On the other hand, NEON performs 2 $m/2$ -bit multiplication, 2 $m/2$ -bit exclusive-or and 1 m -bit exclusive-or operations. This unbalanced overheads show that Karatsuba technique for co-design of ARM/NEON is inefficient to perform in parallel way. Instead, we perform Karatsuba technique as outer routine for the proposed co-design implementations. We adopted the Karatsuba technique on ARM–NEON from previous paper [9]. For 1024-bit and 2048-bit integer multiplication operations, we use 1-level/2-level Karatsuba methods, respectively.

2.2 Squaring

Similarly, previous integer squaring implementations on ARM–NEON processor mainly use NEON instructions to improve the performance of squaring operation. As we witnessed the performance enhancements through co-design of ARM/NEON in previous sub-section, the integer squaring operation is also accelerated.

First we divide the m -bit integer squaring into 2 $m/2$ -bit multiplication and $m/2$ -bit squaring. Among them 2 $m/2$ -bit multiplication operations are performed in DOS method on NEON engine. On ARM module, $m/2$ -bit squaring is performed in sliding block doubling (SBD) method [7]. For 256-bit operand case, total 12 general purpose registers are used. Specifically, 8 registers for 256-bit operands, 3 registers for 96-bit intermediate results and 1 register for temporal storage are used. The execution timing for ARM instruction is optimized away in co-design, since both ARM and NEON instructions are performed in parallel way.

The detailed descriptions are available in Fig. 2. The squaring is divided into three sub-parts. The parts (①②) are performed in DOS method on NEON engine and the other part (③) is performed in SBD method on ARM module. Since both ARM and NEON are separated modules, both instructions are performed, simultaneously. The comparisons between NEON and ARM/NEON implementations are drawn in Fig. 3(b). In traditional DOS method (NEON only), 3 sub-multiplication operations are sequentially performed and execution timing is about 3 sub-multiplication operations. On the other hand, the proposed co-design approach performs 2 sub-multiplication operations on NEON engine (Algorithm 2 Step 1, 2, 3) and 1 sub-squaring operation on ARM module (Algorithm 2 Step 4), simultaneously. Afterward, both intermediate results are added together to output the combined final result (Algorithm 2 Step 5). This approach ensures the optimization of one sub-squaring operation on ARM module. The operation complexities are 1 $m/2$ -bit squaring and 1 m -bit addition for ARM and 2 $m/2$ -bit multiplication for NEON, respectively. Finally, we also perform the outer Karatsuba technique to enhance the performance of

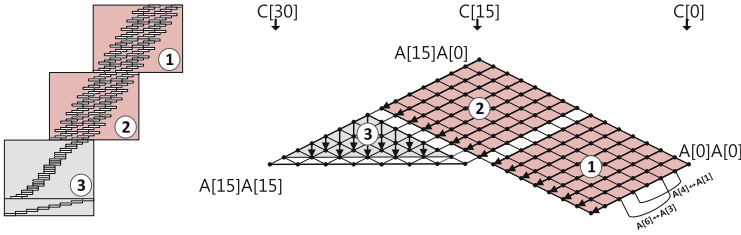


Fig. 2. ARM/NEON co-design for squaring on ARM-NEON, ①②: NEON, ③: ARM

Algorithm 2. ARM/NEON Co-design of integer squaring

Require: n -word operand $A(A_H A_L)$	3: $C \leftarrow C_1 + C_2 \cdot 2^{\frac{n}{2}}$	{NEON}
Ensure: $2n$ -word result (C)	4: $C_3 \leftarrow A_H \cdot A_H$	{ARM}
1: $C_1 \leftarrow A_L \cdot A_L$	{NEON}	5: $C \leftarrow C + C_3 \cdot 2^n$
2: $C_2 \leftarrow 2 \cdot A_L \cdot A_H$	{NEON}	{ARM}

long integer cases. For 1024-bit and 2048-bit integer squaring operations, we use 1-level/2-level Karatsuba methods.

3 Results

3.1 Target Platform

In this paper, we selected high-end IoT architecture, namely ARMv7. Particularly, we execute the implementations on the ARM Cortex-A9 processor which supports full functionalities of ARMv7 architecture and NEON engine. The ARMv7 and NEON supports 32-bit and 64/128-bit wise registers, respectively. The ARMv7 instruction set performs 32-bit SISD operations. On the other hand, the NEON engine provides vectorized instructions in 16 8-bit, 8 16-bit, 4 32-bit and 2 64-bit, performing multiple data in single instruction. The ARMv7 processor is widely used in high-end IoT processors including mini computers and

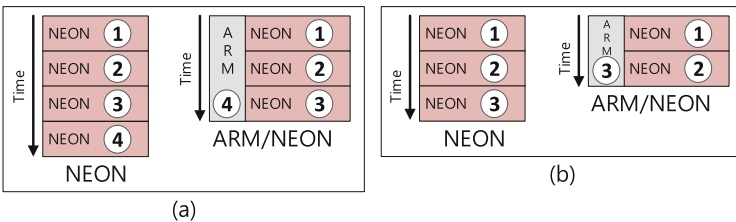


Fig. 3. Comparison between standalone and co-design in execution time, (a) multiplication, (b) squaring

Table 2. Results of integer multiplication and squaring operations in clock cycles on ARM Cortex-A9 platform

Bit	Proposed	[9]	[8]	GMP
Multiplication				
512	851	1048	1050	2176
1024	3191	3791	4298	6256
2048	10657	13736	17080	19618
Squaring				
512	595	850	-	1343
1024	2244	3315	-	4063
2048	7495	9180	-	14399

mobile devices. The proposed implementations are working on all ARMv7 architectures including Cortex-A7, A8, A15 without modifications of source codes.

3.2 Evaluation

In order to provide fair comparison results, we compare the performance with previous works in identical target processors (Cortex-A9) [2, 8, 9]. For our evaluation, we only employ the single core on the processor³. The optimization level is set to maximum (`-O3`), which fully optimizes the inefficient program routines in compiler level. In Table 2, the execution timing for integer multiplication and squaring are drawn. We measured the performance ranging from short integer (512-bit) to long integer (2048-bit) and compared the results with state-of-art implementations [8, 9].

For short integer case (512-bit), the proposed multiplication method achieves an execution time of 851 clock cycles and squaring requires 595 clock cycles. On the other hand, Seo et al. achieved 1,048 and 850 clock cycles for integer multiplication and squaring, respectively. Thus, the proposed methods outperform Seo et al. by approximately 18.8% and 30.0% for integer multiplication and squaring operations, respectively. For medium integer case (1024-bit), 1-level Karatsuba technique is used to enhance the performance. The proposed multiplication and squaring implementations achieved 3,191 and 2,244 clock cycles, respectively. Seo et al. achieved 3,791 and 3,315 clock cycles for multiplication and squaring. The performance enhancements are 15.8% and 32.3% for multiplication and squaring operations. For long 2048-bit operand, we used 2-level Karatsuba technique on both operations. The proposed multiplication requires 10,657 clock cycles, while Seo et al.’s implementation requires 13,736 clock cycles. In terms of 2048-bit squaring, our method requires 7,495 clock cycles, while Seo et al.’s implementation requires 9,180 clock cycles. The result shows that proposed methods

³ If we define multi-core processing through OpenMP library and execute multiple threads, the performance is enhanced by the number of threads.

outperform Seo et al. by approximately 22.4% and 18.3% for multiplication and squaring, respectively. We observe that Karatsuba multiplication is more efficient than that of Karatsuba squaring in long integer since the complexity reduction in squaring operation is smaller than that of multiplication parts. Furthermore, the proposed implementations satisfy the operand scalability. We can conduct various length of integer multiplication or squaring in a single code by altering the level of Karatsuba. This is practical implementations for random numbers.

Comparison to GMP. The most well known multiple precision arithmetic library is GNU Multiple Precision Arithmetic Library (GMP). The GMP also uses asymptotically fast Karatsuba algorithm for long integer and many assembly level optimizations. In order to compare the performance with public library (GMP), we executed the benchmark program on the target platform. The detailed execution timing is drawn in Table 2. The performance gap between proposed method and GMP is 60.9% and 45.7% for 512-bit multiplication and squaring, respectively. This gap is roughly maintained in long operands (2048-bit), since both implementations use Karatsuba techniques. Finally, we conclude that our implementations significantly outperform the public library.

Comparison in Operation Complexity. In this section, we compare the complexities of integer multiplication and squaring operations. The detailed descriptions are available in Table 3. In previous COS method, main multiplication parts are performed in NEON engine [9]. Afterward, the intermediate results are accumulated by using ARM instruction in the last step, which costs $\frac{m}{2}$ -bit addition. For this reason, NEON and ARM instructions are executed in sequential way. On the other hand, the proposed approach assigns $\frac{m}{2}$ -bit multiplication and 2 $\frac{m}{2}$ -bit addition operations to ARM module. The $\frac{m}{2}$ -bit multiplication is performed while NEON engine performs 3 $\frac{m}{2}$ -bit multiplication operations, which ensures high-optimized parallel computations. After parallel computations, the separated intermediate results are combined together through 2 $\frac{m}{2}$ -bit addition in ARM instruction. This shows that the proposed method replaces $\frac{m}{2}$ -bit multiplication into $\frac{m}{2}$ -bit addition.

Table 3. Complexities of integer multiplication and squaring on ARM–NEON, where A , M and S represent $\frac{m}{2}$ -bit addition, multiplication and squaring, respectively.

Method	ARM	NEON	Total	Optimization
Multiplication				
COS [9]	A	$4M$	$A + 4M$	–
Proposed	$2A + M$	$3M$	$2A + 3M$	M
Squaring				
DOS [9]	A	$3M$	$A + 3M$	–
Proposed	$2A + S$	$2M$	$2A + 2M$	S

In terms of squaring operation, $\frac{m}{2}$ -bit addition and 3 $\frac{m}{2}$ -bit multiplication operations are required in DOS method [9]. On the other hand, the proposed method performs 2 $\frac{m}{2}$ -bit addition and $\frac{m}{2}$ -bit squaring operations in ARM module and 2 $\frac{m}{2}$ -bit multiplication in NEON engine, respectively. Among them, latency for $\frac{m}{2}$ -bit squaring operation of ARM instruction is hidden into that of NEON. Finally, $\frac{m}{2}$ -bit squaring operation is replaced into $\frac{m}{2}$ -bit addition in proposed method.

4 Conclusion

In this paper, we presented optimization techniques to improve the performance of multi-precision multiplication and squaring on ARM–NEON architecture. Unlike previous approaches, we divide the computations into sub-sections and properly perform the operations in both ARM and NEON instruction sets. Since the computations are performed in parallel way, the latency of ARM instruction is hidden into that of NEON instruction. On an ARM Cortex-A9 processor, our proposed implementations perform 2048-bit multiplication and squaring operations only 10,657 and 7,495 clock cycles, which are 22.4% and 18.3% faster than state-of-art implementations [9]. Based on these optimizations, the most obvious future work is to apply the proposed methods to public key cryptography such as (pre-quantum) RSA and (post-quantum) SIDH [3, 9] since these cryptography applications require long integer multiplication and squaring, which are performance bottleneck. We believe that the proposed method will push the speed boundaries even further from the state-of-art implementations.

References

1. Bernstein, D.J., Schwabe, P.: NEON crypto. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 320–339. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33027-8_19
2. Bos, J.W., Montgomery, P.L., Shumow, D., Zaverucha, G.M.: Montgomery multiplication using vector instructions. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 471–489. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43414-7_24
3. Koziel, B., Jalali, A., Azarderakhsh, R., Jao, D., Mozaffari-Kermani, M.: NEON-SIDH: efficient implementation of supersingular isogeny Diffie-Hellman Key exchange protocol on ARM. In: Foresti, S., Persiano, G. (eds.) CANS 2016. LNCS, vol. 10052, pp. 88–103. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48965-0_6
4. Martins, P., Sousa, L.: On the evaluation of multi-core systems with SIMD engines for public-key cryptography. In: 2014 International Symposium on Computer Architecture and High Performance Computing Workshop (SBAC-PADW), pp. 48–53. IEEE (2014)
5. Martins, P., Sousa, L.: Stretching the limits of programmable embedded devices for public-key cryptography. In: Proceedings of the Second Workshop on Cryptography and Security in Computing Systems, pp. 19–24. ACM (2015)

6. Pabbuleti, K.C., Mane, D.H., Desai, A., Albert, C., Schaumont, P.: SIMD acceleration of modular arithmetic on contemporary embedded platforms. In: 2013 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1–6. IEEE (2013)
7. Seo, H., Liu, Z., Choi, J., Kim, H.: Multi-precision squaring for public-key cryptography on embedded microprocessors. In: Paul, G., Vaudenay, S. (eds.) INDOCRYPT 2013. LNCS, vol. 8250, pp. 227–243. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03515-4_15
8. Seo, H., Liu, Z., Großschädl, J., Choi, J., Kim, H.: Montgomery modular multiplication on ARM-NEON revisited. In: Lee, J., Kim, J. (eds.) ICISC 2014. LNCS, vol. 8949, pp. 328–342. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15943-0_20
9. Seo, H., Liu, Z., Großschädl, J., Kim, H.: Efficient arithmetic on ARM-NEON and its application for high-speed RSA implementation. IACR Cryptology ePrint Archive 2015:465 (2015)
10. Seo, H., Liu, Z., Nogami, Y., Park, T., Choi, J., Zhou, L., Kim, H.: Faster ECC over $\mathbb{F}_{2^{521}-1}$ (feat. NEON). In: Kwon, S., Yun, A. (eds.) ICISC 2015. LNCS, vol. 9558, pp. 169–181. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30840-1_11
11. Seo, H., et al.: Parallel implementations of LEA, revisited. In: Choi, D., Guilley, S. (eds.) WISA 2016. LNCS, vol. 10144, pp. 318–330. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56549-1_27

Web Security and Emerging Technologies



WheelLogger: Driver Tracing Using Smart Watch

Joon Young Park, Jong Pil Yun, and Dong Hoon Lee^(✉)

Graduate School of Information Security, Korea University, Seoul, Korea
{null,2015572008,donghlee}@korea.ac.kr

Abstract. Location-related data is one of the most sensitive data for user privacy. Theft of location-related information on mobile device poses serious threats to users. Even though the extant confirmation of permissions feature on modern smart devices can prevent direct leakage of information from location-related sensors, recent research has shown that leakage of location-related information is possible through indirect, side-channel attacks. In this paper, we show that the travel path of a vehicle can be inferred without acknowledging the user using a zero-permission smart watch application. The sensor we used in our experiment is the accelerometer sensor on Apple Watch. We find that a targeted user can be traced with 83% accuracy. We suggest that our approach may be used to successfully attack other smart phone devices because it was successful on Apple Watch, which is considered as the most constrained device in the market. This result shows that the zero-permission application on a smart watch, if manipulated adequately, can transform into a high-threat malware.

1 Introduction

Evolution of features on mobile phones has significantly changed how we share information and access online services. The advent of new mobile phone applications has brought huge benefits, but also extensive privacy risks. Leakage of location information is among the major privacy risks rendered by use of mobile phone applications, as it can enable high-threat attacks such as tracking users, identity discovery, and identification of home and work locations. Furthermore, it can allow an attacker to discover target persons behavior, habits, preferences and personal connections, which can be deployed in targeted social engineering.

The subject of location privacy has thus been widely studied since early years of mobile devices. Cellular communication systems, as early as GSM, have been adapted to protect user identity. Temporary identifiers (e.g., TMSI), for instance, have been used to thwart attempts to track users. In recent years, ubiquity of mobile and sensing devices, open mobile platforms (running untrusted code) and ubiquitous connectivity have expanded the attack surface of location privacy significantly. Users appear to have followed up, increasingly aware of and concerned about the implications of location information disclosure, as reported in recent

surveys [5], and acknowledged in the US Congress Location Privacy Protection Act of 2014 [12].

Rich sensors on wearable devices offer useful data that enable applications for mobile health, activity tracking, games, etc. Unfortunately, data collected by such sensors can often be sources of privacy breach, when they leak information about the private aspects of users lives. In our attempt to define the appropriate level of data disclosure, we arrive at the question of: what can be inferred from a given sensor data? Every so often, we find that substantial inferences can be made from apparently harmless data, setting us up for information exposure. While much research has been done in this area, and measures suggested from them implemented to enhance security, information leakage via sensor data continues to occur. With leakage of sensor data of wearable device, the words typed into the phone [14], passwords typed into other smart devices, and certain speech-related information [6] can be exposed [7]. In this paper, we raise the question of if an accelerometer data from smart watches can be mined to infer the roads that the user has driven. In other words, can motion data from the users wrist moves while steering the wheel and accelerating the vehicle be used to find out the roads that the user traveled? If so, the problems are serious - a smart watch app can be disguised as a travel path tracer to leak data on users commuting route or frequent locations.

So far, the simplest way to breach data on a users location is to access the mobile device location services, which relies on GPS, Wi-Fi, or cellular signals. To mitigate breaches of location privacy, mobile phone operating systems, such as Android, provide mechanisms through which users can manage applications permissions to access sensitive resources and information.

While a careful user may refuse such request for access his or her location information, the greater challenge of protecting users location privacy against side channel attacks, which do not involve permission requests, remains. A variety of sensors, including as gyroscope, accelerometer, and magnetometer, are embedded in mobile phones. These sensors expand the attack surface, rendering the mobile phone an attractive target for those seeking to exploit privacy information [4, 9, 15], especially when users attempt to minimize their exposure by disabling, removing or limiting usage of tracking applications.

There has previously been research that addressed this topic. Narain et al. infer the user routes and locations in [10], using gyroscope, accelerometer and magnetometer sensors in smart phone. Narain et al. used their algorithm order to infer user travel paths and locations of mobile phone users in 11 cities, including Boston and Waltham. They inferred 10 routes in each experiment, and reported that 30% to 60% of the total distance travelled, was accurately inferred. The level of accuracy achieved in this study, however, is unlikely to offer sufficient data for those devising further attack on the target.

We investigate the threat and potential of tracing users mobility without explicitly requesting permissions to access the device sensors or location services. Currently, any Android and iOS application can access the accelerometer without requiring the user permission. Even security conscious user tends to

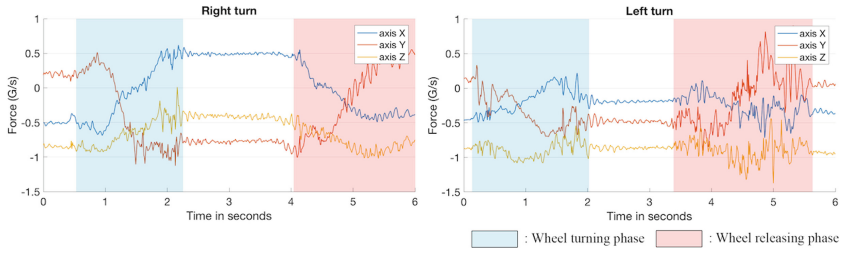


Fig. 1. Steering data in stationary condition.

underestimate the risks associated with installing an application that does not request access to sensitive information such as that on location. We focus on the case of a user riding a vehicle on public roads without peculiar conditions.

The contributions of this paper are as follows:

- An efficient approach for handling the accelerometer data to identify the distance and curves traveled, and the velocity at which the user/driver traveled using mathematical and machine learning method.
- A proposal of the simple methodology of map matching framework, drawing the draft version of route and adjusting by referencing the form over the map.
- Evaluation of the feasibility and effectiveness of travel path inference attacks by implementing our approach, which used the accelerometer as side-channel. As Apple is known to provide highest protection of user privacy, our research has proven that the side-channel attack using rich sensors should be considered as a threat equivalent to that of mobile malware intimidation.

In Sect. 2, we introduces the motivation of this research and Sect. 3 will discuss about the threat model and assumptions of our attack. In Sect. 4, we refined the coarse sensor data to the meaningful data to draw the victim’s route traveled. Section 5 matches the preprocessed route to the real map data using image processing and Sect. 6 evaluates the attack accuracy and energy consumption upon sampling rate of the sensor data. Finally, Sect. 7 concludes the paper.

2 Driving Motions: Inspiration

To understand the problem of this research, we take a first look into the data from smart watches. While we are driving, we use the pedals and the steering wheel to accelerate, decelerate and change the direction of the car. These movements are completely controlled by the drivers’ foot or hands. If the driver have a smart devices which has sensors (i.e., accelerometer, gyroscope), its data will be logged as the car moves. Particularly, in case of the driver wear a smart watch on his wrist, the smart watch will be moved over time and the sensors will follow the movement of the steering.

Noting this, two of the authors wore a smart watch and recorded the accelerometer data of the steering and acceleration. In this paper, all of the



Fig. 2. Hand movement while curve to the right.

experiments are done with the Apple Watch 38 mm (watchOS ver. 3.2.1) and the application we made is installed to the watch to collect the accelerometer data, using the **CMAccelerometerData** class [1]. In this experiment, sampling rate of accelerometer was at 100 Hz (the highest sampling rate of Apple Watch), and subjects placed their hands as *Side-grip* position, one of the standardized grip position of the drivers. Details will be described in Sect. 5.

2.1 Steering Data

We first attempted to collect the accelerometer data of the turning-wheel while the car is in stationary condition. The starting hand position on the wheel is in the direction of 9 to 10 o'clock. The stable part (0 to 100 ms) of the Fig. 1 represents the idle status of the wheel. As the wheel starts to turn, each axis data varies on the tilt angle of the watch (Fig. 2). We extracted the features of 300 labeled steering data by calculating the mean, root-mean-square, median absolute deviation values of each axis and trained the dataset with several learning algorithms. The labeled feature vectors were used to train the classifiers in the learning phase, whereas additional 100 unlabeled features are mapped to the closest matching class by the trained classifiers in the attack phase. In this experiment, we used SVM (Support Vector Machines), Bagged Trees and K-Nearest Neighbor classifier. As a result, each algorithm results at least 77% and up to 100% which is enough to infer the direction from the unlabeled data.

2.2 Acceleration Data

Conversely, acceleration of the car cannot be captured without difficulty. With the naked eye, no any data can be seen related to the acceleration of the vehicle. The data from the watch have enormous noise (i.e., the vibrations from the ground, a fine steering control of the driver, noise from the sensor itself, etc.), besides, we could not simply analogize the velocity or traveled distance of the vehicle from the accelerometer data. Afterward, we designed a system for reducing the noise and inferring the expected result using filter and applied mathematics in Sect. 5.

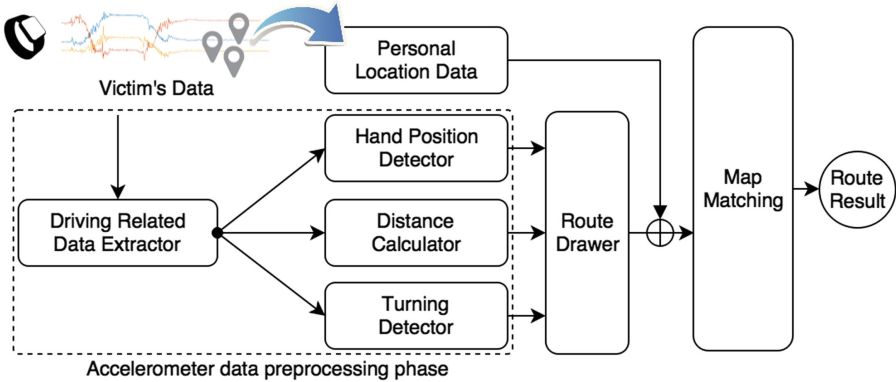


Fig. 3. System overview of the WheelLogger.

3 System Overview

We designed an overall system WheelLogger, represented as Fig. 3. We designed and implemented an Apple Watch application to collect accelerometer data from the watch. The application does not require any permissions to collect and send the accelerometer data. Once the application is installed, it continuously logs and sends the accelerometer data to the attacker’s colluding server. The server first marks the starting and ending point of the driving data with *Driving Related Data Extractor* module. After the drive data is extracted, *Hand Position Detector* recognizes the grip characteristic of the driver, which can affect the result of each processing module significantly. Afterward, *Distance Calculator* and *Turning Detector* find the results of distance traveled and turning branches of the victim. With these preprocessed results, we can draw the draft route for the victim and matches the route with the real map data based on the victim’s personal location data. To this end, the system results the inferred route of the victim.

3.1 Threat Model

The proposed attack assumes that an attacker has infected a specific target user to compromise with a carefully-designed malicious application. This application running in the background on smart watch can collect accelerometer output which are available on almost all major smart watches, and uploads the reading to remote servers through any available wireless networks. This kind of malware is not hard to create as both the accelerometer sensors can be accessed without requiring the permission of users. Although installing the malwares on the smart phone is inconsiderable, installing the application to the smart watch can be another problem. However, most of all applications for the smart watch are installed in silence when paired smart phone application is installed.



Fig. 4. Steering wheel grip positions

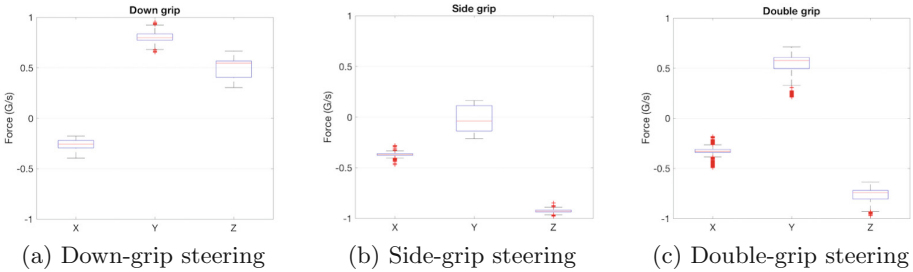


Fig. 5. Plot data according to the position of the driver’s wrist

3.2 Assumptions

Before moving forward, we intend to list a number of assumptions. The assumptions could be relaxed with further works.

- The driver should wearing a smart watch on the main hand, which steers the wheel. The main hand should be remained on the wheel while in driving. The other hand can be used to assist main hand, however, can **NOT** be substituted or replaced with the main hand.
- An adversary should select a victim who is an acquaintance. The adversary knows the victim’s frequent locations, such as home or the company. This information will be used for matching with the expected route to the actual map.

4 Refining Sensor Data

In order to extract the route data from raw data, we designed four phases to refine sensor data.

4.1 Driving Related Data Extractor

From huge amount of the raw sensor data from the victim’s smartphone, an attacker should extract driving-related data first. We found the features that

Algorithm 1. FindDrivingData (Starting point)

Input: A sequence of accelerometer data, $AccData$; A mean values of the each hand position data set, $HPos$

Output: A starting point of driving related accelerometer data sequence, sp ;

- 1: **Initialization:** $sp, trigger, from \leftarrow 0, w \leftarrow 20, to \leftarrow w$;
- 2: **while** $sp = 0$ **do**
- 3: **while** $to < length(AccData)$ **do**
- 4: $to \leftarrow from + w$;
- 5: **if** $sp = 0$ **then**
- 6: $pMean = mean(AccData[from:to])$;
- 7: $nMean = mean(AccData[from+w:to+w])$;
- 8: $absPN \leftarrow |pMean - nMean|$;
- 9: **if** $absPN < thr$ **and** $pMean \cap HPos \neq \phi$ **then**
- 10: **if** $trigger = 0$ **then**
- 11: $tempSp \leftarrow from - w$;
- 12: **end if**
- 13: $trig \leftarrow trig + 1$;
- 14: **if** $trig \geq 3$ **then**
- 15: $sp \leftarrow tempSp$;
- 16: **end if**
- 17: **end if**
- 18: **end if**
- 19: $from \leftarrow to$;
- 20: **end while**
- 21: $thr \leftarrow thr - 0.02$;
- 22: **end while**
- 23: **return** sp

can be inferred from the data generated while the victim is in the car. After the victim gets on the car to drive, he grabs the wheel with his main hand and starts the engine with the other hand. At that time, sensor data from the watch on driver's hand is in the stable phase, similar to the head part of Fig. 1. Moreover, the type of the driver's grip style is formalized, typical types are described in Fig. 4 and the accelerometer data from each position is described in Fig. 5. Of course, these aspects of sensor data can be seen not only in the car, but also any other status (i.e., hands on the table, typing a keyboard, etc.). In contrast, when the driver steers the wheel, the feature of the sensor data from the watch is unique enough to recognize whether the victim is in drive or not (i.e., turning phase). Consequently, if turning phase is immediately followed by the stable phase, we can certain that the victim is in drive.

We designed Algorithm 1 to find the starting point of the driving related data. The algorithm first repeats to calculate the absolute value between two mean values within the time window 0.2s, and check whether the data varies over the threshold or not to find the stable state. Additionally, the algorithm compares the mean value of the stable data with pre-defined *Hand Position* data in Fig. 5. If the data determined as in range of one of the *Hand Position* data,

the algorithm finally recognizes the data as the driving related data and marks the stable part as a starting point. In the same way, the algorithm can find the ending point of the driving data when *AccData* is reverse-inputted.

4.2 Hand Position Detector

The hand position on the wheel is the most critical factor for finding the route from accelerometer data. Statistically, the hand positions for most of the drivers can be divided into three groups; *Down grip*, *Side grip* and *Double-handed grip*, as described in Fig. 4. However, a driver has his/her own driving habits (i.e., other than the position of the hands on the wheel such as switching the signal light, opening the car window, and etc.). These habits could add the noise to the sensor data and can prevent from acquiring the correct hand portion. Fortunately, as shown in Fig. 5, each hand position has a distinct range of vibrations and hence can be well recognized even with data with noise.

We note that the sensor data of acceleration, deceleration and gyration have significant difference in correlation to the position of the driver's main hand. This implies that different accelerate and curve detection algorithms should be used according to the result of the hand position detector. In the rest of the paper, for simplicity, we assume that the hand position is *Side grip*. Algorithms for other hand positions can be designed in similar manner.

4.3 Distance Calculator

In this section, we calculate velocity and traveled distance of a vehicle using accelerometer data from Apple Watch. In the beginning, we collected x, y, z -axis of accelerometer data recorded with the unit G/s. It means that the object is experiencing ground force and its acceleration relatives to free-fall, physically. First, we convert data in G/s into data in meter per second squared (m/s^2), with $1 \text{ G/s} \approx 9.8 \text{ m/s}^2$.

In order to calculate the velocity and the distance traveled, we have to find the exact equation of time-acceleration graph. For this purpose, we use *Lagrange interpolation* [2]. The effect of adopting other methods such as regression could be further analyzed. Lagrange polynomial is computed as follows:

Let $P(x)$ be polynomial of degree n . With $n+1$ points $(x_i, f(x_i))$ where $f(x_i)$ is acceleration data at time x_i for $0 < i \leq n+1$. Then $P(x)$ is computed as:

$$P(x) = \sum_{i=1}^n (f(x_i) \prod_{\substack{i \leq m \leq n \\ i \neq m}} \frac{x - x_m}{x_i - x_m}). \quad (1)$$

That is, $P(x)$ of Formula (1) is the polynomial representing $n+1$ points.

However, raw accelerometer data from the sensor has tremendous noise and effects the accuracy of the distance traveled from accelerometer data. Hence, a specific process is required to filter the noise from the raw data. The easiest method is *moving-average filter* that estimates from the average of data a number

of before data and after. To improve the result of the method, we used *Savitzky-Golay filter* [11] to diminish the noise. This filter is a much better procedure than simply averaging points because it performs a least squares fit of a small set of sequential data points to a polynomial and takes the computed central point of the fitted polynomial curve as the new smoothed data point. *Savitzky-golay filter* is a digital filter that can be applied to a set of digital data points for the purpose of smoothing the data, that is, to increase the signal-to-noise ratio without greatly distorting the signal. As a result, we could get clear data and polynomial equation.

Background Theory. The best way to calculate velocity and position of vehicle is by physical approaching to mathematical integration [3, 13]. The acceleration is the rate of change of the velocity of the vehicle. At the same time, the velocity is the rate of change of the distance traveled of the vehicle. In other words, the velocity is the derivative of the position and the acceleration is the derivative of the velocity. Hence:

$$\mathbf{a} = \frac{d\mathbf{v}}{dt}, \quad \mathbf{v} = \frac{d\mathbf{s}}{dt} \quad \therefore \mathbf{a} = \frac{d(ds)}{dt^2}. \quad (2)$$

The integration is the opposite of the derivative. If we know the acceleration of the object, we can obtain the position data during a specific time range from u to w by computing the double integration as:

$$\mathbf{v} = \int_u^w \mathbf{a}(t)dt, \quad \mathbf{s} = \int_u^w \mathbf{v}(t)dt. \quad (3)$$

$$\therefore \mathbf{s} = \int_u^w \left(\int_u^w \mathbf{a}(t)dt \right) dt. \quad (4)$$

Our Experiment. The aim of this experiment is to compute the velocity and the distance of the vehicle traveled. Since we have obtained three axis (x, y, z) of accelerometer data, we compute three polynomial equation of time-acceleration graph ($\mathbf{a}_x - t$, $\mathbf{a}_y - t$ and $\mathbf{a}_z - t$ graph). By applying the integration method (3) and (4), we obtained a polynomial equation of the time-velocity graph ($\mathbf{v}_x - t$ and $\mathbf{v}_y - t$ and $\mathbf{v}_z - t$ graph). In order to get distance, the integration have to be performed again. Applying the same method and procedure to this obtained velocity data, we can get a fairly exact polynomial equation of the time-distance traveled graph ($\mathbf{s}_x - t$, $\mathbf{s}_y - t$ and $\mathbf{s}_z - t$ graph).

$$\mathbf{a}_x(t) \xrightarrow{\int} \mathbf{v}_x(t) \xrightarrow{\int} \mathbf{s}_x(t). \quad (5)$$

$$\mathbf{a}_y(t) \xrightarrow{\int} \mathbf{v}_y(t) \xrightarrow{\int} \mathbf{s}_y(t). \quad (6)$$

$$\mathbf{a}_z(t) \xrightarrow{\int} \mathbf{v}_z(t) \xrightarrow{\int} \mathbf{s}_z(t). \quad (7)$$

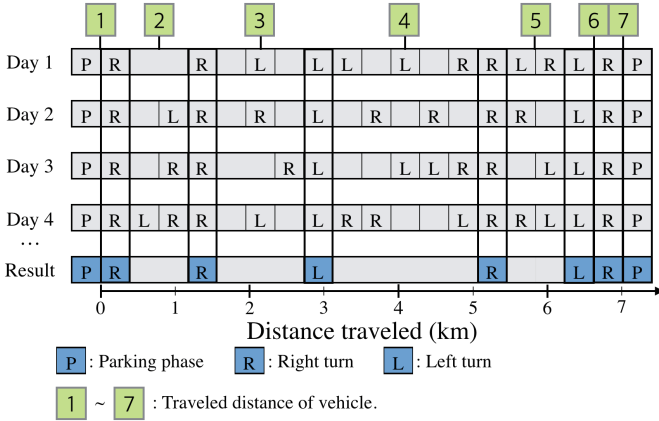


Fig. 6. Correcting algorithm of the curve data

Experiment Conclusion. Using the method (5), (6) and (7), we can compute the approximation of the traveled distance d of the vehicle. According to the polynomial above, the distance traveled of the vehicle at time $t = k$ is as follow:

$$d = \sqrt{s_x(k)^2 + s_y(k)^2 + s_z(k)^2}. \tag{8}$$

Through about 300 times Experiment, we get a margin of error of around ten percent, using formula (8). For example, we have results 163 and 210m from calculating experiment about the real distance traveled 180 and 230m respectively.

4.4 Turning Detector

While in driving, the driver not only goes forward, but also turns left, right or changes the lane. These movements happen when the driver steers the wheel. Certainly, the data of the accelerometer varies on the movements of the steering motions of the driver. The only two manners of steering the wheel are *right* or *left*, and they have their own distinct features which can be recognized clearly through the accelerometer data with appropriate classification algorithms in Sect. 2.

Nevertheless, real world situation is not ideal. Drivers steer the wheel not only to turn, but also to change the lane. While the vehicle is in the high speed state, the wheel needs to be changed only 10° to 15° to shift the lane. In contrast, the driver should steer more to shift the lane while the vehicle is running slow. For this reason, lane shifting data can be recognized as a turn data. Furthermore, we can not calculate the exact turning angle of the vehicle without the gyroscope sensor thus we can not avoid errors from changing the lane.

To overcome this problem, we collected same data from the same route for several days. Each data of one travel can be represented as a line. Turning spots and lane changing spots are marked above the line. The length of the line means the distance of the whole travel. Note that each line includes the lane

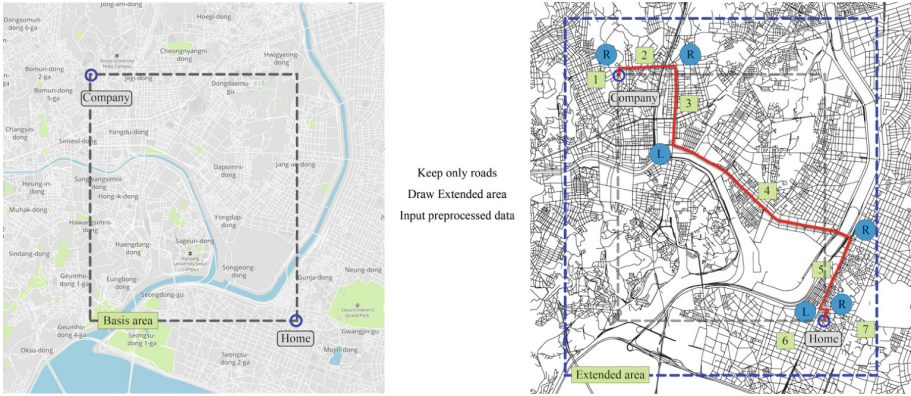


Fig. 7. Map matching process on the map. We can easily match the actual road to the map with adjusting the angles and road forms. (Color figure online)

shifting data. While the turning spot appears on the same length of the line, the lane shifting spot appears irregularly. Ignoring the irregular spots as the lane shifting, we can get fixed data which only consist the distance and the turning data without the lane changing data (Fig. 6).

5 Matching on Map

As a result of the preprocessing phase, expected route of the victim can be represented as a sequence of distances and curves. However, the data from the accelerometer only is not enough to infer the route exactly since we can not know the direction data of the vehicle. To overcome this problem, we need the victim's frequent locations such as home or office to set the start and end spot of the travel. Since these information is public, anyone close to the victim can collect them easily. We used OpenStreetMap [8], which is available map related data publicly.

5.1 Drawing Draft Route

From the results of the preprocessing phase, we can draw a draft version of expected route. The drawing consists of lines and corners, each component implies the distance traveled and turning direction. Thereafter, the drawing is rotated to the right direction by referencing the victim's personal location data (e.g., home, company, school), as described as the red line of the left map of Fig. 7.

5.2 Adjusting the Angles

Of course, the road does not only have the right-angled turns. However, the algorithm we proposed can not infer the exact angle of the turns of the vehicle.

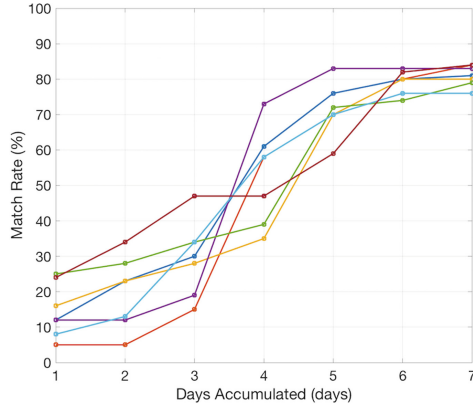


Fig. 8. Match rate against days accumulated.

In this point, we can overlap the drawing of the draft route to the road-only generated map and adjust the angle of the joints by referencing the form of the map. Although this method can be considered to be a force-matching, anyone can match this easily over the map with exact two spots of start point and end point, fixed length of lines and turning directions. As a result of this adjusting, we can infer the overall traveled route of the victim.

6 Evaluation

In this section, we evaluated the attack scenario with real driving experiments. WheelLogger has been implemented on the Apple Watch, which runs the latest watchOS platform. When activated, the WheelLogger client on the watch continuously logs accelerometer readings at 100 Hz, along with timestamps. The sensor data is stored locally during data collection and transferred to the backend (MATLAB) server for analysis.

WheelLogger is evaluated with 7 subjects, recruited by advertising about these experiments in the university campus. The subjects were offered an incentive of \$15 per hour, and each subject drove our laboratory’s experiment vehicle, SOUL from KIA. All subjects were familiar with driving, at least have 5-years actual driving experience. Each subject was asked to drive and collect the data twice a day, both from home to company and from company to home, using the same road for seven days. Each travel road was varied from 3 km to 7 km, including 5 to 15 turns and signal lamps. In total, we collected 98 accelerometer data across all users.

Figure 8 plots the match rate of each subject, against the accumulated days of the data. Through the correcting algorithm of Fig. 6, the results show that the match rate grows further with more route data is collected. The match rate is estimated of the length of the correct route per whole travel distance. In this figure, several data shown in low match rate at 3 to 4 days are determined

that has many alleyways. Therefore the calculated value can infer the actual distance only up to 91%, alleyways within several meters can not be determined precisely. Thus, this type of routes needs more data to infer the actual driven routes correctly. Consequently, when the result is collected more than 7 days, the match rate of overall attack improves at least 76% up to 84%.

7 Conclusion

This paper demonstrates that sensor data collected by smart watch can be used to infer the travel path of its user. By processing the accelerometer signals, we managed to infer the turns taken and the distance traveled, and identify the travel path of the Apple Watch user.

Our approach can be considered particularly threatening because it managed to identify the travel path of the smart watch user using data from Apple Watch's accelerometer sensor only. Note that other smart watches can access more diverse kinds of sensor data, allowing the attacker to make more substantial inferences about the user. The troubling fact is that these applications can be uploaded on AppStore and downloaded widely; the risk posed by such applications is very real.

In order to get a much better result, the further research will be needed. If we are able to obtain the data from additional sensor in smart phone such as gyroscope or magnetometer, we can get additional information which can be used for driver tracing.

Acknowledgement. This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (R7117-17-0161, Anomaly detection framework for autonomous vehicles).




References

1. Apple: API Reference: CMAccelerometerData. <https://developer.apple.com/reference/coremotion/cmaccelerometerdata>. Accessed 16 Aug 2016
2. Berrut, J.-P., Trefethen, L.N.: Barycentric lagrange interpolation. *SIAM Rev.* **46**(3), 501–517 (2004)
3. Halliday, D., Resnick, R., Walker, J.: *Principles of Physics*. Wiley, Hoboken (2011)
4. Hua, J., Shen, Z., Zhong, S.: We can track you if you take the metro: tracking metro riders using accelerometers on smartphones. [arXiv:1505.05958](https://arxiv.org/abs/1505.05958) (2015)
5. Fawaz, K., Shin, K.G.: Location privacy protection for smartphone users. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS 2014, pp. 239–250 (2014)
6. Maiti, A., Jadhwal, M., He, J., Bilogrevic, I.: (Smart)watch your taps: side-channel keystroke inference attacks using Smartwatches. In: *The 2015 ACM International Symposium*, pp. 27–30 (2015)
7. Michalevsky, Y., Boneh, D., Nakibly, G.: Gyrophone: recognizing speech from gyroscope signals. In: *Proceedings of the 23rd USENIX Security Symposium* (2014)
8. OpenStreetMap: OpenStreetMap Project. <https://www.openstreetmap.org/>. Accessed 16 Aug 2016

9. Narain, S., Sanatinia, A., Noubir, G.: Single-stroke language-agnostic keylogging using stereo-microphones and domain specific machine learning. In: Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks (2014)
10. Narain, S., Vo-Huu, T.D., Block, K., Noubir, G.: Inferring user routes and locations using zero-permission mobile sensors. In: Symposium on Security and Privacy (2016)
11. Savitzky, A., Golay, M.J.E.: Smoothing and differentiation of data by simplified least squares procedures. In: Analytical chemistry, pp. 1627–1639 (1964)
12. Senate Judiciary Committee: S.2270-Location Privacy Protection Act of 2015 (2015). <https://www.congress.gov/bill/114th-congress/senate-bill/2270>. Accessed 16 Aug 2016
13. Stewart, J.: Calculus. Cengage Learning, Boston (2015)
14. Wang, H., Lai, T., Choudhury, R.R.: MoLe: motion leaks through smartwatch sensors. In: Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (2015)
15. Michalevsky, Y., Schulman, A., Veerapandian, G.A., Boneh, D., Nakibly, G.: PowerSpy: location tracking using mobile device power analysis. In: Proceedings of the 24th USENIX Conference on Security Symposium, pp. 785–800, August 2015



Evolution of Spamming Attacks on Facebook

Minsu Lee¹ , Hyungu Lee² , and Ji Sun Shin² 

¹ Cyber Security Research Center, KAIST,
291, Daehak-ro, Yuseong-gu, Daejeon, Korea
minsulee@kaist.ac.kr

² Department of Computer and Information Security, Sejong University,
209, Neudong-ro, Gwangjin-gu, Seoul 05006, Korea
hyunguana@naver.com, jsshin@sejong.ac.kr

Abstract. Defence techniques against spamming attack have been introduced and developed in many different areas, such as e-mail, web, and even social network service, over the past decades. Whereas, we have still been suffering from the attack though the service vendors as well as academia have been making best effort on winning such arms race. Such being the case, Facebook have also been inevitable to confront spamming campaign in order not to be overwhelmed by massive junk messages. Certain spamming patterns have recently been remarkably common on Facebook which collaborates with other social network messenger. We study such the advanced spamming campaign so as to demystify how it has been worked and settled down on Facebook ecosystem. We build a crawler and analyser which collect 0.6 million of comments; afterwards, extracts the targeted spams. Our data shows that the spams are systematic, well-structured, obfuscated and even localized.

Keywords: Spamming attacks · Social network service
Localized spam analysis

1 Introduction

Facebook has dominated social networking services based on western culture-familiar countries. As the most popular social networking platform, the service has had 1.03 billion daily active users within 1.59 billion total users (at the end of 2015) as well as 2 trillion posts has accumulatively uploaded until now [4]. These facts indicate that Facebook is still leading the SNS market and conquered a significant portion of the users' daily lives. Through Facebook, not only communicate users with each other, but they also create, share, and scrap diverse pieces of information on their own pages or others.

Although most information on Facebook is showed up depending on dozens of factors such as friends, personal tastes, friends' interests, adverts and so on, it is rather to be said 'exposed by' an information flood without intention. Considering characteristics of social network services, it is more likely to show users anything somehow connected to usually their friends or often even friends of friends. Even though users can control what will be showed or blocked in their 'News Feed', it seems difficult hiding from the fashion of the moment in this age of SNS; every moment of everyone is uploaded even at an accident moment nowadays.

Almost Facebook users are exposed indiscriminate information, whether it is profitable or not, against their will, as we mentioned, and this condition makes Facebook spams age arrive also. As Facebook gets old and upgrades to a new version, more people join in and more users do it more easily and simply, but studies for increasing spams in Facebook are at standstill. Studies for them, of course, have been in progress, but they are not sufficient to catch up spams trends and as many as spams increase.

In Korea, spams have increased in Facebook without exception and now they are rampant; even many benign Facebook users are upset and feel uncomfortable with doing Facebook. Most page managers and famous personal users, having many followers, have tried to block spams by hook or crook, but it seems unequal to block increasing and changing spams. Among them, a new mean bothering both the benign users and managers is appeared recently and we especially focus on it in this paper. The new mean uses not only Facebook but Kakaotalk, a messenger application in Korea. A problem is that Kakaotalk is the most popular messenger and hence spams in Facebook become more powerful and tricky now.

Based upon the situation on Facebook, we built simple applications, a crawler and analyser, in order to investigate Facebook spamming comments. The crawler was able to collect approximately 0.6 million of comments for 20 days. Afterwards, the analyser examined the spamming comments on the basis of certain keywords including 'Kakaotalk'. Not only did we find that 'Kakaotalk' related comments have been prevalent, but we also discovered that 'comment' related comments have been overwhelmed the spamming ecosystem.

Our contribution in this paper is that alerting and informing about the new spams with structure which shows how they work, data we collected in Facebook, and statistics. We also discuss former studies for other types of spams in Facebook. We conclude this paper with the result we analyzed and more studies needed to do in the future.

2 Related Works

Intuition that increasing frequency of exposure to Facebook spamming through comments (even with Facebook posts) and outrage at the spamming messages triggered to study deeper in Facebook spams. Even though users' dissatisfaction with cohabitation comments with massive spams has been arguably detected for the past couple of years, the studies related to Facebook spam were less likely to be dealt with as a high priority. Considering that most of the papers related to Facebook spams were written in the initial era of Facebook, we contrast our contribution with previous works in this chapter.

To protect users and the social graph, containing user information, from phishing attacks and malware, Stein et al. [7] described Facebook Immune System (FIS). Instead of more traditional learning system, FIS uses an adversarial learning system that is responsive and scalable. The authors coped simply with spamming attacks, as a small piece of attacks, throughout the entire Facebook threats. The challenges for FIS to improve its mechanisms are remained.

One of the earliest papers which covered the Facebook spams (with the malicious posts) indicated the increasing exposure of users to malicious activities on SNS. Abu-Nimeh et al. [2] used the Defensio Facebook application to classify and determine posts as legitimate, spam, or malicious, and protect users from spams and malicious posts. According to the study, the Defensio consists of two components: one is for detecting spam posts/comments and the other is for detecting malicious URLs. They, at the end of the paper, urged that the more research in this area should be necessarily needed.

On Facebook, depending on intention of hackers, socware is possible to be hosted either on or outside Facebook (e.g., URL). Existing security mechanisms, however, are not sufficient against socware so that Rahman et al. [6] presented MyPageKeeper, the Facebook application they developed to protect Facebook users. MyPageKeeper consists of six modules and uses specific classification algorithms to identify and detect socware. According to their data, it shows that 97%, 58,388 out of 60,191, are true positives.

Forms of spams in social media network are usually texts, images, and social network behaviours so that Jin et al. [5] proposed a spam detection system that considers features of them. This detection system collects spam activities automatically, performs online active learning, and sends warnings to users.

Those two papers went one step further that not only did they study Facebook spams in their own ways perfectly, but they also built their own anti-spammers as a Facebook application to detect malicious activities on Facebook.

To identify spam profiles on Online Social Networks (OSNs), Facebook and Twitter, Ahmed and Abulaish [3] considered social networks as four basic components and provided a set of 14 generic statistical features. They used three different classification algorithms, naïve Bayes, Jrip, and J48, to analyze properties of identified features. According to their experiments, J48 classification outperformed the others in dataset of both Facebook and Twitter in the last result.

However, our approach outweighs more on studying in Facebook spamming comments recently occurred in the Facebook ecosystem. As the time gap between the previous works and the recent Facebook system is not trivial, the assumption that spams also greatly evolved for the past years should be take into account. So, we picture the evolution of the Facebook spam sceneries in this paper via investigating Facebook graph API.

3 Advanced Spamming Attacks

Although many old Facebook spamming-related papers tried to demystify spamming ecosystems on Facebook, current spamming attacks are rather more clever and slyer. The old school Facebook spammers used to merely spread spams out to someone's Facebook walls or Facebook pages to expose malicious URLs with attractive comments. As the old fashioned spams heavily relied on URLs, the previous papers focused consequently on the URLs in order to identify spams.

Unlike the previous spams, a new type of spams is less likely to rely on URLs. The new spammers have abused Facebook as an outpost for their malicious service advertisements rather than directly connect between Facebook users and the malicious services through the URLs. From a slight evolution of the spamming strategy, it seems that the existing anti-spam schemes, that are especially for the keyword based anti-spam systems, would not properly work for the recent spams. The new spamming scheme is first of all explained as well as characteristics of the recent spams are described in the following sections.

3.1 New Spamming Scheme

The new spamming propagation strategy is consisted of three phases, which are Facebook, Kakaotalk, and Web phase. Each phase has its unique role such that it provides segregation of each part, which breaks one another's connectivity. Due to the disconnection between spams and actual malicious services, a chance to detect spamming activities is assumably low via existing spam detection systems.

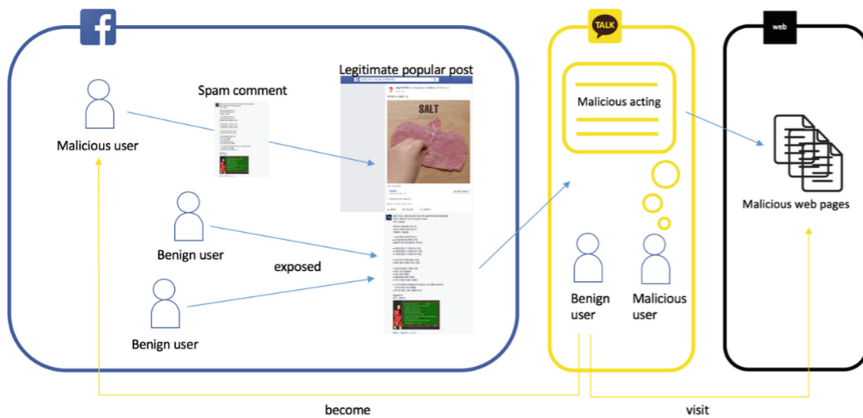


Fig. 1. Spamming flows on Facebook. The flows go through three sections: Facebook, Kakaotalk, and web pages. The Facebook section, on the left hand side, shows the spread of the spams on Facebook. Kakaotalk, in the middle, is a messenger application, which dominates a Korean messenger market. The web, at the end, is an actual destination where potential victims (benign users) finally access on

Facebook: The Facebook phase is literally where the spams are spread out and observed; the benign users are exposed to the malicious activities. In the Facebook phase, malicious activities are practically taken place through commenting on relatively popular Facebook posts. Spammers simultaneously and consistently leave the same comments, which include spam, on the target posts. Since the spamming comments steadily overwhelm the other legitimate comments on the target posts with popularity

of Facebook pages where the target posts belong to, the exposure of the spams can easily be maximized to benign Facebook users.

Kakaotalk: The Kakaotalk phase is a core of the new Facebook spamming scheme in the recent Facebook spamming trend. “Kakaotalk” is a messenger application, which dominates the Korean market (more than 90% of Korean people use this application [1]). Unlike the previous spamming scheme, the biggest difference is that the recent spamming strategy hires a broker in between Facebook and Malicious services. The messenger, Kakaotalk, as a mean of the broker segregates Facebook and web services, so that a direct access on to the service is less likely to be taken place but the victims primarily face the broker via the messenger. Considering the previous spams, that are merely broadcasting their services, exposing Kakaotalk IDs and dealing with the benign users out of Facebook have a huge advantage in disguising spamming activities.

Web: The web page is a final destination in consequence of succeeding in spamming attacks. Although all spamming attacks do not lead the victims to their online services, a vast majority of the attacks aim to attract the benign Facebook users to the web pages. Those online services usually cope with illegal web services, such as online casino, online sports betting, and pornography.

3.2 Targets (New Spams)

One common characteristic of the new spam is exposing Kakaotalk IDs, which are meaning nothing and randomly created so that there are no lexical connections with each other. As mentioned earlier, there was a change of the way the spammers entice benign users, and Kakaotalk is one at the center of the change. They have to expose, therefore, IDs somehow and finally make benign users to visit their web pages.

There are other characteristics of the new spam: they mimic benign comments, attracts others interest, or exaggerates and glamorizes their intention and what they do. Mimicking benign comments is to trick benign users to read them to the end where photos, Kakaotalk IDs, or URLs are exposed. They usually quote famous stories or meaningless conversations unrelated to the posts. Also, spammers even fake the numbers of ‘like’ of their comment sometimes and use familiar photos but containing spam texts or sometimes URLs to attract the users’ eyes. Once the users start to read the comments somehow, their eyes and attention will reach where the spammers’ secret design is soon. For this reason, the spammers exaggerate and glamorize their messages and use words that involve users more easily such as ‘safe’, ‘family’, or ‘ensure’.

The last considerable feature which should take into account is that the spamming ecosystem where we especially concentrated on basically the Korean written spams settled down Korea-based pages. Since the spamming attack is specialized in the Korean digital culture, the spams can be said that it always contains Korean letters rather than any other countries’ languages (Fig. 2).

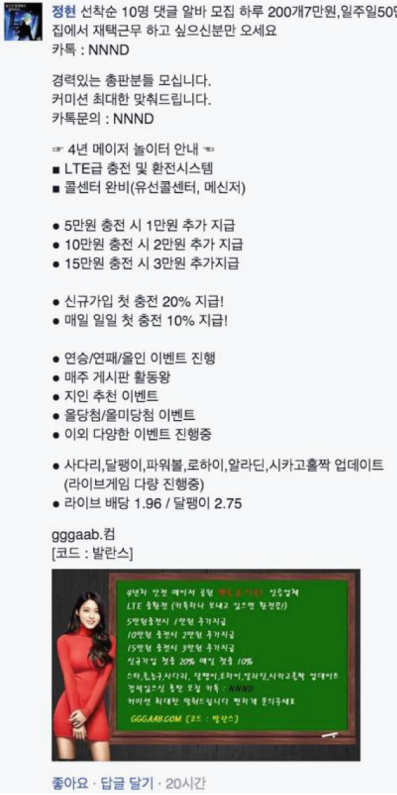
 <p>정원 선착순 10명 댓글 알바 모집 하루 200개7만원,일주일50만원 집에서 재택근무 하고 싶으신분만 오세요 카톡 : NNND</p> <p>경력있는 총판분들 모집입니다. 커미션 최대한 맞춰드립니다. 카톡문의 : NNND</p> <p>☞ 4년 메이저 놀이터 안내 ☞</p> <ul style="list-style-type: none"> ■ LTE급 충전 및 환전시스템 ■ 콜센터 완비(유선콜센터, 메신저) <ul style="list-style-type: none"> ● 5만원 충전 시 1만원 추가 지급 ● 10만원 충전 시 2만원 추가 지급 ● 15만원 충전 시 3만원 추가지급 <ul style="list-style-type: none"> ● 신규가입 첫 충전 20% 지급! ● 매일 일일 첫 충전 10% 지급! <ul style="list-style-type: none"> ● 연승/연패/올인 이벤트 진행 ● 매주 게시문 활동량 ● 지인 추천 이벤트 ● 올담침/올미담침 이벤트 ● 이외 다양한 이벤트 진행중 <ul style="list-style-type: none"> ● 사다리,달팽이,파워볼,로하이,알라딘,시카고홀짝 업데이트 (라이브게임 다양 진행중) ● 라이브 배당 1.96 / 달팽이 2.75 <p>ggaab.컴 [코드 : 발란스]</p> <p>좋아요 · 답글 달기 · 20시간</p>	<p>Recruit first 10 writing comment part-timers who want to work at home for 70,000 won per 200 comments a day (500,000 won per one week.)</p> <p>Kakaotalk : NNND</p> <p>Experienced sole distributors are very welcome.</p> <p>We will satisfy commission as far as possible.</p> <p>Questions (Kakaotalk) : NNND</p> <p>☞Fourth year major playground information☞</p> <ul style="list-style-type: none"> ■ A LTE-speed charging and exchanging system ■ Fully equipped call center services (wire telephone, messenger) <ul style="list-style-type: none"> ● charge 50,000 won and get extra 10,000 won ● charge 100,000 won and get extra 20,000 won ● charge 150,000 won and get extra 30,000 won <ul style="list-style-type: none"> ● Extra 20% at first charge for new members! ● Extra 10% at first charge of a day! <ul style="list-style-type: none"> ● Winning streak/Losing streak/all in events ● Top active user in bulletin board every week ● Acquaintances recommend event ● Win all events/lose all events event ● There are more events as well <ul style="list-style-type: none"> ● Update: ladder, snail, powerball, lohigh, Aladdin, and Chicago even and odd (There are many live-games) ● Dividend rate of live games 1.96 / 2.75 for snail <p>ggaab.com [code : balance]</p>
---	--

Fig. 2. A typical example of the targets. (a) The original, written in Korean. (b) The English version of the original

4 A Design and Implementation

In order to look deeply into the Facebook spamming ecosystem, we first of all built a crawler. With the program, Facebook comments were collected among popular posts within famous pages. The next stage was to analyse the collected data based on keywords using an analyser we created.

4.1 Data Collection

The crawler, which collects Facebook data, was written in Python. Luckily, the data were able to be directly retrieved as the Facebook data can be publically accessible via the graph API provided by Facebook. The public data as a JSON format were then interpreted to a readable form afterwards saved in a format of a CSV file. The CSV files

were separately managed depending on categories, Facebook pages, posts, and comments in the end.

The simple program simply crawls Facebook posts first, and then collects the comments from each post respectively. The data were collected from 16th of January to 2nd of February in 2016, for around 20 days. To effectively retrieve spamming comments, we considerably chose candidate Facebook pages. The pages were selected regarding on popularity of the page, which the popularity threshold was set in case at least 0.3 million of Facebook users had liked. On the basis of the threshold, 23 pages were chosen and monitored during our research.

Based on the selected 23 pages, the uploaded posts were crawled. Each of the posts has been only uploaded by the owners of the pages, which means we could assume that the posts are not related to adversaries or the spammers. Since the interval between real data were uploaded on Facebook and the real data were recorded on the graph API exists as well as we are not able to measure when new posts would be uploaded, the crawler were set to visit each post every 6 h. In total, 476 posts were crawled from the popular pages within the research period.

When comments were harvested in the target posts, all of the information provided on the graph API was recorded, including user IDs, user names, dates, and comments. Like the way of retrieving posts, the JSON format was decoded to the readable format so as for data analysis. Approximately 0.6 million of comments were consequently collected for the research period.

4.2 Data Analysis

The collected comments were analysed to investigate how spams have positioned on Facebook. Of around 0.6 million of comments, The spams were then studied based on keywords such as ‘카톡 (the abbreviation for Kakaotalk)’ and ‘댓글 (comments)’. The main keywords had especially selected regarding ‘Kakaotalk’, the messenger application as discussed in Sect. 3.1, as well as ‘comments’ was deeply investigated since it was found that the keyword had been overwhelmed the Facebook spamming ecosystem during the research.

In order to analyse the spamming comments, an analyser was developed, which was also written in Python. The program simply explores the collected Facebook comment database so as to discern whether spamming comments or benign comments. The spamming comments were filtered on the basis of a combination of several keywords in order not to include benign comments. The analysed results were double-checked as well as some comments were manually re-investigated to minimize a false-positive rate.

5 Results

As discussed in the Sects. 3 and 4, the Facebook spams are investigated and pictured based on ‘Kakaotalk’ and ‘comments’ in this section. Just before diving deeper into the research results, some references should be introduced first as the keywords are consisted of unconventional words in terms of Korean slangs. The appropriate translation

will be explained based on the most significant 6 keywords, which we chose based on the target specified. The interpretation table is as below:

Table 1. Interpretations of spam keywords table

Korean keywords	Interpretations
카톡	The word ‘카톡’ stands for an abbreviation of ‘Kakaotalk’
놀이터	This literally means ‘playground’ which also can be used as ‘casino’ in this context
추천인	The word is exactly the same as a referee
댓글	The word means ‘comments’
알바	This term ‘알바’ is an abbreviation of a ‘part time job’
충전	The term ‘충전’ literally means ‘recharge’ but the word stands for ‘to get cyber money (with free of charge)’ in this context

The collected comments were analysed to investigate how spams have positioned on Facebook. The spams were then studied based on keywords such as ‘카톡’, ‘댓글’. We chose specific keywords not in common usage with ‘카톡’ and also almost spammers used in their comments: ‘카톡’, ‘놀이터’, ‘추천인’, ‘댓글’, ‘알바’, and ‘충전’. Kakaotalk plays important roles as one of the three phases, we mentioned in 3.1, and therefore it seems that most malicious comments should include “카톡”. Among such keywords, thus we especially focused on ‘카톡’ and collected statistics on the keywords based on the collected comments.

Of those keywords related to ‘카톡’, the spams can be categorized of two illegal activities (i.e., illicit online casinos and advertisements to recruit new spammers). The word combination ‘카톡’ with ‘놀이터’, ‘추천인’, and ‘충전’ directly reveals its identity that the spams are concretely related to the online casino services. The other two words ‘댓글’ and ‘알바’ are specific terms which attracts benign users to hire as a new spammer.

‘카톡’ (Kakaotalk): Fig. 3 shows the result that how many spamming comments that include the single keyword without “카톡” or with “카톡” are collected and compares the number of each case. In case of “놀이터”, 452 are spams of 453 comments and 237, 52%, include “카톡”. The others usually include “카카오톡”, a literal translation of Kakaotalk, “단톡방” or “가족방”. Of 715 suspected comments of “추천인”, 571 include “카톡” and it is about 80%. “댓글”, most commonly used as a single word by users, benign or malicious, in the comments has 735 spams with “카톡” and it is 16% of the numerous comments of “댓글”. 1196 comments including “알바” have 787, 65%, spam comments with “카톡”. Among 904 suspicious comments of “충전”, 702 spams include “카톡” and it is 77%.

As shown in the statics of Fig. 3 and the percentages above, the comments including both one of the keywords and “카톡” have higher probabilities as a spam except for the case of “댓글”. This fact strongly proves that Kakaotalk is one of the key points, which is used to connect Facebook spams and the actual malicious services. As the messenger has been positioned in the middle of the Facebook spamming

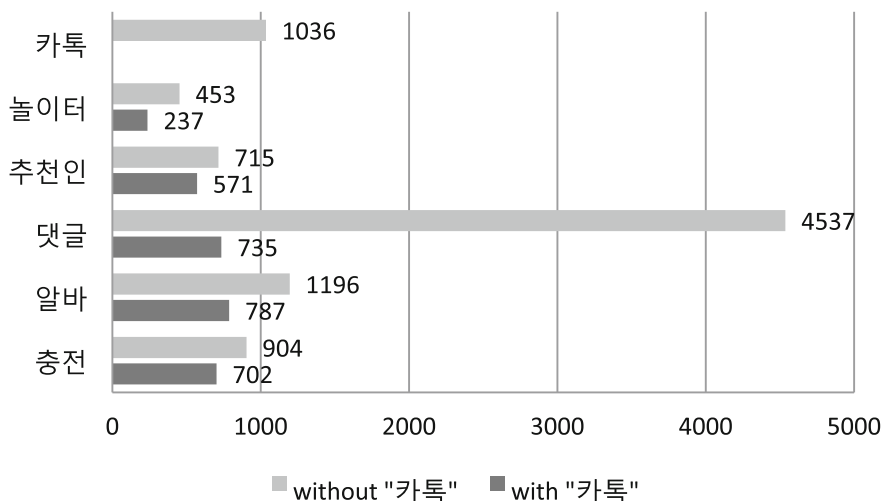


Fig. 3. A comparison of collected comments with keywords, given in Table 1 with English interpretation

ecosystem, the spamming comments including ‘Kakaotalk’ seem effectively to disguise themselves as benign comments as well as the messenger has been abused to provide further information of illicit services.

‘댓글’ (comments): One remarkable feature that we found during the research is the outstanding number of ‘comments’ keywords in the spamming comments. The ‘댓글’ related comments were intentionally hired in order to recruit new spammers. The keyword ‘알바’, a part time job, is followed right after ‘댓글’, which a combination of the two keywords completes the advertisement that spammers are hiring other Facebook spammers in the presence of Facebook.

The assumption that we have first made was merely focusing on digging Facebook spams up regarding ‘Kakaotalk’. However, such the result was unexpected and quite surprising in an aspect of a spamming propagation strategy. Unlike the other spamming attacks such as e-mail and popup ads as typical online spamming technique, the Facebook spammers try to hire new spammers on Facebook where they propagate spams. In other words, both recruiting and spreading malicious comments have been taken place in the Facebook posts as comment forms.

Consider the significant number of ‘댓글’ related comments, not only are ‘Kakaotalk’ related comments problematic, but the ‘댓글’ related comments are also troublesome. Looking at the Fig. 1, the arrow line connecting from benign users to malicious users (become line) can be stronger than it used to be presumed. The fact seems that this kind of spams should be dealt with as a higher priority in order to reduce spams due to its characteristics.

In addition the ‘댓글’ related comments are less likely to rely on ‘Kakaotalk’, which indicates that the spamming comments are rather independent from a ‘Kakaotalk’ strategy. Furthermore, as mentioned before (about the portion of ‘댓글’ related

comments), it is seemingly assumed that it may a more effective way of propagating spams as spammers can be recruited in the same place where spams are released.

6 Discussion

Since Facebook graph API had been updated from v1.0 to v2.0 (released April 30th, 2014), information researchers (or any others) can access and be provided is only about Facebook pages' dataset. Such being the case, accessing spamming comments is limited to the comments written on Facebook pages. Things making matters worse are accounts who imitate Facebook pages; they have lots of followers, upload posts, and manage their Timeline like Facebook pages (e.g., <https://www.facebook.com/jaebigood>). With personal IDs, it was unable to retrieve their posts and comments of such accounts. More to the point, some of the posts of theirs have the contents same with the spamming comments we targeted. We presume that persons who own such accounts are strongly connected somehow with the malicious users (or illicit services) who write the spamming comments. It seems that more studies and researches are needed.

Our study excluded spamming comments that have both an image containing spamming contents and meaningless texts because accessing images in comments are impossible with graph API. We believe that attacking means keep moving from a text spamming to an image spamming and therefore more studies focusing on such spam type are needed.

In this paper, we focused on Kakaotalk, the most commonplace messenger in Korea, and guess that other commonplace platforms in other countries are also used as one of the spamming means in various ways. Considering the Facebook service as a global SNS platform, we assume that other countries have also been suffered from such attacks like we are facing on Facebook in everyday life. More studies would be required for all other countries against the advanced spamming attacks.

7 Conclusion

Spamming trends have changed as time goes by and are evolved in different countries, but Facebook, the worldwide popular SNS, seems to be a crux and last for a long time. The brand-new spamming comments we are exposed are indeed a sort of routine for now whenever turning on Facebook world. Although the online casinos usually hire Facebook spamming comments so as to advertise their services, a vast majority of the Facebook spams are, in fact, surprisingly posted to recruit new spammers.

The research we detailed, however, also alert that it is by no means a panacea seeking solutions for just Facebook. There was one more key mean of spamming, Kakaotalk, and hence solutions for the new trend, using both Facebook and Kakaotalk, are also needed. Studies for upcoming spamming trends and what we excluded (e.g., spamming comments with images) remains as an important issue for the age, we live in, of SNS.

Acknowledgements. This research was supported by the Korea Government (the Ministry of Science, ICT and Future Planning) under Basic Science Research Program (NRF-2013R1A1A3009163) through the National Research Foundation of Korea (NRF), and supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (No. NRF-2016M3C4A7937117).

References

1. Appstickers: Kakao Talk: 152 Million Users and 93% Market Share in South Korea. <http://appstickers.net/kakao-talk-152-million-users-and-93-market-share-in-south-korea/>. Accessed 1 Jan 2016
2. Abu-Nimeh, S., Chen, T.M., Alzubi, O.: A Survey of Malicious and Spam Posts in Facebook (2011)
3. Ahmed, F., Abulaish, M.: A generic statistical approach for spam detection in Online Social Networks. *Comput. Commun.* **36**(10), 1120–1129 (2013)
4. DMR: By the Numbers: 200 Surprising Facebook Statistics. <http://expandedramblings.com/index.php/by-the-numbers-17-amazing-facebook-stats/>. Accessed 1 Jan 2016
5. Jin, X., Lin, C., Luo, J., Han, J.: A data mining-based spam detection system for social media networks. *Proc. VLDB Endow.* **4**(12), 1458–1461 (2011)
6. Rahman, M.S., Huang, T.K., Madhyastha, H.V., Faloutsos, M.: Efficient and scalable socware detection in online social networks. Presented as Part of the 21st USENIX Security Symposium (USENIX Security 2012), pp. 663–678 (2012)
7. Stein, T., Chen, E., Mangla, K.: Facebook immune system. In: Proceedings of the 4th Workshop on Social Network Systems (ACM), p. 8 (2011)

Systems Security and Authentication



Model Parameter Estimation and Inference on Encrypted Domain: Application to Noise Reduction in Encrypted Images

Saetbyeol Lee and Jiwon Yoon^(✉)

School of Information Security, Korea University, Seoul, South Korea
{morningstar_03,jiwon_yoon}@korea.ac.kr

Abstract. One of the major issues in security is how to protect the privacy of multimedia big data on cloud systems. Homomorphic Encryption (HE) is increasingly regarded as a way to maintain user privacy on the untrusted cloud. However, HE is not widely used in machine learning and signal processing communities because the HE libraries are currently supporting only simple operations like integer addition and multiplication. It is known that division and other advanced operations cannot feasibly be designed and implemented in HE libraries. Therefore, we propose a novel approach to building a practical matrix inversion operation using approximation theory on HE. The approximated inversion operation is applied to reduce unwanted noise on encrypted images. Our research also suggests the efficient computation techniques for encrypted matrices. We conduct the experiment with real binary images using open source library of HE.

Keywords: Image processing · Homomorphic encryption
Leveled fully homomorphic encryption · Statistical analysis
Cloud security

1 Introduction

We are currently living in a data deluge era, where large-scale data have been continually generated and accumulated from a variety of sources. Therefore, many data scientists and engineers in various academic communities, including machine learning, statistics, and signal processing, have developed several sophisticated analysis techniques for big data. However, there are security issues in data analysis on the cloud because data is stored and processed, not in the client's, but in the cloud system's storage. Because of this, various privacy preserving approaches, which process and analyze the data on an encrypted domain, have been introduced by much literature.

Homomorphic Encryption (HE) is now considered a powerful solution for providing the security in the cloud system since it is possible to evaluate a

function on encrypted data. That is, the encrypted data are processed in their format without decryption in the cloud. And the advent of Fully Homomorphic Encryption (FHE) scheme [6], which has no limitation on performing additions and multiplications on ciphertexts, has caused a great development in HE. However, FHE has a fundamental limitation in practical implementation although all computational operations can be designed in theory - it is not straightforward to perform complicated operations including division, comparison and conditional branching on ciphertexts. For this reason, FHE was considered hard to be used for real dataset.

In this paper, we overcome the limitation of FHE and show its applicability to real data. We introduce a methodology for applying FHE to real image filtering algorithm. We first explain the background for image filtering and leveled FHE in Sect. 2. Then we propose the inverse operation of the encrypted matrix in Sect. 3, and show how to apply this matrix inversion to the filtering technique in Sect. 4. We also show how to efficiently calculate encrypted matrices in Sect. 5, and Sect. 6 presents our experimental results. Finally, Sect. 7 concludes the paper.

Related Works. The division problem has been actively studied in the area of multi-party computation (MPC). Dahl et al. [7] uses Taylor series to approximate a division operation to a linear function. The authors divide a real number by its bit length, and then use a rounding function to convert it to an integer. Since the bit length of the number is private information, it is kept secret using MPC. Veugen [8, 9] also uses MPC for division approximation. In the paper, the author uses the additive blinding method in order to prevent the input value from being exposed to any parties.

There have also been substantial studies regarding computing a matrix inverse for solving regression problems of machine learning. Hall et al. [10] presented a MPC protocol to solve a linear regression problem on encrypted data. The approach uses Newton's method to compute a matrix inverse. By iterating some linear operations, it can approximate the inverse. Nikolaenko et al. [11] focused on a ridge regression problem by combining the garbled circuit theory [12, 13] with homomorphic encryption. In detail, the approach utilizes homomorphic encryption for linear operations, and garbled circuit for non-linear operations. Lu et al. [16] presented the method of statistical analysis, including linear regression, using FHE. They build the encrypted matrix primitives for data analysis.

After the research of Graepel et al. [14], studies combining HE and machine learning have increased. In [14], the authors present some binary classification algorithms on encrypted data. Similarly, Bost et al. [15] construct the secure comparison operation, and develop three classifiers using the operation.

Contributions

- **Division-free Matrix Operation:** In many statistical analyzes, inverse matrix computation is essential. However, FHE does not support this operation, which is a major obstacle to applying FHE to real data analysis.

We extend the application of FHE by newly introducing the inverse matrix operation method on ciphertext.

- **Training the model parameters on encrypted data:** In order to remove the noise of the image, it is necessary to train the model parameter of the image filter. Most studies that use statistical analysis with FHE focus on applying already trained model parameters to the data. However, training model parameters is the most important step in statistical analysis, and this paper focuses on training step. We show the applicability of this training algorithm by applying the proposed filtering technique to real binary images.
- **Two party model applicable to real-world cloud environments:** Most of the papers presented in related works section require a third-party to assist in the operation between the two parties. The third-party plays a role of authentication and data verification between a server and client. However, organizing the third-party may increase the budget of both sides. In this paper, we propose a novel approach which can run securely without the third-party.

2 Background

2.1 Image Smoothing Filter

Image filtering techniques can be divided into two major categories, low-pass filter and high-pass filter. Low-pass filter, as known as smoothing filter, serves to remove the noise in the image. In contrast, high-pass filter makes the image shaper, emphasizing fine details in the image. Both filtering works the same way, only different with the mask they use. In this paper, we only deal with the low-pass filter.

Image filtering process can be defined as follow:

$$y[m, n] = \sum_{i=-K}^K \sum_{j=-K}^K w_{i,j} x[m + i, n + j]. \quad (1)$$

This equation means that the filtered image pixel $y[m, n]$ is obtained by multiplying values of pixels near $x[m, n]$ by the mask $\mathbf{w} = \{w_{i,j}\}$. The mask \mathbf{w} is a $(2K + 1) \times (2K + 1)$ matrix and we only consider the case when $K = 1$ in this paper. The $K = 1$ case example is shown in Fig. 1.

For smoothing filter, we assume $w_{0,0} = 0$. That is, the filtered image pixel $y[m, n]$ is obtained by weighted sum of the nearby pixels. Then the weight parameter \mathbf{w} should be modeled for filtering. In general, the parameter can be simply estimated by least square estimate. In order to apply least square estimate, we need to transform the Eq. (1) into matrix form. The matrix form of Eq. (1) is as follows:

$$\mathbf{Y} = \mathbf{X}\mathbf{w} + \epsilon,$$

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{pmatrix} \in \mathbb{Z}_t^{N \times 8}, \quad \mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} \in \mathbb{Z}_t^{N \times 1}, \quad \mathbf{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_8 \end{pmatrix} \in \mathbb{R}^{8 \times 1}$$

where $\mathbf{x}_i^T = [x_1^i \cdots x_8^i]$, N is the number of training data, and t is an integer determined by the type of image. For example, $t = 256$ if the image is gray-scale image with all pixel values between 0 and 255. ϵ is signal noise, generally assumed as Gaussian noise. Now, using the least square approach, the weight parameter can be calculated by

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}. \tag{2}$$

Using the estimated weight parameter \mathbf{w} , we can filter the entire image by multiplying \mathbf{w}^* by the total image pixels.

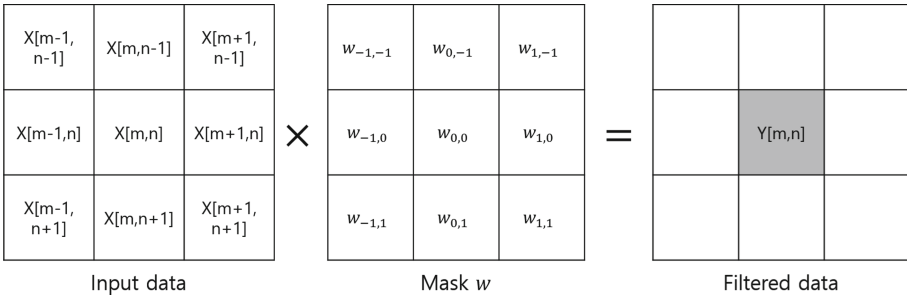


Fig. 1. Image filtering with 3×3 mask

2.2 Leveled Fully Homomorphic Encryption

In this work, we use the Brakerski-Gentry-Vaikuntanathan (BGV)’s scheme [2], which proposed the leveled fully homomorphic encryption. For the sake of simplicity, let $\mathbb{E}[\cdot]$ and $\mathbb{D}[\cdot]$ denote the encryption and decryption algorithm respectively.

The plaintext space of BGV’s scheme is $\mathbb{A}_{p^r} := \mathbb{Z}_{p^r} / \Phi_m(x)$, where p is a prime and $\Phi_m(x)$ is the m -th cyclotomic polynomial. Then the basic operations, addition and multiplication for $a, b \in \mathbb{A}_{p^r}$ work as follow:

$$\mathbb{D}[\text{Add}(\mathbb{E}[a], \mathbb{E}[b])] = a + b \pmod{\Phi_m(x), p^r}$$

$$\mathbb{D}[\text{Mul}(\mathbb{E}[a], \mathbb{E}[b])] = a \times b \pmod{\Phi_m(x), p^r}.$$

One of the most important points of BGV’s scheme is that SIMD is possible. CRT-packing algorithm in BGV’s scheme makes it possible to pack multiple plaintexts into one plaintext. If the cyclotomic function $\Phi_m(x)$ can be factorized into l -irreducible polynomials, that is $\Phi_m(x) = \prod_{i=1}^l F_i(x) \pmod{p^r}$, then

l -elements $\{a_i\}_{i=1}^l \in \mathbb{Z}_{p^r}^l$ can be packed into one element $\mathbf{a} \in \mathbb{A}_{p^r}$. It is said that the plaintext has l -slots, and each slot element a_i satisfies $a_i = a \bmod (F_i(x), p^r)$ for $i = 1, \dots, l$. CRT-packing algorithm increased the efficiency of the scheme, and we show the use of this techniques in Sect. 5.

3 Matrix Inverse Approximation (MIA)

Recall that it is impossible to perform a division operation on ciphertexts. So, we need to design a division-free method for seeking the inverse of encrypted matrices. We use the Neumann series which approximates the matrix inversion by using iterative method. The formula is as follows:

Theorem 1. *Suppose that a real matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ has the infinity norm $\|\mathbf{A}\|_\infty < 1$, then $\mathbf{I} - \mathbf{A}$ is invertible and its inverse is the series: $(\mathbf{I} - \mathbf{A})^{-1} = \sum_{i=0}^{\infty} \mathbf{A}^i$, where \mathbf{I} is the identity matrix. We can replace $\mathbf{I} - \mathbf{A}$ with \mathbf{B} , then we get:*

$$\mathbf{B}^{-1} = \sum_{i=0}^{\infty} (\mathbf{I} - \mathbf{B})^i \quad (3)$$

where the norm of \mathbf{B} follows $0 < \|\mathbf{B}\|_\infty < 2$.

We can also apply Eq. (3) for an arbitrary matrix $\mathbf{C} \in \mathbb{R}^{d \times d}$. If we multiply \mathbf{C} by a constant $t = 1/\|\mathbf{C}\|_\infty$, then the norm of $t\mathbf{C}$ becomes $\|t\mathbf{C}\|_\infty = 1$. Applying this property to Eq. (3), we have $\mathbf{C}^{-1} \approx t \sum_{i=0}^n (\mathbf{I} - t\mathbf{C})^i$. To use this equation for MIA, however, the real number elements of the matrix $t\mathbf{C}$ must be replaced with integers. So we use the round function, $\mathcal{D} = \lceil qt\mathbf{C} \rceil$, where $q = 10^\gamma$. Note that rounding on a matrix means rounding on every element of that matrix. As q increases, the matrix $\mathcal{D} \in \mathbb{Z}^{d \times d}$ loses less information from \mathbf{C} because the fractional part of a real number can be a corresponding integer. Then, we can approximate the integer form of $t\mathbf{C}$ as $t\mathbf{C} \simeq q^{-1}\mathcal{D}$.

However, the round function cannot guarantee that the norm of the matrix $q^{-1}\mathcal{D}$ falls within the range of 0 to 2. Since $\|qt\mathbf{C}\|_\infty = q$, the infinity norm of \mathcal{D} is in the range $(q - \frac{1}{2}d, q + \frac{1}{2}d]$ for $d \times d$ matrix \mathbf{C} . Then, we can see the range of $\|q^{-1}\mathcal{D}\|_\infty$ as follows:

$$1 - \frac{d}{2q} < \|q^{-1}\mathcal{D}\|_\infty \leq 1 + \frac{d}{2q}.$$

So if q is set to satisfy $d/2 < q$, we can guarantee that $q^{-1}\mathcal{D}$ has the infinity norm between 0 and 2. Subsequently, the matrix inverse \mathbf{C} can be derived as follows:

$$\begin{aligned} \mathbf{C}^{-1} &\approx t \sum_{i=0}^n (\mathbf{I} - q^{-1}\mathcal{D})^i \\ &\approx t \cdot q^{-n} \sum_{i=0}^n (q\mathbf{I})^{n-i} (q\mathbf{I} - \mathcal{D})^i. \end{aligned} \quad (4)$$

Finally, Algorithm 1 demonstrates MIA's overall protocol.

Algorithm 1. Overall protocol of MIA

-
- 1: Client calculates $t = \frac{1}{\|\mathbf{C}\|_\infty}$ and determines $q = 10^\gamma$.
 - 2: Client encrypts q and $\mathcal{D} = \lceil qt\mathbf{C} \rceil$. Then, it transfers $\mathbb{E}[q]$, $\mathbb{E}[\mathcal{D}]$ to the server with the approximation order n .
 - 3: Server calculates $\sum_{i=0}^n \mathbb{E}[q\mathbf{I}]^{n-i} \mathbb{E}[q\mathbf{I} - \mathcal{D}]^i$, and returns it to the client.
 - 4: Client decrypts the returned value and multiplies it by $t \cdot q^{-n}$. Then, it can build the approximated inverse matrix $\mathbf{C}^{-1} \approx t \cdot q^{-n} \mathbb{D} \left[\sum_{i=0}^n \mathbb{E}[q\mathbf{I}]^{n-i} \mathbb{E}[q\mathbf{I} - \mathcal{D}]^i \right]$.
-

4 Encrypted Image Filtering

In this section, we calculate the weight parameter of smoothing filter using the operation suggested in Sect. 3.

Data Sampling. Suppose we have a $M \times M$ size image, and each pixel is an element of \mathbb{Z}_t for an integer t . Then the image consists of a total of M^2 pixels, which is a great burden to use for weight parameter estimation. From this point of view, we use a sampling approach. That is, we sample some pixels for training the weight parameter, not using all pixels. For even sampling, we split the image into N grids and extract the center pixel. Then we have the training dataset $\{\mathbf{x}_i^T, y_i\}_{i=1}^N$. For the sake of simplicity, we assumed $w_{-1,-1}, w_{-1,1}, w_{1,-1}, w_{1,1} = 0$ in Fig. 1 and only use the adjacent 4 pixels for filtering. Figure 2 illustrates the process of sampling data and configuring a training set.

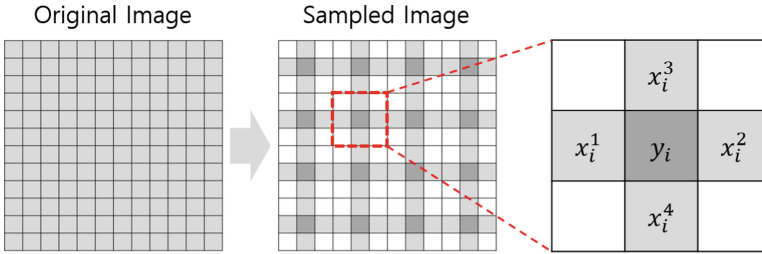


Fig. 2. Method for sampling the data

Model Training. As described in Sect. 2.1, the model parameter can be obtained by the equation $\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$. So, we use the MIA method to compute the inverse of $\mathbf{X}^T \mathbf{X}$. The overall algorithm using MIA is presented in Algorithm 2.

This algorithm has a limitation that once communication between client and server is needed, since it is impossible to calculate the eigenvalue $\|\mathbf{X}^T \mathbf{X}\|_\infty$ from $\mathbb{E}[\mathbf{X}^T \mathbf{X}]$. Of course, there is a way for client to calculate and encrypt $\mathbf{X}^T \mathbf{X}$, then upload it to server. But for large data set, this can be a burden on client and we focus on the convenience of the client rather than the server.

Algorithm 2. Overall Protocol of Weight Estimation

- 1: Client uploads the encrypted training data $\mathbb{E}[\mathbf{X}^T]$, $\mathbb{E}[\mathbf{Y}]$ to the server.
- 2: Server calculates $\mathbb{E}[\mathbf{X}^T \mathbf{X}]$ and sends it back to client.
- 3: Client decrypts $\mathbb{E}[\mathbf{X}^T \mathbf{X}]$ and calculates $t = 1/\|\mathbf{X}^T \mathbf{X}\|_\infty$. It also determines $q = 10^\gamma$.
- 4: Client encrypts q , $\mathcal{D} = \lceil q \cdot t \cdot \mathbf{X}^T \mathbf{X} \rceil$ and transfers $\mathbb{E}[q]$, $\mathbb{E}[\mathcal{D}]$ to the server with the approximation order n .
- 5: Server calculates $\left(\sum_{i=0}^n \mathbb{E}[q\mathbf{I}]^{n-i} \mathbb{E}[q\mathbf{I} - \mathcal{D}]^i\right) \mathbb{E}[\mathbf{X}^T \mathbf{Y}]$, and returns it to the client.
- 6: Client decrypts the returned value and multiplies it by $t \cdot q^{-n}$. Then, it can build the approximated weight parameter

$$\mathbf{w}^* \approx t \cdot q^{-n} \mathbb{D} \left[\left(\sum_{i=0}^n \mathbb{E}[q\mathbf{I}]^{n-i} \mathbb{E}[q\mathbf{I} - \mathcal{D}]^i \right) \mathbb{E}[\mathbf{X}^T \mathbf{Y}] \right]. \quad (5)$$

5 Efficient Computation Techniques

Since the computations on ciphertexts are very heavy, it is important to design an efficient algorithm with as few computations as possible. To do so, we utilize the CRT-packing techniques for matrix encryption and suggest some efficient operations for packed data.

5.1 Encrypted Matrix Operations

Halevi et al. [3] suggested three layouts for encrypting matrix: the row-major order, the column-major order, and the diagonal-major order. In this work, we utilize the row-major order, packing and encrypting each rows of a matrix separately. We use the matrix operation suggested by Lu et al. [16]. Since each rows of matrices are encrypted separately, we can denote the encryption of $d \times d$ matrices \mathbf{X} and \mathbf{Y} as $\mathbb{E}[\mathbf{x}_i^T]_{i=1}^d$ and $\mathbb{E}[\mathbf{y}_i^T]_{i=1}^d$ respectively. The methods proposed by [4] are as follows:

1. Matrix Addition

Addition can be performed simply, just adding each encrypted row:

$$\mathbb{E}[\mathbf{x}_i^T] + \mathbb{E}[\mathbf{y}_i^T] \text{ for } 1 \leq i \leq d.$$

2. Matrix Multiplication

Multiplication is more complicated than the addition operation. To evaluate $\mathbb{E}[\mathbf{X}] \cdot \mathbb{E}[\mathbf{Y}]$, we need to use the Replicate function. $\text{Replicate}(\mathbb{E}[\mathbf{v}], i)$ is the function which replaces all elements of the vector \mathbf{v} with the i -th element. Then using Replicate, we can multiply $\mathbb{E}[\mathbf{X}]$ and $\mathbb{E}[\mathbf{Y}]$:

$$\sum_{i=1}^d \text{Replicate}(\mathbb{E}[\mathbf{x}_j^T], i) \cdot \mathbb{E}[\mathbf{y}_i^T] \text{ for } 1 \leq i \leq d. \quad (6)$$

This matrix multiplication method reduces the complexity from $\mathcal{O}(d^3)$ to $\mathcal{O}(d^2)$.

Algorithm 3. Optimized MIA Computation

```

1: Input:  $\mathbb{E}[\mathbf{q}], \mathbb{E}[q\mathbf{I} - \mathcal{D}]$ , order  $n$ 
2: Output:  $result = \sum_{i=0}^n \mathbb{E}[q\mathbf{I}]^{n-i} \cdot \mathbb{E}[q\mathbf{I} - \mathcal{D}]^i$ .
3:
4:   Initialize  $temp = \mathbb{E}[\mathbf{I}]$ 
5:   for( $i = 0; i < n - 1; i++$ )
6:      $result += \text{Replicate}(\mathbb{E}[\mathbf{q}], n - i) \cdot temp$ 
7:      $temp *= \mathbb{E}[q\mathbf{I} - \mathcal{D}]$ 
8:    $result += temp$ 
9: return  $result$ 

```

5.2 MIA Computation

Recall Eq. (4) of MIA. In order to obtain the inverse of the matrix \mathbf{C} , server should compute $\sum_{i=0}^n \mathbb{E}[q\mathbf{I}]^{n-i} \cdot \mathbb{E}[q\mathbf{I} - \mathcal{D}]^i$. To calculate this expressions in order requires $\mathcal{O}(nd^2)$ multiplications. Furthermore, since $E[q\mathbf{I}]$ and $\mathbb{E}[q\mathbf{I} - \mathcal{D}]$ are vectors with d -ciphertexts, it occupies very large memories. In this section, we suggest a faster and more memory-savvy way to compute MIA. We choose to pre-calculate q^n instead of calculating a heavy matrix multiplication. So client encrypts a vector $\mathbf{q} = [q^1 q^2 \cdots q^n]$ using CRT-packing. By storing a scalar vector \mathbf{q} , server doesn't need to store a large ciphertext matrix. The detailed algorithm is described in Algorithm 3.

Compared to conventional computations, the number of multiplications between two encrypted vectors, represented in Eq. (6), has been reduced by $n(n+1)d$ to $n(d-1)$ through this optimization technique. Also move operation, which is the heaviest operation in CPU, reduced from $3d$ to $2d$.

6 Experiment

For the simplicity, we use a binary image with all pixel values 0 or 1 for our experiment. The size of the image is 100×100 and consists of a total 10,000 binary pixels. We use the sampling method described in Sect. 2.1 to extract 100 data and use it as a training set.

6.1 Error Rate Measurement

Since we use approximation techniques for inverse matrix computations, we need to define the error rate for accuracy measurements. First, we use F1-score to measure the accuracy of image filtering. F1-score is often used in signal processing because it considers recall as well as precision. The definition of F1-score is presented in Fig. 3, where 0 and 1 means the pixel values of the filtered image.

That is, **Precision** means the ratio of pixels having a value of 1 actually among the results filtered by 1, and **Recall** means the ratio of pixels filtered by 1 of the pixels of the actual 1. F1-score is defined as the harmonic mean of **Precision** and **Recall**.

		Our Image Filtering Result	
		1	0
Image Filtering Result	1	a	b
	0	c	d

Precision = $\frac{a}{a+c}$

Recall = $\frac{a}{a+b}$

F1 - score = $\frac{2*Precision*Recall}{Precision+Recall}$

Fig. 3. The definition of F1-score

In addition to F1-score, we also use RMSE(Root Mean Square Error) to measure the accuracy of the model parameter \mathbf{w} . Let \mathbf{w} and \mathbf{w}^* denote the estimated weight using the original scheme and our scheme, respectively. The definition of RMSE is as follows:

$$RMSE = \frac{\|\mathbf{w}^* - \mathbf{w}\|_2}{\sqrt{|\mathbf{w}^*|}}$$

We measure the error rate of the three images using the error rate defined above. Figure 4 shows the result image of our image filtering and its error rate.

6.2 Performance Test

We first used two PCs with Intel(R) Xeon(R) CPU E5-2609 0 @ 2.40 GHz, 64 GB RAM, and Ubuntu 16.04.1 LTS (64-bit) OS. We also used C++ as the programming language.

We used HElib by Halevi and Shoup [5]. This library is based on Brakerski-Gentry-Vaikuntanathan (BGV)’s HE scheme [2]. The parameter p, r, m determines the plaintext space $\mathbb{A}_{pr} := \mathbb{Z}_p / \Phi_m(x)$. m is determined by the function FindM in HElib. m is dependent on prime p , security level δ , and the tolerable noise level L . Also plaintext has l -slots using CRT-packing. In this paper, we use zero-padding in the empty slots. We run performance tests on two cases.

1. $p = 2, r = 32, L = 32, \delta = 64, m = 24929, l = 512$.
2. $p = 2, r = 32, L = 50, \delta = 64, m = 43691, l = 1285$.

The biggest difference between these two cases is L . A high L ensures more operations, but the larger the L , the more exponentially the computation time.

Recall the Eq.(5). Server should compute $\left(\sum_{i=0}^n \mathbb{E}[qI]^{n-i} \mathbb{E}[qI - \mathcal{D}]^i\right) \mathbb{E}[\mathbf{X}^T \mathbf{Y}]$. However, the first case, $L = 32$, cannot support the whole operation. Therefore, we assume that when the server computes $\mathbb{E}[\mathbf{X}^T \mathbf{Y}]$ and sends it to the client, the client re-encrypts it. Therefore, we only measure the time to multiply the already calculated $\mathbb{E}[\mathbf{X}^T \mathbf{Y}]$. In the second case, $L = 50$, the entire operation is possible, so we measure the time that the server performs the whole operation with $\mathbb{E}[\mathbf{X}^T]$ and $\mathbb{E}[\mathbf{Y}]$.










Original Image	Filtered Image	Our Filtered Image	F1-score	RMSE
			0.9962	0.0498
			0.9105	0.0518
			0.9820	0.0546

Fig. 4. Comparison of conventional filtering and our filtering application. We use three images for the experiment. For the original images in the first column, the second column contains filtered images, where the weight is trained on plaintext space. The images in the third column are filtered on ciphertext space using our scheme. The last two columns represent F1-score and RMSE for each image. We use the parameter $q = 10$ and $n = 4$.

Table 1. The result of performance test

L	n	MIA evaluation (s)	The rest evaluation (s)	Decryption (s)
32	1	1.936	1.66	10.840
	2	142.192	2.26	10.796
	3	283.852	2.72	10.844
	4	419.532	3.224	10.620
L	n	MIA evaluation (s)	The rest evaluation (s)	Decryption (s)
50	1	5.744	273.628	34.312
	2	725.996	280.028	34.752
	3	1423.92	276.684	34.436
	4	2181.36	281.624	34.724

Table 1 shows the result of performance test. MIA evaluation means the part of computing $\sum_{i=0}^n \mathbb{E}[qI]^{n-i} \mathbb{E}[qI - \mathcal{D}]^i$, and the rest evaluation means multiplying $\mathbb{E}[\mathbf{X}^T \mathbf{Y}]$ by the MIA result. Each result is the average of the results for the above three images. Both cases are fixed at $p = 10$ because the size of the ciphertext is independent of p , so there is no significant impact on performance time. Decryption time is independent of order n , because n also does not affect the size of the ciphertext.

7 Conclusion

In this paper, we presented a method to apply FHE to image filtering. To calculate the weight of a filter in an encrypted image, we propose an Matrix Inverse Approximation technique that does not require division. We propose an algorithm that removes the noise of the real encrypted image using the MIA technique and suggest an mathematical optimization technique that can efficiently perform the algorithm. Finally we performed the experiment by applying this algorithm to the actual binary image. We conclude that our MIA and optimization techniques will make practical use of FHE.

Acknowledgments. This research was supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under the IITP (Institute for Information & communications Technology Promotion) support program (2017-0-00545).

References

1. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, vol. 9, pp. 169–178 (2009)
2. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory (TOCT)* **6**(3), 13 (2014)
3. Halevi, S., Shoup, V.: Algorithms in HELib. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 554–571. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_31
4. Lu, W., Kawasaki, S., Sakuma, J.: Using fully homomorphic encryption for statistical analysis of categorical, ordinal and numerical data. In: IACR Cryptology ePrint Archive, 2016, 1163 (2016)
5. Halevi, S., Shoup, V.: HELib. <https://github.com/shaih/HELIB>
6. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178 (2009)
7. Dahl, M., Ning, C., Toft, T.: On secure two-party integer division. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 164–178. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32946-3_13
8. Veugen, T.: Encrypted integer division. In: 2010 IEEE International Workshop on Information Forensics and Security (WIFS). IEEE (2010)
9. Veugen, T.: Encrypted integer division and secure comparison. *Int. J. Appl. Cryptogr.* **3**(2), 166–180 (2014)

10. Hall, R., Fienberg, S.E., Nardi, Y.: Secure multiple linear regression based on homomorphic encryption. *J. Off. Stat.* **27**(4), 669 (2011)
11. Nikolaenko, V., Weinsberg, U., Ioannidis, S., Joye, M., Boneh, D., Taft, N.: Privacy-preserving ridge regression on hundreds of millions of records. In: 2013 IEEE Symposium on Security and Privacy (SP), pp. 334–348. IEEE, May 2013
12. Yao, A.C.: Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science, SFCS 2008, pp. 160–164. IEEE, November 1982
13. Yao, A.C.C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science, pp. 162–167. IEEE, October 1986
14. Graepel, T., Lauter, K., Naehrig, M.: ML confidential: machine learning on encrypted data. In: Kwon, T., Lee, M.-K., Kwon, D. (eds.) ICISC 2012. LNCS, vol. 7839, pp. 1–21. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37682-5_1
15. Bost, R., Popa, R.A., Tu, S., Goldwasser, S.: Machine learning classification over encrypted data. In: NDSS, February 2015
16. Lu, W., Kawasaki, S., Sakuma, J.: Using fully homomorphic encryption for statistical analysis of categorical, ordinal and numerical data. IACR Cryptology ePrint Archive 2016, 1163 (2016)



Breaking Text CAPTCHA by Repeated Information

Jae Hyeon Woo¹, Moosung Park², and Kyungho Lee¹(✉)

¹ Graduate School of Information Security, Korea University, Anam-ro, Seongbuk-gu, Seoul, Republic of Korea
{bull10330,kevinlee}@korea.ac.kr

² Agency for Defense Development, Ogeum-ro, Songpa-gu, Seoul, Republic of Korea
parkms@add.re.kr

Abstract. CAPTCHA is a simple challenge-response tool to determine whether the user is a bot or human. The user must answer required text, calculate questions, or choose some images from the provider's choice. *D* portal site, which is one of the most famous web portal site in Korea, asks text response in CAPTCHA image when joining a cafe group, but this CAPTCHA is structured in a very regular format which can be read very simply if used repeatedly. We can read the text characters by bot with very high accuracy through some easy steps, among 2,000 sample CAPTHCAs.

Keywords: CAPTCHA · Portal · Text · Bot · Break · Attack Analysis

1 Introduction

Turing [1] suggested the Turing Test in 1950, which defines the machine's ability of exhibiting human intelligence. Wide spread of internet environment makes the automatic bot acts as a human, doing more speedy actions such as DDoS attack or Camouflage. Humans want to block the auto-deciding machine but the bot wants to pass the test without human, then they want to design a questionnaire system which the machine can't answer easily.

1.1 What Is CAPTCHA?

Lois [2, 24] introduced the concept of CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) in 2000. Early CAPTCHA model (Fig. 1) showed an image including simple texts and the user must type the text message in it. They, who want to pass the questions, need the ability of separating text from background in the image, and distinguishing the exact alphabet or digit character at the general personal level.

CAPTCHA system must be easy for system to make and for human to read, but hard for bot to read. [3, 4] shows how to make good CAPTCHA system.

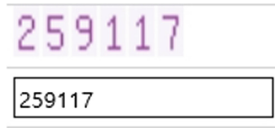


Fig. 1. Simple digit CAPTCHA

1.2 What Kinds of CAPTCHA?

There are so many kinds of CAPTCHA systems. [5–9] suggested an image-based CAPTCHA, and [10] suggested audio-based one. The typical types of them are *text*, *image*, *audio*, *motion*, or *puzzle* types (See Fig. 2), etc.

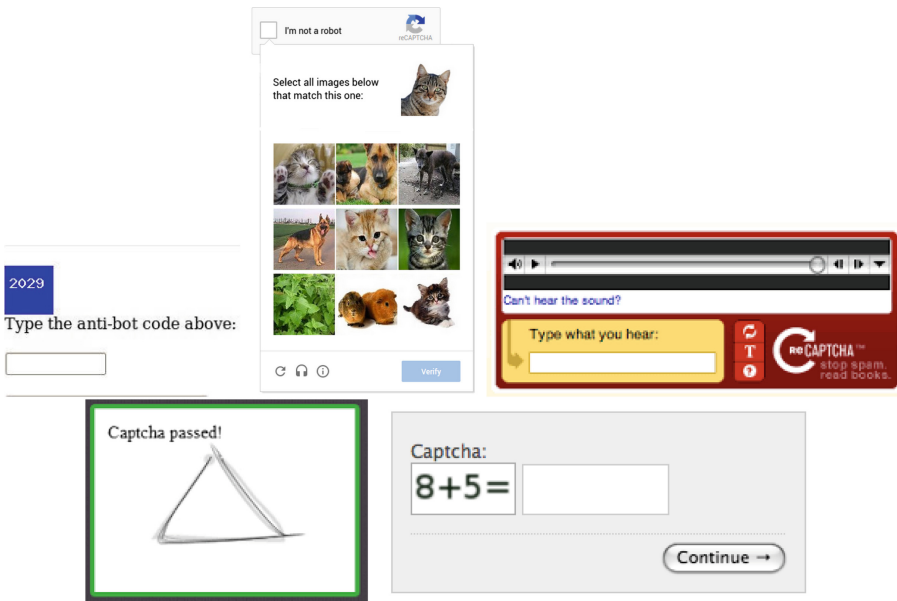


Fig. 2. Different types of CAPTCHA (Searching Google Images)

[11–13] compared the pros and cons between these kinds of CAPTCHA.

1.3 What Kinds of CAPTCHA Analysis?

Two fundamental issues of CAPTCHA are the *usability* and *robustness* [4]. Many servers use the text-based one because it is easy to construct and the answer is clear. But the *usability* is the opposite side of *robustness*. Text-based CAPTCHAs are more studied and attacked [14–18] than other types [19] or [20]. Nowadays, machine learning analysis [21–23] became powerful after the MNIST dataset [25] was provided.

2 D CAPTCHA

Different from the state of the art scheme, we studied CAPTCHA analysis from its basic concept and focused on the *D* site, which is a very famous portal site in South Korea. It uses the text-based CAPTCHA as shown in Fig. 3.



Fig. 3. *D* CAPTCHA (Color figure online)

It has some noises like shadow and curved line to avoid automat analysis, but we attacked this and show that the noises cannot disturb the bot and this CAPTCHA can be attacked almost perfectly.

2.1 Characteristic of *D* CAPTCHA

D CAPTCHA is composed of 5 blue alphabets with white background. No digit is used. Each alphabet has its own gray shadow. More over, there is one curved black line going through the whole characters.

We found that they use only 22 alphabets excluding A, D, O and Q. They might be confused when located at the same image. Fig. 4 shows some sample CAPTCHAs.

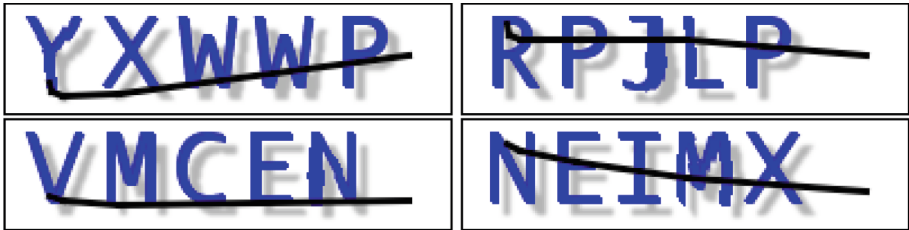


Fig. 4. Some samples

Table 1. Character distribution in 2,000 samples

B	C	E	F	G	H	I	J	K	L	M	N	P	R	S	T	U	V	W	X	Y	Z	Total
433	451	486	451	478	475	471	467	460	426	469	432	464	439	432	450	419	452	460	452	447	486	10,000

Theoretically, there are $22^5 = 5,153,632$ samples regardless of the black curve, but the brute force attack is not necessary in this paper. Table 1 shows the distributions of each characters in 2,000 samples. They are distributed with $m = 454.5$ and $\sigma = 18.6$.

Each CAPTCHA is a 4 channel .png file with the size of (row,col) = (50,200). The 4 channels consist of [R, G, B, Alpha] channels, [R, G, B] channels define the color value and [Alpha] channel defines the degree of clearness. If [R, G, B] = [0,0,0] then the pixel is *black*, [255, 0, 0] is *red*, [0, 255, 0] is *green*, and [0, 0, 255] is *blue*. [255, 255, 255] is *white* and [128, 128, 128] is *gray*. But the Alpha channel can make the black look white if Alpha is small. So [0, 0, 0, 0] is black but seems to be white.



Fig. 5. Original image (Color figure online)

Let’s see Fig. 5. The *blue* pixel consists of [0, 0, 255, 255]. The *black* curve consists of [0, 0, 0, 255]. But the *white* background consists of [0, 0, 0, 0] and *gray* shadow is [0, 0, 0, X] where the X is near 70. The white background and gray shadow depend not on the R,G,B but on Alpha channel. The [R, G, B] of them were all [0, 0, 0], *black*.

2.2 Weak Points of D CAPTCHA and Breaking Process

There are many weak points in D CAPTCHA. Figure 6 shows the channel analysis of each pixel.

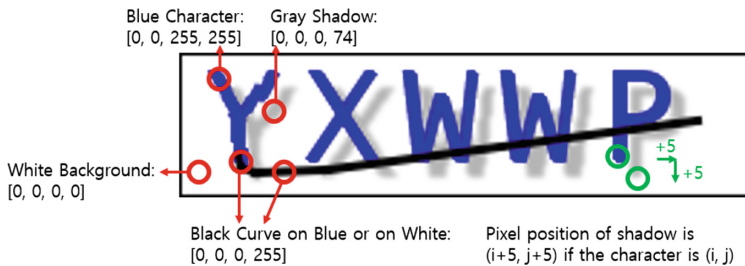


Fig. 6. Channel analysis (Color figure online)

First of all, the color difference is an easy clue to separate the object from background. The blue pixel, which is [0, 0, 255, 255], is exactly the character

pixel. Figure 7 shows the extraction of blue pixels, which has the noise made by the black curve.



Fig. 7. Extraction of blue pixels

Second, the gray shadow is located at the 5 pixel shift to right and down direction from its original blue character position, with Alpha channel blurred. Shifting 5 steps of gray part to left and up direction accords with the blue part if there is no black curve. So we can recover the screened blue pixel from the information of its shadows. Figure 8 shows the recovered image.



Fig. 8. Recovering the screened (or noise) pixel

Third, there is no tilting nor transformation of character and the font size of each character is fixed between some ranges. So we can easily split into five groups. The width of each character may be different (width is around 22), but the height is almost same, 32. Figure 9 shows the separation of each character.



Fig. 9. Split into 5 groups

Although the width of each character is different, the altitude and the height of them are almost fixed. If the location of leftmost top pixel is (0, 0) and the rightmost bottom pixel is (49, 199), the altitude may be between 0 and 49, but the characters are actually located between 6 and 37 of their altitude position (between red lines in Fig. 10).

Then we can split each character like Fig. 11. All of them have the height of 32, but the width is not fixed.

Lastly, we resize the characters into fixed size of 32×25 pixel (Fig. 12).



Fig. 10. Superposition of 2,000 images and boundary of location



Fig. 11. Split each image



Fig. 12. Resize the width of each image

Until now, we introduced how to make the template of each character. The differences between same character in different CAPTCHAs, such as **P** in **YXWWP** and **RPJLP**, are small, then the combination of same characters can make the template image very efficiently in an easy way. We took the median value of regular and frequent characters. There are various shapes of **E** as shown in Fig. 13, but we gathered the leftmost **Es**.

Figure 14 shows the template set. Every split character must be compared with these templates, and the smallest error will decide the answer.

3 Experimental Result

We stored 2,000 CAPTCHA images and each of them has 5 characters, so we compared 10,000 characters in this paper. Only 102 characters have errors among them by comparing with such a simple templates. Table 2 shows the result of error count. 23 of **E** in Table 2 means that 23 **Es** are recognized as other character such as **F**. **Ns** are mostly confused with **M** or **H**, **Ks** are sometimes confused with **X**, **Vs** are done with **Y**, and so on.

3.1 Accuracy

We can get two kinds of accuracy. The first accuracy is that of pass as follows:

$$\frac{2000 - 102}{2000} = 94.9 \tag{1}$$

Each error occurs from different images, then the bot failed 102 times among 2,000 tries. The rate of pass is nearly 95%.



Fig. 13. Various shapes of E



Fig. 14. Template images of 22 characters

Table 2. Error Count

E	H	I	K	M	N	R	V	W	X	Total
23	8	3	12	4	38	6	5	1	2	102

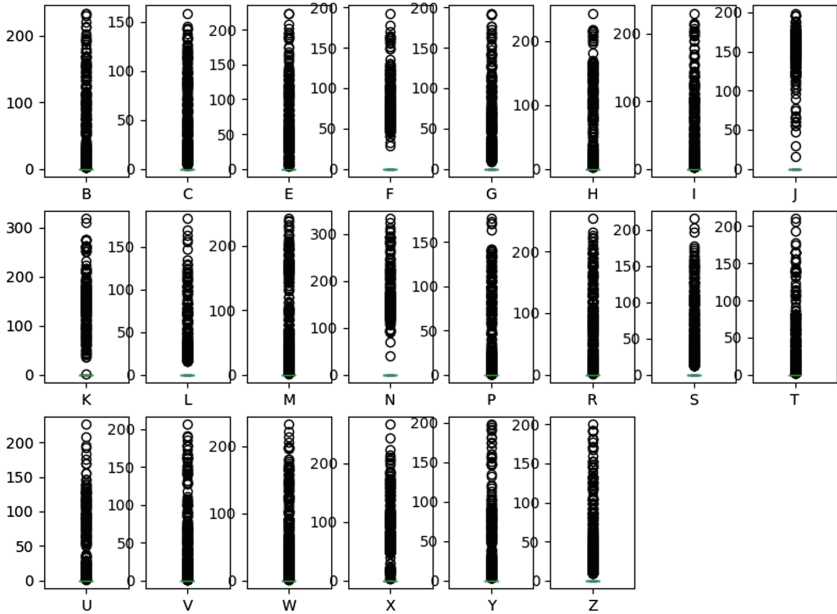


Fig. 15. Box-plot of right matching with template

But the second is the accuracy of readability as follows:

$$\frac{10000 - 102}{10000} = 98.98 \tag{2}$$

The bot can read 9,898 characters exactly among 10,000 characters, and the readability is about 99%.

Figure 15 shows the box plot of matching distances by each characters. The plot of **B** is the distances between all **B**s and the template of it. Every template has $32 \times 25 = 800$ pixels, then the maximum error is 800 when the split of character is totally different from the template. The distance is 0 if the split is exactly same with the template.

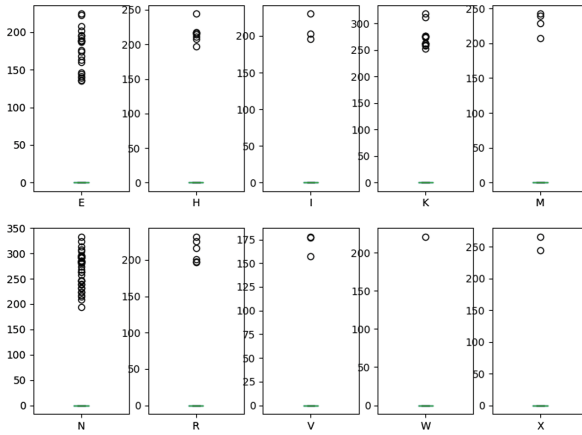


Fig. 16. Box-plot of wrong matching with template

Figure 16 shows the box plot of error cases. The distance between split and the template is large in this cases. It is more than 130 in **E** plot, so the split is recognized as **I**.

4 Conclusion

CAPTCHA is a good tool to distinguish the bot from human, but designing well is not so easy. Any repetitive action or question becomes the clue of solving that system. Same color, same voice, same method of puzzle, same pixel position, same type of noise, etc. All of them give the hint of easy solution. This paper shows that the simple manipulation of repeated CAPTCHA images can read the characters with nearly 99% accuracy because of the repeated color and location of CAPTCHA system, by using same channel sharing method. If we made more accurate templates like two or more **E** templates, we can reduce the distance error and increase the accuracy up to nearly 100%. Easy design gives the easy solution to the partners.

Acknowledgments. This work was supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract (UD060048AD).

References

1. Turing, A.M.: Computing machinery and intelligence. *Mind* **59**(236), 433–460 (1950)
2. von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: CAPTCHA: using hard AI problems for security. In: Biham, E. (ed.) *EUROCRYPT 2003*. LNCS, vol. 2656, pp. 294–311. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_18
3. Chellapilla, K., Larson, K., Simard, P., Czerwinski, M.: Designing human friendly human interaction proofs (HIPs). In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 711–720. ACM (2005)
4. Yan, J., El Ahmad, A.S.: Usability of CAPTCHAs or usability issues in CAPTCHA design. In: *Proceedings of the 4th Symposium on Usable Privacy and Security*, pp. 44–52. ACM (2008)
5. Datta, R., Li, J., Wang, J.Z.: IMAGINATION: a robust image-based CAPTCHA generation system. In: *Proceedings of the 13th Annual ACM International Conference on Multimedia*, pp. 331–334. ACM (2005)
6. Elson, J., Douceur, J.R., Howell, J., Saul, J.: Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In: *ACM Conference on Computer and Communications Security*, vol. 7, pp. 366–374 (2007)
7. Gossweiler, R., Kamvar, M., Baluja, S.: What’s up CAPTCHA?: a CAPTCHA based on image orientation. In: *Proceedings of the 18th International Conference on World Wide Web*, pp. 841–850. ACM (2009)
8. Zhu, B.B., Yan, J., Bao, G., Yang, M., Xu, N.: CAPTCHA as graphical passwords—a new security primitive based on hard AI problems. *IEEE Trans. Inf. Forensics Secur.* **9**(6), 891–904 (2014)
9. Thangavelu, S., Purusothaman, T., Gowrison, G.: Emoji CAPTCHA: a secured picture character approach against OCR Attacks. *Prevent* **7**(1) (2017). <https://ijcsits.org/papers/vol7no12017/1vol7no1.pdf>
10. Gao, H., Liu, H., Yao, D., Liu, X., Aickelin, U.: An audio CAPTCHA to distinguish humans from computers. In: *2010 Third International Symposium on Electronic Commerce and Security (ISECS)*, pp. 265–269. IEEE (2010)
11. Singh, V.P., Pal, P.: Survey of different types of CAPTCHA. *Int. J. Comput. Sci. Inf. Technol.* **5**(2), 2242–2245 (2014)
12. Bhalani, S.D., Mishra, S.: A survey on CAPTCHA technique based on drag and drop mouse action. *Int. J. Tech. Res. Appl.* **3**(2), 188–189 (2015)
13. Abdalla, K., Kaya, M.: An evaluation of different types of CAPTCHA: effectiveness, user-friendliness, and limitations. *Int. J. Sci. Res. Inf. Syst. Eng. (IJSRISE)* **2**(3) (2017). <http://ijsrise.com/index.php/IJSRISE/article/view/51>
14. Mori, G., Malik, J.: Recognizing objects in adversarial clutter: breaking a visual CAPTCHA. In: *Proceedings of the 2003 IEEE Computer Society Conference on IEEE Computer Vision and Pattern Recognition*, vol. 1, p. I (2003)
15. Yan, J., El Ahmad, A.S.: A low-cost attack on a Microsoft CAPTCHA. In: *Proceedings of the 15th ACM Conference on Computer and Communications Security*, pp. 543–554. ACM (2008)
16. Bursztein, E., Martin, M., Mitchell, J.: Text-based CAPTCHA strengths and weaknesses. In: *Proceedings of the 18th ACM Conference on Computer and Communications Security*, pp. 125–138. ACM (2011)
17. Starostenko, O., Cruz-Perez, C., Uceda-Ponga, F., Iarcon-Aquino, V.: Breaking text-based CAPTCHAs with variable word and character orientation. *Pattern Recogn.* **48**(4), 1101–1112 (2015)

18. Kim, J., Kim, S., Kim, H.J.: Breaking character and natural image based CAPTCHA using feature classification. *J. Korea Inst. Inf. Secur. Cryptol.* **25**(5), 1011–1019 (2015)
19. Chew, M., Tygar, J.D.: Image recognition CAPTCHAs. In: Zhang, K., Zheng, Y. (eds.) *ISC 2004*. LNCS, vol. 3225, pp. 268–279. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30144-8_23
20. Tam, J., Simsa, J., Hyde, S., Ahn, L.V.: Breaking audio CAPTCHAs. In: *Advances in Neural Information Processing Systems*, pp. 1625–1632 (2009)
21. Chellapilla, K., Simard, P.Y.: Using machine learning to break visual human interaction proofs (HIPs). In: *Advances in Neural Information Processing Systems*, pp. 265–272 (2005)
22. Golle, P.: Machine learning attacks against the Asirra CAPTCHA. In: *Proceedings of the 15th ACM Conference on Computer and Communications Security*, pp. 535–542. ACM (2008)
23. Wang, Y., Huang, Y., Zheng, W., Zhou, Z., Liu, D., Lu, M.: Combining convolutional neural network and self-adaptive algorithm to defeat synthetic multi-digit text-based CAPTCHA. In: *2017 IEEE International Conference on IEEE Industrial Technology (ICIT)*, pp. 980–985 (2017)
24. CAPTCHA. <http://www.captcha.net/>
25. MNIST. <http://yann.lecun.com/exdb/mnist/>



Automatic Mitigation of Kernel Rootkits in Cloud Environments

Jonathan Grimm¹, Irfan Ahmed¹(✉), Vassil Roussev¹, Manish Bhatt¹,
and ManPyo Hong²

¹ Department of Computer Science, University of New Orleans, Lakefront Campus,
2000 Lakeshore Dr., New Orleans, LA 70122, USA

jlgrimm1@uno.edu, {irfan,vassil}@cs.uno.edu, mbhatt@my.uno.edu

² Graduate School of Information and Communication,
Ajou University, Suwon, South Korea
mphong@ajou.ac.kr

Abstract. In cloud environments, the typical response to a malware attack is to snapshot and shutdown the virtual machine (VM), and revert it to a prior state. This approach often leads to service disruption and loss of availability, which can have much more damaging consequences than the original attack. Critical evidence needed to understand and permanently remedy the original vulnerability may also be lost. In this work, we propose an alternative solution, which seeks to automatically identify and disable rootkit malware by restoring normal system control flows. Our approach employs virtual machine introspection (VMI), which allows a privileged VM to view and manipulate the physical memory of other VMs with the aid of the hypervisor. This opens up the opportunity to identify common attacks on the integrity of kernel data structures and code, and to restore them to their original state.

To produce an *automated* solution, we monitor a pool of VMs running the same kernel version to identify kernel invariants, and deviations from them, and use the observed invariants to restore the normal state of the kernel. In the process, we automatically handle address space layout randomization, and are able to protect critical kernel data structures and all kernel code. We evaluate a proof-of-concept prototype of the proposed system, called *Nixer*, against real-world malware samples in different scenarios. The results show that changes caused by the rootkits are properly identified and patched at runtime, and that the malware functionality has been disabled. We were able to repair kernel memory in all scenarios considered with no impairment of the functionality and minimal performance impact on the infected VMs.

Keywords: Virtual machine introspection · Malware · Virtualization

This work was supported in part by the NSF grant # 1623276.

1 Introduction

Kernel rootkits compromise the OS kernel to maintain unrestricted access to system resources including physical memory and disk. They are used by attackers to hide the footprints of their malicious activities on a compromised system, such as files/folders containing malware executables on disk, or a backdoor process running in the physical memory to provide unauthorized remote access.

Kernel rootkits use two common techniques for infection: *direct kernel object manipulation* (DKOM), and *kernel object hooking* (KOH). DKOM rootkits subvert the kernel by directly modifying data objects. For instance, the *FU* rootkit [18] manipulates doubly linked-list of EPROCESS data structure in Microsoft (MS) Windows to hide processes. It modifies the data pointers of an EPROCESS node representing the process to be hidden to delink it from the process list. KOH-based rootkits hijack the kernel control flow by either modifying function pointers in kernel objects, or overwriting existing fragment of code with malicious code. For instance, the *basic_6* rootkit [1] changes a function pointer in the system call table to redirect it to a malicious code that hides files and directories. The *suterusu* rootkit [7] modifies the prologues of a target function in the kernel code in memory with malicious routines, and manipulates CPU registers to disable the write-protection of kernel code.

The primary focus of this work is KOH attacks, and rootkits that hijack system control flow. The major focus of existing solutions is to ensure the integrity of system control flow such as CFIGuard [22], KCoFI [16], and kBouncer [3]. Unfortunately, they are not accurate, and may fail to prevent an attack on system control flow [15]. In any case, their solution is incomplete—if a compromised system is identified on network, they lack the ability to surgically restore the known good state of the kernel. Therefore, the typical defensive response is to remove the system from service, and initiate a full recovery process. This often induces a period of low, or zero, availability, which is an increasingly unacceptable situation.

In this paper, we propose *Nixer* - a first responder to mitigate an ongoing rootkit attack on the system control flow while ensuring the continuity of essential operations of the system under attack in order to gain critical time for a complete defensive response. *Nixer* is specifically designed for a cloud computing environment. It runs at a hypervisor (higher privileged) level and operates outside the address space controlled by a rootkit thereby, providing leverage against the rootkit.

To detect a rootkit's malicious modifications/infections, *Nixer* utilizes VMI to access the pool of VMs running same OS kernel in a cloud environment, compares code and invariant data structures within a pool to obtain a baseline, and identifies any discrepancies in a VM pointing to malicious modifications. The kernel code (including modules) and invariant data structures (such as the interrupt and system call tables) do not change after they are setup in memory and are often targeted by rootkit for persistent modifications in system control flow [18]. The baseline is used to identify the original content, which are then replaced with modified (malicious) content in an infected VM. The latter step

recovers the normal system control flow and disrupts the execution of malicious code injected by rootkit, apparently disable or halt the rootkit without compromising service availability, user data and forensic evidence.

The implementation of Nixer is challenging due to address space layout randomization (ASLR). Kernel code (and modules) load into different memory locations that change the values of same function pointers across VMs within a pool. Also, the kernel code contains relocatable code having absolute addresses that make the code different across VMs, and therefore, their cryptographic hash values do not match. To solve this problem, we employ a cross-comparison based de-randomization technique to compare kernel code and function pointers (in kernel data structures) effectively when ASLR is enabled in the VMs.

The rest of the sections are organized as follows: Sect. 2 describes the related work. Section 3 presents an overview of the proposed approach, and challenges and solutions, followed by implementation details in Sect. 4. Section 5 presents the evaluation results. Section 6 discusses implementation decisions and the limitations followed by a conclusion in Sect. 6.

2 Related Work

There are many OS constrained attempts to preserve control flow including CFI-Guard [22], KCoFI [16], and kBouncer [3]. These techniques have some advantages, no semantic gap and more fine grained activity monitoring, but they all share a common weakness. They are all potentially vulnerable to the same malware they are attempting to prevent. This sort of attack has been demonstrated for PatchGuard [8]. With this in mind we have focused on VMI based solutions, because they share common benefits and challenges with our approach.

ModChecker [12] and IDTchecker [11] are VMI based solutions to check the integrity of kernel modules and interrupt descriptor table. ModChecker performs one-to-one comparison of a kernel module across multiple VMs, and is able to detect code modification attacks such inline hooking, and DLL hooking. IDTchecker is similar in approach but focuses on checking the integrity of IDT. It uses pre-defined rules depicting the normal structure of the table to detect any unusual modifications. ModChecker and IDTchecker are different from Nixer in that they can only monitor the modules and IDT for any modifications. Nixer on the other hand, is a proactive solution that changes the state of a VM to mitigate a rootkit attack. Nixer also has wider coverage of physical memory that includes system call table.

HookLocator [10] is a VMI based solution for kernel pool scanning. It locates function pointers in the kernel pools and reports changes to those function pointers. It obtains function pointers in a learning phase requiring two VMs or snapshots with the kernel located at different locations. This gives HookLocator a list of function pointer values to scan the kernel pools. It then monitors instances of these values which were shown not to change during their lifetimes during the learning stage, and reports changes to them. Nixer has no learning stage, it provides its coverage immediately using a constantly generated baseline from

the VMs that it is guarding, but the areas of coverage of Nixer and HookLocator do not overlap at all, so different approaches are expected.

Livewire [17] is an intrusion detection system based on VMI. It uses policy modules to detect malicious activity in monitored VMs. It has a variety of options including signature scanning, integrity scanning of executables, monitoring of statistics misreporting. It also monitors attempts to use suspicious functions like changing memory protections or using raw sockets. Livewire is able to disrupt attacks based on signatures or activities including IDT and SSDT changes, but it communicates with the OS to do so. It also suspends the VM when scanning is required. Nixer is able to stop IDT and SSDT attacks without pausing the VMs while scanning and does not have any presence inside the VM.

VMWatcher [20] and libGuestfs [4] allow running antivirus tools outside the VM to scan the disk of a running VM. VMWatcher also demonstrates malware detection based on differences between in VM and out of VM views of files and processes. Nixer focuses on preserving the in VM functionality for security tools rather than enabling them to run outside the VM.

Win et al. [21] apply machine learning techniques specifically Support Vector Machines to identify malware and rootkits in Linux virtual machines. Their approach utilize an in-VM monitor to capture events. The monitor is installed through VMI. This is a completely different approach from Nixer that focuses on monitoring memory invariants and has no component running inside a VM.

3 Mitigation of Kernel Rootkits

3.1 Problem Statement and Assumptions

Given a pool of virtual machines in a cloud environment, our goal is to recover the system control flow of a compromised VM (hijacked by a rootkit) without affecting the availability of services, losing user data, and destroying forensic evidence of rootkit presence in memory. To achieve our goal, we make the following assumptions about a cloud environment we operate in:

- *All the VMs in a pool run the same kernel configuration, including additional drivers/modules.* This assumption is realistic because the VMs are usually generated from a reference VM to simplify maintenance processes.
- *Semantics of kernel data structures are consistent across VMs.* Since the VMs run same code, this assumption must hold true. (Semantics Attacks such as *direct kernel structure manipulation* (DKSM) [13] are out of scope of this work.)
- *Pausing VMs for a brief time period is acceptable for recovery.* If the pause is short enough not to disrupt ongoing network connections, it is indistinguishable from routine network-induced service stalls.

3.2 Overview of Nixer

Hardware virtualization is the critical mechanism by which resources and workloads are managed in cloud environments. It allows full-stack installations

(including an OS kernel) to be built, cloned, distributed, executed in isolated environments, paused, resumed, and discarded. In effect, the virtual machine manager (VMM), or hypervisor, is the new control and resource management layer, which provides individual VMs with CPU, RAM, storage, and network resources. The VMM also provides virtual machine introspection (VMI), which allows a privileged VM to examine and modify the current state of a guest VM. Our proposed solution, Nixer, is designed to take advantage of this capability to enable the automated recovery from an ongoing rootkit attack.

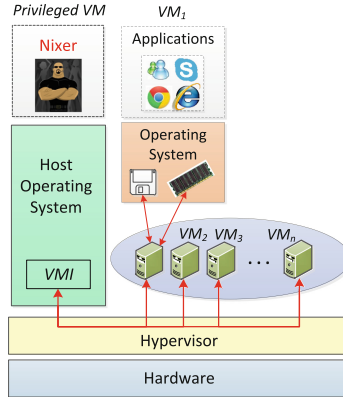


Fig. 1. Overview of Nixer

Figure 1 presents an overview of Nixer. Nixer’s operation consists of three distinct phases: baselining, anomaly detection, and control flow recovery. In baselining, the tool utilizes a pool of VMs running the same kernel configuration to obtain a baseline of normal in-memory content of VMs within the pool. In particular, it seeks to detect code and invariant data structures as they are the most common targets of rootkits to intercept the system control flow. The tool uses a majority-wins approach and considers the VMs that have the same content in physical memory, and are in the majority to create the baseline, if the content appear to be different across VMs. This approach allows Nixer to quickly develop baseline on the fly without the need for offline pre-processing, and the creation of cryptographic signatures for known-good content. This dynamic adaptation is particularly valuable when VMs are run at scale and are patched regularly.

During anomaly detection, Nixer compares the established baseline with the content of VMs in the pool to detect any differences. Since it considers only invariants, the content must be same unless the VM is compromised by a rootkit and is currently under attack. Rootkits target invariants for persistent change in system control flow. During control flow recovery, the system identifies the original content of the compromised VM, and replaces the modified (malicious) content with it. It is worth mentioning that the contents may vary across VMs

within a pool due to address space layout randomization (ASLR). However, if the contents are de-randomized, they must appear same.

3.3 Challenges and Solutions

Recall that the idea behind ASLR is to assign random base addresses on each execution. This means that, in different VM instances, all the absolute addresses in the kernel code and data structures will be different. In particular, function pointers will be different, which makes direct comparison (for the purpose of baselining) meaningless. Therefore, *Nixer* performs some pre-processing that allows the normalization of absolute addresses across VMs.

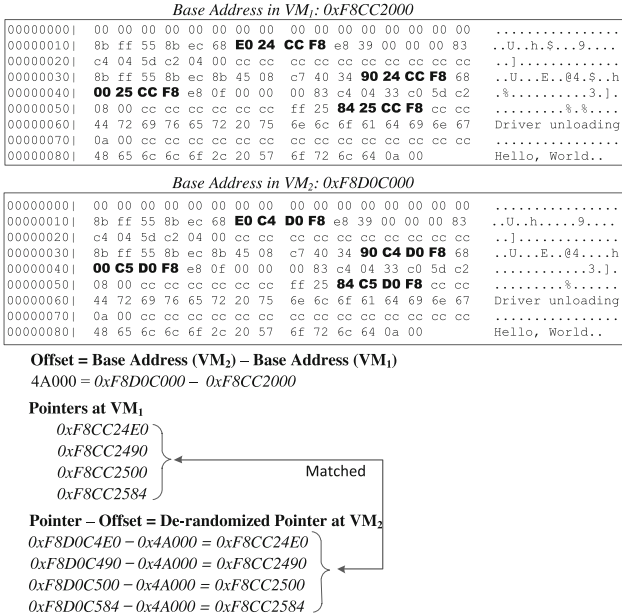


Fig. 2. De-randomization of the pointers in a Kernel Module. The memory snapshots of the module are taken from the two virtual machines VM_1 and VM_2 .

De-randomization of the Kernel. Figure 2 presents the impact of ASLR on two virtual machines VM_1 and VM_2 . Both VMs have a same kernel module in their physical memory but due to ASLR, they are loaded into two different memory locations. The base address for VM_1 and VM_2 are 0xF8CC2000 and 0xF8D0C000 respectively. Consequently, the four absolute addresses in the module are different, making it difficult to compare the whole code effectively.

We develop a method to de-randomize the module to a single base-address α . The method first computes the offset between α and the base address of the module, and then subtracts the offset from the absolute pointer address to de-randomize the pointer value. For example, in Fig. 2, α is the base address of

VM_1 's module, and we want to de-randomize the VM_2 's module to α . Let say, β is the base address of VM_2 's module. The offset can be computed as $offset = \beta - \alpha$, where $\alpha < \beta$. Furthermore, given n number of absolute addresses in the module, $\gamma(i)$ represents an address value. The de-randomized address $\theta(i)$ for $\gamma(i)$ can be computed as

$$\forall 0 < i < n, \theta(i) = \gamma(i) - offset \quad (1)$$

Computing Original Values. When Nixer identifies any discrepancies in a VM, it further figures out the original contents of the VM to replace the modified (malicious) contents with them. Unfortunately, the original contents are lost because they are overwritten during a rootkit attack. To recover the code, if the code does not contain any absolute addresses, then Nixer obtains it from the baseline as is and patches the VM to recover from the infection.

On the other hand, to recover the data structures and the code containing absolute addresses, Nixer cannot obtain the original contents from the baseline because of the ASLR impact. To solve this issue, Nixer computes the original pointer values and put them back to their correct location. For example, the missing value is θ , and the base address of kernel module in compromised and benign VMs are α and β . The value of same pointer in a benign VM is γ . The missing value θ is computed as

$$\theta = \gamma - (\beta - \alpha), \text{ where } \beta > \alpha \quad (2)$$

Furthermore, some data structures may contain pointers to different kernel modules. Nixer treats each pointer independently as θ . It obtains the address range of each kernel module and maps the pointer value (in consideration) to its correct address range in order to find its correct kernel module. The rest of the process is same as described above to obtain θ value.

Reducing the Semantic Gap. Nixer gets a raw access to the physical memory of a VM. That is, it only sees ones and zeroes without any semantic information about them. Since it runs outside the VM, it does not have operating system support running inside the VM to determine the semantics. The existing solutions [19] that reduce the semantic gap automatically are complex. For instance, they require modifications in hypervisor, and the support of an additional VM. Since the focus of this paper is not on solving the semantic gap problem, Nixer reduces the semantic gap manually - a common approach used by many existing VMI solutions [14] and is sufficient for us to implement a proof of concept tool. This approach generally requires the knowledge of operating system internals and some reverse engineering to figure out traversal trees of data structures to reach the content being monitored and recovered from any rootkit infection.

4 Implementation

Nixer is implemented in C++ and has almost 12,400 lines of code. It utilizes libVMI [5] for VMI capabilities and opdis [6] for disassembly. LibVMI is a wrapper

library that provides a generic interface to the VMI capabilities of Xen, QEMU, KVM and allows that same interface to be used on both physical memory of a live VM and memory dumps.

Nixer is a proof-of concept tool to mitigate rootkit attacks on Windows. In particular, it covers kernel code (and modules) and two well-known kernel data structures i.e. *Interrupt Descriptor Table* (IDT) and *System Service Descriptor Table* (SSDT) or system call table. Nixer parses the memory and extracts the kernel code and data structures. For the code, it finds the base addresses of kernel code and each kernel module by traversing the doubly-linked list of modules. Each node in the list is represented by the data structure `LDR_DATA_TABLE_ENTRY` containing the name and base address of a module. The pointer to the list is obtained from a system variable `PsLoadedModuleList`, already populated by lib-VMI. Nixer further parses kernel code and each module into Portable Executable (PE) format to extract headers and sections of code and data.

For the kernel data structures, the pointer to IDT is obtained from `IDTR` register and `KeServiceDescriptorTable` system variable points to SSDT. In each IDT entry, the pointer value consists of the combination of two fields i.e. `Offset` and `ExtendedOffset`. In SSDT table, each entry has `ServiceTableBase` field that points to a system call table – an array of pointers to system calls.

Furthermore, Nixer generates a list of patches to apply to each VM within a pool. The VM is paused for these patches to be applied as we are borrowing pages from the guest VMs as described above, and Xen will not allow that without pausing the VM. This is why, we apply all the patches at the end of the scanning process to minimizing the cost of pausing and resuming VMs.

5 Evaluation

We conducted several experiments using real world malware (containing rootkit functionalities) to evaluate the mitigation of malware behavior, and the impact of mitigation on the services (such as web server and antivirus) running on a compromised VM. Our evaluation results demonstrated improved behavior of in-VM security and other services. For instance, after identifying and disabling the rootkit component of a piece of malware, the antivirus service already installed on the system was able to quickly identify the (previously hidden) infected files. This section discusses the experimental setup and the details of the experiments.

5.1 Experimental Setup

All experiments are conducted on a workstation with an Intel i7-4770 @ 3.70 GHz and 16 GB of RAM with Fedora 22 and Xen 4.5.3 hypervisor installed. Virtual machines are created with 512 MB of RAM and 50 GB of disk space on a LVM volume. We run Windows XP SP2 operating system on the VMs because the available sample rootkits are known to work on XP systems.

Clones are created using LVM's copy-on-write snapshots with 20GB allocated to the clone's writes. The physical machine is not under CPU or memory pressure

during the experiments. We did not run any unnecessary applications/services on host machine.

We use real world malware for experiments, and Volatility [9], a memory forensic framework to obtain independent ground truth and verify our claims. The memory dumps are taken with libvmi's dump-memory utility. We use WinXPSP2x86 profile in Volatility with several plugins including idt, ssdt, malfind.

We perform the following procedure to ensure that each experiment is performed in a fresh setup that does not contain any remnants of previous experiments. It starts with taking a snapshot of the booted VM before experiment, and then make modifications in the VM such as by running malware, debugger or any other tool, followed by another snapshot of the VM to verify changes. Nixer is then used to revert the changes, after which, a snapshot of the VM is taken again to verify that Nixer has mitigated the malicious behavior. Volatility is used for the verification process.

5.2 Accuracy Evaluation

Initial experiments aim to verify that Nixer does not damage the functionality of running VMs or make unwarranted changes to memory. We start with one shutdown Windows XP SP2 VM and generate 2 clones. Both clones are booted. No malware is used for this experiment because some malware creates instability, and we want to verify the stability of VMs guarded by Nixer without confounding factors. We test the Nixer on two complementary scenarios: (1) when all the VMs in the pool are working normally without going through any rootkit modifications. This scenario verifies whether Nixer make unwanted changes in memory. (2) When the prelude of a kernel routine is synthetically replaced with NOPs by another of our VMI tools to observe whether Nixer can revert the changes accurately.

Scenario 1 - No Modifications. We use two VMs for this experiment. One is used as reference for Nixer and other as a target VM to evaluate any modifications by Nixer. Both the VMs are logged in after boot. The reference VM is left untouched. The target VM is left idle for 5 min to provide it sufficient time to settle down. We take the memory snapshots of the target VM periodically and compare one version with it immediate previous version using BinDiff [2]. Some memory changes are observed in the snapshots as expected due to continued operation but they are not significant, since the VM is idle. We run Nixer to introspect the VM with the reference VM. Nixer reported making no change as expected since no malicious modifications are made in the VM. To verify Nixer's claim, we also take a snapshot and compare it with the last snapshot of the VM using BinDiff. Apparently, no change is identified. The experiment is repeated several times and also with other VMs that concludes that Nixer does not patch VM unexpectedly.

Scenario 2 - Modifications in a Kernel Routine. We constructed a custom (malicious) program using our VMI capabilities to replace MS Windows function prelude with NOPs for the first function in a targeted PE file in memory.

The experiment starts with taking a memory snapshot when the VM is paused. After the snapshot, We unpause the VM and run our custom (attack) tool, and then take another memory snapshot and confirmed the changes using BinDiff. We run Nixer this time to revert the changes, and then take another memory snapshot to confirm that the memory contents are matched with the original. This demonstrates that Nixer can effectively identify the (malicious) modifications in Kernel code and revert them to original state. Nixer uses the reference VM from last scenario to identify the modifications and original contents.

The experiment is repeated several times with the other VMs with or without pausing the target VM during scanning. The experimental results conclude that the code changes have been repaired by Nixer successfully.

5.3 Experiments with Real-World Rootkits

We evaluate Nixer functionality with four malware samples (refer to Table 1) that specifically target IDT and SSDT.

For IDT hooking, we run Strace Fuzen malware in a freshly cloned VM. Volatility confirmed IDT entry had changed from 0x8053C651 pointing to \$KiSystemService as expected to 0xF8A182A0. After running Nixer to fix the modifications, Volatility confirmed that IDT entry 0x2E was changed back to the original value.

Table 1. Evaluation results of Nixer on real-world malware samples.

	STrace Fuzen	Basic_6	F.gen!Eldorado	Backdoor X.AHUO
IDT/SSDT	IDT	SSDT	SSDT	SSDT
Index entry	0x2E	0x91 & 0xAD	0x42	0x42
Original pointer	0x8053C651	0x8056F266 & 0x80608852	0x8056E634	0x8056E634
Pointer target	\$KiSystemService	ntsokrnl.exe	ntsokrnl.exe	ntsokrnl.exe
Infected pointer	0xF8A182A0	0xF7BA53D0 & 0xF7BDB3D0	0xF7B503D0	0xF7BD43D0
Pointer target	Unknown	quadw.sys & sraslk.sys	objnts.sys	quasfd.sys
Patched by Nixer	✓	✓	✓	✓

For SSDT hooking, we used three malware samples, viz. Basic_6 (from [rookit.com](https://github.com/r00kit)) mirror on github, PcClient.F.gen!Eldorado and BackdoorX.AHUO (from openmalware.org). Volatility confirmed that these malware samples made changes in the SSDT entries. For Basic_6, Volatility

confirmed that SSDT entries for `NTQueryDirectoryFile` (0x91) and `NTQuerySystemInformation` (0xAD) had changed. After Nixer was run to revert the changes, `Volatility` confirmed that SSDT entries were changed back to the original values. For `F.gen!Eldorado`, `Volatility` confirmed that the SSDT entries for `NtDeviceIoControlFile` owned by `ntsokrnl.exe` was changed to `objnts.sys`. After running Nixer to revert the changes, `Volatility` verified that the changes made by `F.gen!Eldorado` were reverted. For `BackdoorX.AHUO`, `Volatility` confirmed that the SSDT entries for `NtDeviceIoControlFile` owned by `ntsokrnl.exe` had changed to `quasfd.sys`. Moreover, after Nixer was run to revert the changes, `Volatility` confirmed that the changes had been successfully reverted to their original values.

Finally, to verify that Nixer recovers the system control flow, we further experimented with `Basic.6` as its source code was available to us to determine `Basic.6`'s functionality. `Basic.6` was run in our VM using OSR loader. `Basic.6` hides files with the name `"_root_"`. We created files with these names and ran `Basic.6` with OSR loader as before. The files disappeared from *Windows Explorer*, `dir` command run from `cmd.exe` and other programs as expected. After Nixer was run, these utilities and program can find these files again, despite OSR loader still reporting that the `Basic.6` kernel module was loaded.

5.4 Improving the Effectiveness of In-VM Programs

We have further tested the `Basic.6` rootkit functionality and the impact of Nixer in two different scenarios: (1) a web server running on a VM when the rootkit hides some web resources and make them unavailable for users. (2) an antivirus running on a VM when the rootkit hides some malware files making the antivirus ineffective to detect them.

Scenario 1 - Web Server. We install the `nginx` web server in the target VM and create few pages for it to serve including an `index.html` and a `_root_.htm`. These files are accessible remotely from a web browser. When we run the rootkit, it hides the web server files, making the web server unable to serve them to web browser request. Now the browser gets an error message, instead of the page. We use Nixer to detect and revert the changes in SSDT, apparently unhide the web server files. We observe that the web server is now able to serve the files. The rootkit, however is still running as kernel module, but its functionality (of hiding files and folders) is disabled.

Furthermore, we also observed the typical speed of serving the web pages, which is 5 ms in our setup and it remains 5 ms while the VM was being scanned by Nixer. Even while `Basic.6` is being removed, the performance is 25 ms for one request 18ms for the next then returned to the normal 5 ms. These empirical results show that Nixer has successfully recovered the infected VM, and enabled the web server to serve more pages.

Scenario 2 - Antivirus. We install *ClamAV* in the target VM, and put `FU` rootkit in a folder named `_root_` that is protected (hidden) by `Basic.6`. We run the rootkit on the VM and then, start a scan with `ClamAV`, which apparently

is not able to detect FU rootkit. We use Nixer to revert the rootkit modifications, apparently making the hidden rootkit files available to ClamAV. Now the antivirus is able to detect FU rootkit. The Basic_6 rootkit is still running in the system. However, it is unable to hide files and folders anymore.

5.5 Performance Evaluation

We executed pairwise comparison in Nixer 100 times to obtain an understanding of how it would perform in a continuous scanning environment. The average time for a pairwise comparison was 60.2 ms with 26.9 ms on average being spent in Nixer’s user code and 23.9 ms system time with virtual machines idle. The CPU used by Nixer was 98% utilized during execution. The test was repeated with the VMs loaded to 100% with prime95. The average execution time for the comparisons was 60 ms with 28.2 ms on average being spent in Nixer’s user code and 25.1 ms system time with virtual machines loaded. The core used by Nixer was 98% loaded during execution. The execution time was remarkably similar regardless of the state of the targeted VMs. The maximum memory used by Nixer in these 200 runs was 20.1 megabytes of memory. The CPU usage is high, but Nixer is a normal process running in a protected VM. Nixer’s VM or process could be throttled to allow for whatever other needs the system has, but it is obvious Nixer is extremely CPU bound. This is a testament to libVMI and Xen’s introspection capabilities that Nixer spend very little time waiting.

6 Conclusions and Future Work

Nixer mitigates a rootkit infection by restoring system to normal control flow. It acts as a first responder and enables in-VM security programs to work effectively in the face of a rootkit attack. Nixer has a small performance penalty and does not disrupt the availability of essential services.

As part of future work, we will improve the memory coverage of Nixer, utilize pdb files from WinDbg for accessing a greater number of OS data structures, and extend it to other hypervisors (qemu, kvm) and other guest operating systems. For performance optimization, we will parallelize this code to scan more VMs at once or different structures simultaneously or add event support so we only scan what is needed when it is changed.

References

1. Basic_6 rootkit (2016). https://github.com/bowlofstew/rootkit.com/tree/master/hoglund/basic_6
2. BinDiff (2016). <https://www.zynamics.com/bindiff.html/>
3. kBouncer (2016). <http://www.cs.columbia.edu/~vpappas/papers/kbouncer.pdf>
4. Libguestfs (2016). <http://libguestfs.org/>
5. LibVMI (2016). <http://libvmi.com>
6. Opdis (2016). <http://mkfs.github.io/content/opdis/>

7. Suterusu rootkit (2016). <https://github.com/mncoppola/suterusu>
8. Understanding and Defeating Windows 8.1 Kernel Patch Protection (2016). http://www.nosuchcon.org/talks/2014/D2.01_Andrea_Allievi_Win8.1_Patch_protctions.pdf
9. Volatility (2016). <http://www.volatilityfoundation.org/>
10. Ahmed, I., Richard, G.G., Zoranic, A., Roussev, V.: Integrity checking of function pointers in kernel pools via virtual machine introspection. In: Desmedt, Y. (ed.) ISC 2013. LNCS, vol. 7807, pp. 3–19. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27659-5_1
11. Ahmed, I., Zoranic, A., Javaid, S., Richard, G., Roussev, V.: Rule-based integrity checking of interrupt descriptor tables in cloud environments. In: Peterson, G., Sheno, S. (eds.) DigitalForensics 2013. IAICT, vol. 410, pp. 305–328. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41148-9_21
12. Ahmed, I., Zoranic, A., Javaid, S., Richard III, G.G.: ModChecker: kernel module integrity checking in the cloud environment. In: 2012 41st International Conference on Parallel Processing Workshops, pp. 306–313. IEEE (2012)
13. Bahram, S., Jiang, X., Wang, Z., Grace, M., Li, J., Srinivasan, D., Rhee, J., Xu, D.: DKSM: subverting virtual machine introspection for fun and profit. In: Proceedings of the 29th IEEE Symposium on Reliable Distributed Systems (2010)
14. Bauman, E., Ayoade, G., Lin, Z.: A survey on hypervisor-based monitoring: approaches, applications, and evolutions. *ACM Comput. Surv. (CSUR)* **48**(1), 10 (2015)
15. Burow, N., Carr, S.A., Brunthaler, S., Payer, M., Nash, J., Larsen, P., Franz, M.: Control-flow integrity: precision, security, and performance. *ACM Comput. Surv.* **50**, 16 (2016)
16. Criswell, J., Dautenhahn, N., Adve, V.: KCoFI: complete control-flow integrity for commodity operating system kernels. In: Proceedings of the IEEE Symposium on Security and Privacy (2014)
17. Garfinkel, T., Rosenblum, M., et al.: A virtual machine introspection based architecture for intrusion detection. In: NDSS, vol. 3, pp. 191–206 (2003)
18. Hoglund, G., Butler, J.: *Rootkits: Subverting the Windows Kernel*. Addison-Wesley Professional, Boston (2006)
19. Jain, B., Baig, M.B., Zhang, D., Porter, D.E., Sion, R.: SoK: introspections on trust and the semantic gap. In: 2014 IEEE Symposium on Security and Privacy, pp. 605–620. IEEE (2014)
20. Jiang, X., Wang, X., Xu, D.: Stealthy malware detection through VMM-based out-of-the-box semantic view reconstruction. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 128–138. ACM (2007)
21. Win, T.Y., Tianfield, H., Mair, Q.: Detection of malware and kernel-level rootkits in cloud computing environments. In: 2nd IEEE International Conference on Cyber Security and Cloud Computing (CSCloud), pp. 295–300 (2015)
22. Yuan, P., Zeng, Q., Ding, X.: Hardware-assisted fine-grained code-reuse attack detection. In: Bos, H., Monrose, F., Blanc, G. (eds.) RAID 2015. LNCS, vol. 9404, pp. 66–85. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26362-5_4



Glitch Recall: A Hardware Trojan Exploiting Natural Glitches in Logic Circuits

Jungwoo Joh¹, Yezee Seo¹, Hoon-Kyu Kim², and Taekyoung Kwon¹(✉)

¹ Information Security Laboratory, Yonsei University, Seoul 03722, Korea
{joh.jungwoo, seoyz0716, taekyoung}@yonsei.ac.kr

² Agency for Defense Development, Seoul, Korea
hunk@add.re.kr

Abstract. As the IoT era comes to the full-fledged, hardware Trojans that involve malicious modifications of circuitry are becoming a growing security concern. To avoid a detection mechanism, hardware Trojans may need a stealthy nature in their existence for being dormant, and even when triggered. In this paper, we devise a new hardware Trojan concept that exploits natural glitches and their control mechanisms for information leakage in a stealthy manner. We indeed reversely exploit the glitch control mechanisms to be bypassed when triggered, and try to recall the natural glitches for the purpose. An adversary who triggered the hardware Trojan may infer multiple input values from a single output of the target logic, thereby obtaining multiple outputs of the preceding logics, by monitoring the existence of the natural glitches. We perform experiments and discuss the results and threats, not to be neglected, along with a possible mitigation.

Keywords: Hardware Trojan · Glitch · Hazard · Glitch control
Multiplexer · Programmable delay element

1 Introduction

Malware standing for malicious software has increased dramatically and placed great concerns about security in computing systems for the last decades, e.g., including but not limited to computer viruses, Internet worms, Trojan horses, spyware, and ransomware. Many researchers, accordingly, have developed various kinds of detection mechanisms and mitigation techniques against such malware, but mostly under the strong assumption that hardware, which is underneath software, is rather reliable and trustworthy [1]. Unfortunately, such an assumption is no longer valid at least because of the rise of hardware Trojan horses, or hardware Trojans in short. A typical form of hardware Trojans consists of a *trigger* and a *payload*, where the former implies activation mechanisms and the latter

This work was supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract (UD160066BD).

refers to the part of the circuit or the functionality affected by the activation of hardware Trojans [5]. Trigger conditions vary, e.g., specific node values for combinational logic trigger, specific input sequences for sequential logic trigger, and specific sensor signals or voltage levels for analog trigger [4]. Malicious effects, saying payloads, of hardware Trojans also vary, e.g., abnormal functional behaviors, strange memory contents, information leakage, and denials of service [5]. It is worriedly possible to alter and exploit parts of hardware, e.g., the circuitry of an integrated circuit in the design and manufacturing/fabrication processes that involve outsourcing and third-parties, for hardware Trojans trying to leak sensitive information or disable functional components when triggered [3]. Besides, it is very difficult to detect such a small and stealthy alteration in the circuitry [2].

A *glitch*, which means a brief irregularity or a fast spike in signals, is an unwanted natural phenomenon occurring in digital logic circuits because it may cause power dissipation and data corruption in the middle of circuits [8, 13]. Natural glitches, which imply ones occurring naturally from an inherent nature of digital logic circuits, are frequently created because digital gates have multiple inputs arriving at different instances of time through multiple paths [16]. Thus, indeed, various glitch control techniques have been developed and deployed to reduce and if possible, eliminate the natural glitches, for instance, by inserting negative edge triggered flip-flops or programmable delay logics [10–15]. Interestingly, some hardware Trojans artificially generate glitches in logic circuits by manipulating one or more operating clocks, and exploit the artificial glitches for information leakage, for instance, a secret cryptographic key [6–8]. One problem, however, is that it is relatively easy to detect because they explicitly require additional components for clock manipulation, such as a delay locked loop (DLL) in FPGA [6, 7]. It is a component with a complex structure consisting of elements such as delay lines, phase detector, and control logic [9].

In this paper, we conceive a new hardware Trojan technique that reversely exploits the glitch control mechanisms and recalls the natural glitches for stealthy information leakage. Our main idea is to maliciously modify a portion of the glitch control mechanism, e.g., adding multiplexers, so that it is bypassed only when the hardware Trojan is triggered, and observe the occurrence of natural glitches on outputs. Note that because the natural glitch usually comes from a path imbalance in a circuit and a specific pattern applied at the multiple inputs, it would be possible for an adversary who triggered the hardware Trojan to infer the multiple input values from a single output of the target logic, thereby obtaining multiple outputs of the preceding logics. With collected input and output pairs, the adversary may obtain information about the tasks a target system was performing.

This paper is organized as follows. We review natural glitches and glitch control mechanisms in Sect. 2. We present how the glitch control mechanisms can turn into hardware Trojans as we briefly described above in Sect. 3. We perform experiments with a full adder logic as an example in Sect. 4, and discuss the results in Sect. 5. Related works and conclusions follow in Sects. 6 and 7, respectively.

2 Background

We review the natural glitches created on logic circuits, and then explore the representative techniques for the control of natural glitches.

2.1 Natural Glitches in Logic Circuits

In logic circuits, *hazard* implies undesirable and unexpected behaviors and consequences of a system [17]. There are three kinds of glitches created in (synchronous) logic circuits with regard to the hazard: glitches from *static hazard*, *functional hazard*, and *dynamic hazard*. Static and dynamic hazards stand for glitches generated by single input changes, whereas functional hazard mean glitches created by two or more concurrent input changes. All these glitches are naturally generated when logic circuits operate with transitions of multiple input values. The key reason for the glitches is the different arrival and propagation delays of the input signals to logic gates [13]. Once a glitch occurs in signal, an output value fluctuates. For static and functional hazards, a glitch bounces once, whereas for dynamic hazard, a glitch bounces more than once [18].

Glitches may result in unexpected, maybe wrong, behaviors of a given system. In addition, glitches consume extra power of a system, which is critical and undesirable for IoT devices that have limited power resource.

2.2 Glitch Control Mechanisms

Various glitch control mechanisms have been developed and deployed in the real world systems [10–15]. For glitches of static hazard, a systematic glitch elimination method has been developed by adding a redundant logic element. Once sources of static hazard are eliminated, so are the dynamic hazard, hence no more glitches are at the output. Regarding the glitches created by functional hazard, on the other hand, there are not systematic approaches to eliminate them. One possible way to evade glitches from functional hazard is to attach flip-flops at the outputs of a logic circuit that produces glitches, from which catching settled values at every rising or falling edge of clocks [19]. Another method for suppressing glitches from functional hazard is to balance input signals arriving at different instances of time, by inserting programmable delay elements (PDEs) into the input wires and/or output wires [14, 15, 20].

Besides the above mentioned routine approaches, there have been various methods proposed to control glitches on logic circuits. Czajkowski and Brown proposed an algorithm that inserts negative edge-triggered flip-flops (nFFs) to LUTs (Lookup Tables), which produce glitches, on an FPGA [10]. The logic values produced by the LUTs in the previous cycle are maintained by the inserted nFFs to prevent propagation of glitches. Techniques that use disabled flip-flops in logic blocks of an FPGA are also proposed in [11, 12] to control glitches. Lim et al. proposed a technique that reactivates and operates flip-flops with phase-shifted clocks in [11]. Hsieh et al. proposed a technique that utilizes unused flip-flops between functional units and their original registers as “firewall registers” in [12].

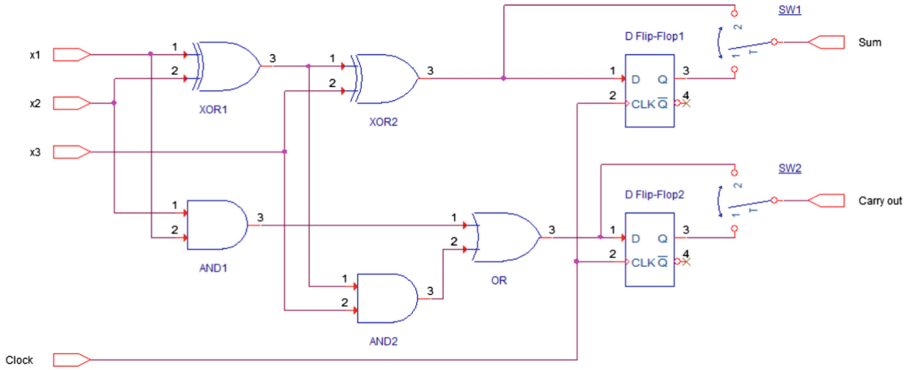


Fig. 1. A proposed hardware Trojan that recalls natural glitches by bypass operations

The flip-flops postpone data propagation for one clock cycle, and hence they control glitches. Vijayakumar and Kundu presented a glitch control mechanism using clock skew scheduling [13]. In this work, the relation between glitches and input signals arriving at various instances of times to a logic gate is studied and used for glitch control. In [14], Lamoureux et al. presented “GlitchLess,” a glitch control technique using PDEs that control signal transmissions, to align the edge of input signals in logic blocks. Dong and Lemieux utilized the concept of GlitchLess with clock skew scheduling and delay padding in [15] to control glitches.

3 Designs of the Proposed Hardware Trojan

Based on the glitch control mechanisms reviewed in the previous section, we design a natural glitch recalling hardware Trojan. While the nature of devised hardware Trojan, exploiting natural glitches, is the same, specific implementation approaches are two-fold according to the glitch control methods: One for bypassing a glitch evading technique, and the other for maliciously modifying programmable delay elements. We illustrate representative hardware Trojans for each of the glitch exploitation cases on a full adder logic circuit, a component of an ALU (Arithmetic Logic Unit).

3.1 Recalling Natural Glitches by Bypass Operations

A hardware Trojan that recalls glitches by bypass operations on a full adder is presented in Fig. 1. x_1 , x_2 , and x_3 are inputs. *Sum* returns summation of the three inputs, and *Carry out* returns a carried value from the summation. On top of the typical full adder logic circuit configuration, there are additional logic elements, D flip-flops and multiplexers that are represented as toggle switches. The purpose of the two D flip-flops is to avoid natural glitches by picking up

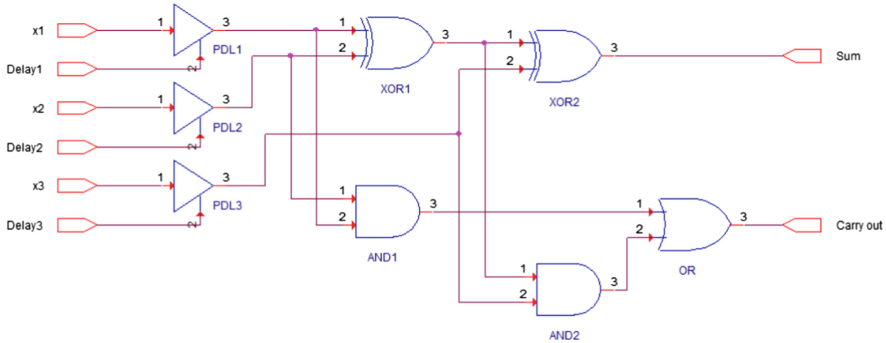


Fig. 2. A proposed hardware Trojan that recalls natural glitches by manipulating programmable delay elements

values at every rising or falling edge of clocks. The *Clock* drives the two D flip-flops. By toggling the switch to position 1, we receive *Sum* and *Carry out* values without glitches through the flip-flops. On the contrary, by toggling the switch to position 2, *Sum* and *Carry out* bypass the inserted flip-flops, thereby we receive output values with naturally generated glitches.

The whole attack scenario with this hardware Trojan is: Selection bit ‘0’ of multiplexers allows normal operations of a full adder in an ALU, whereas selection bit ‘1’ generates outputs with glitches. An adversary can apply a trigger signal to the selection bit of multiplexers based on a rare combination of signals in a given system, or sensor values met specific conditions. Once triggered, with selection bit ‘1’, the hardware Trojan starts to recall glitches by bypass operations. Based on the observed glitches, the adversary may infer values of multiple inputs fed into. By collecting the deduced input values and corresponding output values, the adversary may restore what tasks the full adder or the ALU was performing. So the payload of this hardware Trojan is information leakage.

3.2 Recalling Natural Glitches by Manipulating Programmable Delay Elements

A hardware Trojan that recalls glitches by manipulating programmable delay elements (PDEs) on a full adder is presented in Fig. 2. On top of the typical configuration elements of a full adder, three additional logic components, PDEs, are inserted into each of the three input wires. By setting appropriate *delay* parameters that control the amount of signal propagation delays, one can balance input signal deliveries. By doing so, one can suppress as many glitches as possible. If an adversary manipulates the delay parameters maliciously, so that the PDEs work to break the balance among input signals, natural glitches will revive.

The whole attack scenario of this hardware Trojan is: By maintaining the appropriate delay parameters of PDEs, the outputs will have much lesser natural glitches, whereas the delay parameters are maliciously adjusted, natural

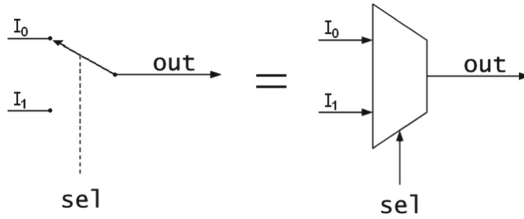


Fig. 4. The switch represented in Fig. 1 (left), and its equivalent logic element, multiplexer (right)

After that, we exploit the glitch avoidance mechanism by adding two 2-by-1 multiplexers at the end of the D flip-flops each. The mechanism of a multiplexer is equivalent to that of a switch with a control knob. A typical multiplexer and a switch with control knob are illustrated in Fig. 4. We connected the output lead of the D flip-flop 1 in Fig. 1, which provides values of *Sum* without glitches, to the input I_0 of a multiplexer, and linked the output of the *XOR2*, which provides values of *Sum* with glitches, to the input I_1 of the multiplexer, where the multiplexer is *SW1* in Fig. 1. Similarly, we fed the output lead of the D flip-flop 2 in Fig. 1, which gives values of *Cout* without glitches, to the input I_0 of another multiplexer, and wired the output of the *OR*, which gives values of *Cout* with glitches, to the input I_1 of the multiplexer, where the multiplexer is *SW2* in Fig. 1. By setting the selection bits of the two multiplexers ‘0’, which is corresponding to input I_0 , the *Sum* and the *Cout* will generate normal output values without glitches. By putting the selection bits of the two multiplexers ‘1’, on the other hand, the *Sum* and the *Cout* will spew outputs with glitches. As can be seen in Fig. 3, there are no glitches in the outputs with selection bit ‘0’, whereas natural glitches are recalled in the outputs with selection bit ‘1’, activating the proposed hardware Trojan.

4.2 Hardware Trojans Exploiting Programmable Delay Elements

Switching gears, we implemented a programmable delay element following the idea illustrated in [20]. The key idea of implementing a programmable delay element is to utilizing a lookup table (LUT) of an FPGA as a buffer or an inverter. A typical high-level structure of a LUT with three input pins a , b , and c is presented in Fig. 5. Each of the SRAM cells has binary values based on a logic expression. Given binary inputs for a , b , and c , these are fed into selection bits of the 2-by-1 multiplexers (MUXs). Based on a routing established by the selection bits of MUXs, certain value in an SRAM comes out through the output pin y . By setting the binary values on the SRAM cells so that the output y is always the same to the input value of c , and the y is nothing to do with the other input values of a and b , the LUT operates like a buffer. In the meanwhile, even if the other two input values of a and b are irrelevant to the output value y , they affect paths from the input c to the output y . In other words, binary

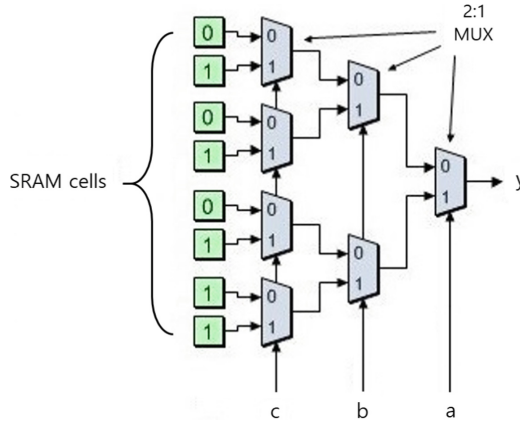


Fig. 5. A typical high-level structure of a LUT with three inputs

value $ab = 00$ allows the shortest path, whereas binary value $ab = 11$ imposes the longest route, from the input c to the output y . By adjusting the value of input pins a and b , we can adjust, or program the amount of delay of the buffer. A LUT in Xilinx Artix-7 FPGA has six input pins and one output pin, so that by adjusting the value of five input pins, we can program delays of it to the resolution of $2^5 = 32$ levels.

With one programmable delay element inserted into the wire of input $x3$ in Fig. 2, we could obtain the *Sum* and the *Cout* without glitches. This is represented in Fig. 6 (upper). In the meanwhile, when we added additional programmable delay elements in the wire of input $x3$, the balance of input signal deliveries among the three inputs $x1$, $x2$, and $x3$ broke, and natural glitches in the *Sum* and the *Cout* revived. This is represented in Fig. 6 (lower).

5 Discussions

5.1 Information Leakage Attacks

From the experiment for the hardware Trojan that recalls natural glitches by bypass operations, it has been clear that an adversary can obtain outputs *Sum* and *Cout* with or without glitches. By setting the selection bit of the two multiplexers ‘0’, the *Sum* and *Cout* generated normal output values without glitches. By putting the selection bit to ‘1’, the *Sum* and *Cout* spewed outputs with glitches. By observing the glitches in *Sum* and *Cout*, the adversary may infer what inputs were fed into the full adder. With the inferred inputs, the adversary may obtain corresponding outputs. By aggregating the inferred input and output pairs, the adversary may finally restore the tasks the full adder was performing, succeeding in an information leakage attack.

As can be seen from the experiment for the hardware Trojan that recalls natural glitches by manipulating programmable delay elements, an adversary can

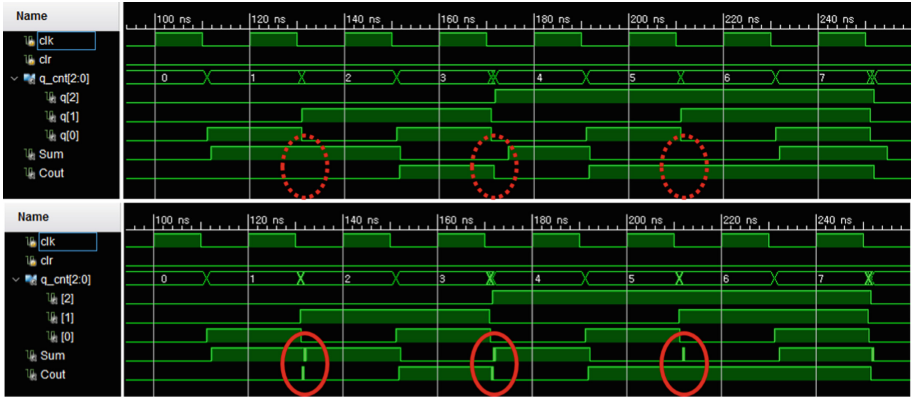


Fig. 6. The outputs of a full adder with a hardware Trojan that recalls natural glitches by manipulating programmable delay elements (PDEs). There are no glitches in outputs with one PDE (upper), whereas glitches in outputs with additional PDEs (lower).

manipulate the parameters of the programmable delay elements. By maliciously adjusting the delay parameters, so that the balance of the signal delivery times among the three inputs is broken, the adversary may obtain *Sum* and *Cout* with glitches. From the two outputs with glitches, the adversary may deduce input values. By collecting the input and output pairs, the adversary may eventually reconstitute the tasks the full adder was performing, accomplishing an information leakage attack.

5.2 Stealthy Nature of the Proposed Hardware Trojan

The proposed hardware Trojan would be usually very tiny compared to a local logic circuit where the hardware Trojan is inserted into, as well as against a whole system; it only needs a certain number of additional multiplexers, or nothing else but maliciously adjusted delay parameters. Accordingly, it would be difficult to point out the place(s) where the hardware Trojan is planted by schematic inspections. In addition, if an adversary figures out any rare trigger conditions, it would be hard to detect the hardware Trojan just by observing operating behaviors of a target logic system.

5.3 Mitigation and Possible Countermeasures: Detection

There have been detection measures proposed against hardware Trojans, and a *logic testing* is one of them, comprehensive and powerful. Logic testing is a technique to develop directed test patterns to activate unknown Trojan instances and propagating their effects to output ports. It is especially effective for detecting ultra-small Trojans [4,5]. The key is how to prepare well for test inputs that reflect rare input combinations. That is, an adversary would always struggle to

devise rare hardware Trojan triggering input combinations. If test inputs cannot cover the rare cases, hardware Trojans will survive the logic testing. On the contrary, test patterns that span well rare combinations would be able to detect hardware Trojans, no matter how small they are, with higher chances.

6 Related Work

6.1 Hardware Trojan Classifications

Banga and Hsiao classified hardware Trojans into two types, *combinational* and *sequential* Trojans [21]. Combinational Trojans are activated at specific conditions of an internal circuit, and sequential Trojans are activated if specific sequences occur in a circuit. Wang et al. classified hardware Trojans based on the three characteristics, *physical*, *activation* and *action* [22]. The physical characteristic is classified further into type (addition/deletion of gates or modification of logics), size (the number of modified components), distribution (the location of a Trojan) and structure (modified layout of a chip). The activation characteristic means that how the Trojans are activated. Inserted Trojans can be activated externally or internally. The last characteristic, action, implies the behavior of inserted Trojans.

This paper follows the classification of [5]. Chakraborty et al. classified hardware Trojans into two mechanisms, *trigger*, activation mechanisms of Trojans, and *payload*, the negative effects by Trojan activation. According to this classification criterion, the trigger of the hardware Trojan proposed in this work is a combinationaly triggered, and the payload of this hardware Trojan is information leakage.

6.2 Information Leakage Using Artificial Glitches

Endo et al. proposed a glitch generator for fault injection in [6]. They generated a glitchy-clock signal using two clock signals with different phases. Two delay locked loop (DLL) circuits in an FPGA are used for the purpose. The glitchy clock is injected for safe-error attacks on “square-and-multiply always” method implemented in an RSA processor. An attacker injects faults at the multiplication process and captures a pair of power traces with and without the fault injection. By comparing differences of the power traces, exponent bits can be leaked.

Similarly, Agoyan et al. proposed a glitch clock generation method using embedded DLLs in an FPGA [7]. Two delayed clocks are generated from an original clock and these clocks are combined based on a trigger signal to create a faulty glitch clock. The faulty glitch clock is injected for Giraud’s one-bit attack to an AES implementation, and is used to leak the round key of AES.

Fukunaga and Takahashi used a glitch clock to attack a cryptographic LSI (Large-Scale Integration) [8]. In this work, the glitch is generated by two external clocks with a two-channel pulse generator. With generated glitch clocks, Piret’s DFA (Differential Fault Analysis) attack is conducted to leak an AES key.

7 Conclusion

We proposed a method to exploit natural glitches, rather than artificial glitches, for information leakage when a hardware Trojan is triggered, and conclude that natural glitches are one of the potential attack vectors, of which the threats should not be neglected. We targeted a full adder in arithmetic logic unit as an example. A limitation of our experiment that implemented the programmable delay element in an FPGA, was the impossibility of the elaborate experiments due to the lack of data on how much delay would actually occur when we adjusted the delay parameters. Since the locations that natural glitches actually occur are relatively fixed, in some cases it is infeasible to achieve information leakage for specific input values. In this case, we can consider mixing artificial glitches into the natural glitches. By mixing them, it would be possible to add minimal logic elements and consume lesser power resource to better avoid detection, which can be evaluated and proved by conducting a comparative research. In the future study, more target examples and experiment studies along with their mitigation work would be expected.

References

1. Xiao, K., Forte, D., Jin, Y., Karri, R., Bhunia, S., Tehranipoor, M.: Hardware Trojans: lessons learned after one decade of research. *ACM Trans. Des. Autom. Electron. Syst. (TODAES)* **22**(1), 6:1–6:23 (2016)
2. Tehranipoor, M., Koushanfar, F.: A survey of hardware Trojan taxonomy and detection. *IEEE Des. Test Comput.* **27**(1), 10–25 (2010)
3. Alkabani, Y., Koushanfar, F.: Designer’s hardware Trojan horse. In: *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)* (2008)
4. Bhunia, S., Hsiao, M.S., Banga, M., Narasimhan, S.: Hardware Trojan attacks: threat analysis and countermeasures. *Proc. IEEE* **102**(8), 1229–1247 (2014)
5. Chakraborty, R.S., Narasimhan, S., Bhunia, S.: Hardware Trojan: threats and emerging solutions. In: *IEEE International High Level Design Validation and Test Workshop (HLDVT)* (2009)
6. Endo, S., Sugawara, T., Homma, N., Aoki, T., Satoh, A.: An on-chip glitchy-clock generator for testing fault injection attacks. *J. Cryptogr. Eng.* **1**, 265–270 (2011)
7. Agoyan, M., Dutertre, J.-M., Naccache, D., Robisson, B., Tria, A.: When clocks fail: on critical paths and clock faults. In: Gollmann, D., Lanet, J.-L., Iguchi-Cartigny, J. (eds.) *CARDIS 2010. LNCS*, vol. 6035, pp. 182–193. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12510-2_13
8. Fukunaga, T., Takahashi, J.: Practical fault attack on a cryptographic LSI with ISO/IEC 18033–3 block ciphers. In: *Fault Diagnosis and Tolerance in Cryptography (FDTC)* (2009)
9. Plants, W.C., Mazumder, N., Kundu, A., Joseph, J., Wong, W.W.: Delay locked loop for an FPGA architecture. Google Patents, U.S. Patent No. 7484113 (2009)
10. Czajkowski, T.S., Brown, S.D.: Using negative edge triggered FFs to reduce glitching power in FPGA circuits. In: *44th ACM/IEEE Design Automation Conference (DAC)* (2007)
11. Lim, H., Lee, K., Cho, Y., Chang, N.: Flip-flop insertion with shifted-phase clocks for FPGA power reductio. In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 335–342 (2005)

12. Hsieh, C.T., Cong, J., Zhang, Z., Chang, S.C.: Behavioral synthesis with activating unused flip-flops for reducing glitch power in FPGA. In: Proceedings of the 2008 Asia and South Pacific Design Automation Conference, pp. 10–15 (2008)
13. Vijayakumar, A., Kundu, S.: Glitch power reduction via clock skew scheduling. In: IEEE Computer Society Annual Symposium on VLSI (ISVLSI) (2014)
14. Lamoureux, J., Lemieux, G.G.F., Wilton, S.J.E.: GlitchLess: dynamic power minimization in FPGAs through edge alignment and glitch filtering. *IEEE Trans. Very Large Scale Integr. VLSI Syst.* **16**(11), 1521–1534 (2008)
15. Dong, X., Lemieux, G.G.F.: PGR: period and glitch reduction via clock skew scheduling, delay padding and GlitchLess. In: International Conference on Field-Programmable Technology (FPT) (2009)
16. Hashimoto, M., Onodera, H., Tamaru, K.: A practical gate resizing technique considering glitch reduction for low power design. In: Proceedings of the 36th Annual ACM/IEEE Design Automation Conference (1999)
17. Valachi, A., Aignătoaiei, B.I., Timiș, M.G.: The comparative study of two analytical methods for detection and elimination of the static hazard in combinational logic circuits. In: 15th International Conference on System Theory, Control, and Computing (ICSTCC) (2011)
18. Givone, D.D.: *Digital Principles and Design*. McGraw-Hill, New York (2003)
19. Shah, K.: An innovative approach to detect glitches in hardware implementations on FPGAs. Master of Science thesis (2013)
20. Majzoobi, M., Koushanfar, F., Devadas, S.: FPGA PUF using programmable delay lines. In: IEEE International Workshop on Information Forensics and Security (WIFS) (2010)
21. Banga, M., Hsiao, M.S.: A region based approach for the identification of hardware Trojans. In: IEEE International Workshop on Hardware-Oriented Security and Trust (HOST) (2008)
22. Wang, X., Tehranipoor, M., Plusquellic, J.: Detecting malicious inclusions in secure hardware: challenges and solutions. In: IEEE International Workshop on Hardware-Oriented Security and Trust (HOST) (2008)



Design and Implementation of Android Container Monitoring Server and Agent

Kwon-Jin Yoon¹, Jaehyeon Yoon², and Souhwan Jung³(✉)

¹ Lotte Data Communication, Seoul, Korea
jinful@naver.com

² Department of Software Convergence, Soongsil University, Seoul, Korea
yjh7593@naver.com

³ School of Electronic Engineering, Soongsil University, Seoul, Korea
souhwanj@ssu.ac.kr

Abstract. Security companies have been struggling with malware analysis for many years as they become more and more intelligent. In order to yield better analysis result, the analysis environment must be well-equipped to cover wide range of applications. For instance, applications are analyzed dynamically in a period of time in various environments, including a virtual environment and a real device. Yet many intelligent Android malware still find a way to stop running when they inspect the environment. In order to solve this problem, Android container technology has been studied, but there is still a lack of research on monitoring server that can analyze operation information in malicious application. This paper proposes a server-agent model to monitor application behaviors in Android container. We design and implement agents that collect behavioral information from malicious applications running in the Android containers, and monitoring server that organizes these information for further analyses.

Keywords: Monitoring server and agent · Android container

1 Introduction

As the IT technology advances, the BYOD age has come, and individuals are increasingly using mobile applications such as smart phones and tablet PCs to conduct business activities or to store personal information [1]. For smartphones and tablet PCs, Android OS is the most popular mobile OS. In particular, according to research by Gartner, the market share of smartphones sold in the second quarter of 2016 was 86.2%, up 4% from the same period last year [2].

By this way, Android OS continues to increase its share, taking advantage of openness in the mobile OS market. Malicious applications may take actions such as stealing private information stored in the smartphone or inducing charging to the user, or may perform DDoS attacks and massive distribution of spams by making the user's device a zombie phone through remote control [3].

There are two methods for analyzing malicious applications: dynamic analysis and static analysis [4]. In the beginning, the malicious application used only the method of

forgery in a very simple form, so it was easy to judge whether or not the malicious application was detected by the above static analysis method. However, it is getting more difficult to analyze malicious applications statically [5], as hackers make malicious applications start to use methods such as application update method and code obfuscation that update only a specific part without user's approval process. For this reason, security vendors who need to block malicious applications in a short period of time use dynamic analysis rather than limited static analysis to determine whether they are malicious. However, if the dynamic analysis is performed in a real device, it takes a long time to analyze, and when the debugging environment is detected in the app, analysis is limited and the device may be damaged during operation of the malicious application. For that reason, security vendors are doing dynamic analysis in a virtual environment like emulators.

In the case of the recent Android malicious applications, it is becoming difficult to analyze the dynamic environment in the virtual environment, as well as checking whether the real environment of the execution environment is rooted and connected with debugging connection or not. In such a situation, technology is being developed to provide a non-rooted Android environment that can be analyzed using container technology to create an Android environment like a real device. However, there is still a lack of research on monitoring technology that provides dynamic analysis information about malicious Android applications running in a container environment.

In this paper, we propose a method to design and implement a monitoring server and agent that provides malicious behavior information and analysis files to analysts who want to monitor malicious behavior of applications in Android container in real time.

In Sect. 2 of this paper, we discuss the method of dynamic analysis of Android malicious application and the methods that we have already monitored. Section 3 describes the design method and implementation method of the proposed monitoring agent and server. In Sect. 4, we review the implemented server and agent, analyze the implementation results, and conclude the paper in Sect. 5.

2 Related Works

Android OS users can easily develop their applications using the Android emulator regardless of their hardware type [7]. Nowadays, the Android Market has provided many apps through open source offering, but malicious applications have also been developed and circulated on the market. Although the Android application market has been trying to prevent malicious applications, it is hard to prevent registration of malicious applications by circumventing the market. That is why many security companies are participating in the security market to prevent damage to Android devices.

Initially, malicious applications can be easily analyzed and judged through static analysis. However, at present, it is difficult to analyze by static analysis only because of obfuscation of Android DEX file or multi DEX file. In addition, some malicious applications of Android may download malicious applications after they have been installed first. Therefore, it is necessary to perform dynamic analysis to monitor system

calls and APIs by dynamically running malicious applications [8]. The current dynamic analysis monitoring method uses a method of installing malicious application, monitoring malicious activity information generated during execution, and judging whether malicious application is based on information [9]. In the process, various methods are proposed and used in order to prevent the dynamic analysis environment from being detected by malicious applications. However, some of the latest malicious applications use anti-emulator technology to avoid running in the emulator. In this Section, we will discuss the various methods proposed for dynamic analysis using phones and emulators, and the techniques used by malicious applications to avoid detection in the emulator.

2.1 Dynamic Analysis for Android

Dynamic analysis uses a method of extracting an API using a specific technology or hooking system call functions to collect various information.

Miao et al. [10] proposed a method to record system calls and attributes called by API Capture tool to malicious applications running on emulator without changing the system kernel. This method integrates the API Capture tool created in the QEMU-based emulator and runs a malicious application on the host OS to monitor the system call or variable values through a surveillance environment that is not detected by malicious applications.

Yan and Yin [11] proposes a dynamic analysis method that adds tools for tracking native and Dalvik commands, a tool for collecting API-level activity information, and a taint analysis tool for tracking information leakage over DroidScope, an Android analysis platform. This method provides a unified interface to the user, enabling dynamic analysis of both native and Dalvik bytecodes, and has the advantage of keeping the JIT compilation time intact without changing the Android system.

Blasing et al. [12] proposed a platform for performing static analysis and dynamic analysis on Android called AASandbox. Sandboxes and detection algorithms in the cloud will quickly and extensively search for suspicious software in mobile markets similar to Google Android Market, and then perform static and dynamic analysis. `System.getRuntime ()` and `exec (..)`, which can be used to create a raw child process and exceed the normal application lifecycle, reflection that can be used to circumvent API restrictions, services and IPC terms that may cause battery drain or device CPU overload and Code to determining which permissions to grant when installing using Android permissions. For dynamic analysis, the system call function is hooked up for logging, the application is executed, and then the system level log is generated using the Android Monkey tool, and the log is used to determine whether the application is a malicious application.

Burguera et al. [13] used an existing approach to analyze behavior information of applications running on the Android platform. And a method of collecting information generated from the phone by installing an application capable of analyzing the phone to a large number of users. The installed application uses an analysis method to monitor the system calls and uses the method proposed in [14]. As a result of using system calls, `open ()`, `read ()`, `access ()`, `chmod ()` and `chown ()` were used as the system calls most frequently used by malicious code and they were able to figure out the attack on the Trojan horse.

Dash et al. [16] proposed a method to classify malware based on the actions collected through dynamic analysis. This paper uses CopperDroid for dynamic analysis of applications [17]. CopperDroid collects dynamic behaviors using a qemu-based emulator and a customized OS.

Saracino et al. [18] introduces a method to prevent malicious behavior by analyzing the behavior of the app in real time in the real device. The behavior of the app is divided into kernel, application, user, and package levels. This method aggregates the collected information to classify the app and notify the user. This dynamic analysis performed on a real device requires the device to be rooted. The app can detect whether the device is rooted and perform only normal actions.

2.2 Anti-emulator Technology

Among the latest malicious applications, there are malicious applications that have anti-emulator technology to avoid analysis in the emulator [6]. The techniques used in the anti-emulator technology are used to identify both the static and dynamic elements and to identify the characteristics that occur when using the hypervisor. An anti-emulator technology using static elements identifies the following information: The unique number for the phone called IMEI, Information related to SIM card called IMSI, Information related to system properties in the current build, the network properties that only the emulator has. Verification is possible by using the dynamic elements of various sensors present in the mobile phone. Typically, the fingerprint approach used in [15] detects the emulator by focusing on smartphone fingerprinting based on sensor defects and incompleteness.

As described above, among the latest Android malicious applications, anti-emulator technology is used to avoid analysis through the emulator, and research on Android container technology similar to a real device is being studied, but there is still a lack of technology for monitoring malicious application on the corresponding platform. Therefore, in Sect. 3 of this paper, we present a server and an agent for monitoring an Android container.

3 Proposed Monitoring Methods

In this paper, we propose an Android container monitoring technology that monitors the malicious behavior of Android malicious application running in the Android container in another space. This monitoring can detect the processes of the Android container in Linux, but the processes running on Linux can't be detected by the Android container. Therefore, it has the advantage that it can use technologies that have not been used in the development of detection technology of malicious Android applications and can use general Linux tools that were not available in mobile. Also, when analyzing using real device, it took a lot of time to recover the device if the OS was tampered by malicious application. Even if the crash occurs in the Android container, this monitoring technology using container environment can know the reason the environment crashed.

3.1 Agent Design

In order to design the agent proposed in this paper, we proceeded with the following procedures. We determine the agent’s location and determine the language to use when creating the agent. Based on the basic information to be collected by the agent and the advantages of using the Android container, the information that can be collected is determined and finally the agent is implemented.

The agent operates on the Linux operating system, which is a space independent of the Android container, as shown in Fig. 1. Space where the agent is running can monitor Android space without being detected by malicious application detection techniques. Information gathered for Android malicious application analysis is shown in Table 1.

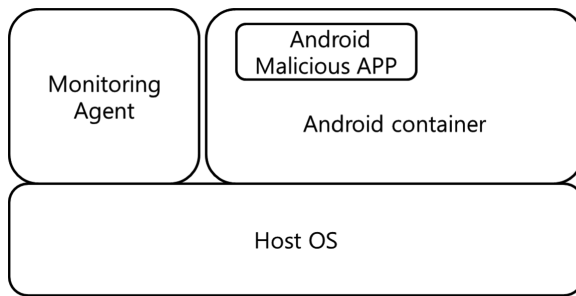


Fig. 1. Location of monitoring agent

Table 1. Information collected for application analysis.

Source of information	Description
<i>/proc/cpuinfo</i>	CPU information
<i>/proc/version</i>	Linux version information
<code>ip addr grep eth0</code>	IP information on the platform
<code>ps -auxf</code>	Processes information
<code>android logcat -d grep "E Error"</code>	Android logcat error information
<i>/var/log/kern.log</i>	Kernel panic information
<i>/sys/kernel/debug/binder/failed_transaction_log</i>	Binder transmission failure log
<i>/sys/kernel/debug/binder/transaction_log</i>	Binder transmission log
<code>android logcat</code>	Logcat full information
<i>/sys/kernel/debug/binder/state</i>	Binder usage information
<code>Tcpdump</code>	Network packet information
Automatically generated when a process crash occurs	Crash dump information
<i>/proc/pid/maps, /proc/pid/mem</i>	Memory dump information

In the implementation, we construct an agent that collects and transmits the information as shown in Fig. 2. As soon as the agent is executed, it checks whether the agent is re-executed due to the kernel panic. If the agent is re-executed by panic, it transmits the information to the server and waits for the Android container to be executed. Then, when the Android container is executed, tcpdump is used to collect network information, and thereafter periodically collects and transmits information related to the malicious Android application.

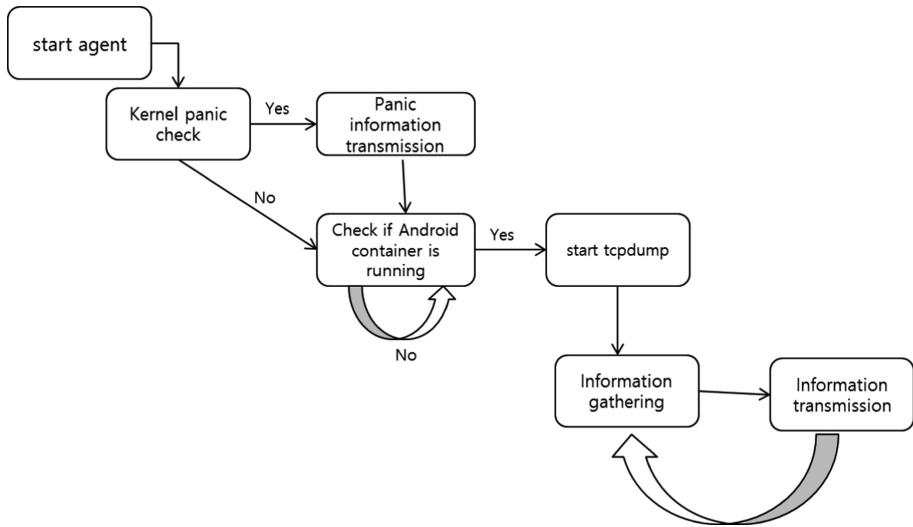


Fig. 2. Monitoring agent operation process

3.2 Server Design

The design process of the server proposed in this paper proceeded in the following order. First of all, the method of receiving information was determined and then the form of information storage was determined. After that, we decided how to provide the information to the user and finally we built the server.

The information transmitted to the server is received and stored based on size as shown in Table 2.

The small information (CPU, Linux version, IP, process, Android logcat error, kernel panic, binder transmission log, binder transmission log information) coming from the Curl command was received as argument using PHP and stored in DB. We decided to use FTP to send and receive files for the file containing the whole Android log information, the file storing the binder information, the file containing the network related packet information, and the crash dump file created by the crash, because the size of the information is large. In order to provide only the information assigned to the user, the user ID and the IP are mapped so that only the information corresponding to the IP is accessible to the user.

Table 2. Method of transmitting and receiving information.

Transmission method	Information description
Sent with the Curl command => Get PHP parameter and save to DB	CPU information
	Linux version information
	IP information on the platform
	Processes information
	Android logcat error information
	Kernel panic information
	Binder transmission failure log
	Binder transmission log
Transfer using FTP => Receive using FTP	Logcat full information
	Binder usage information for processors
	Network packet information
	Crash dump information
Sending to socket => Receive data by socket and save to file	Memory dump information

Finally, the functions and operation procedures of the server are as follows. When a user subscribes for membership, the administrator maps the user's ID and IP, and grants the user the right to the information. Later, when the user connects, you can see information about the Android container running on the mapped IP.

4 Implementation and Results

In order to monitor malicious applications running in the Android container, we designed and implemented an agent running on Linux and a server that receives the information sent by the agent and the agent sends it to each user. The monitoring server provides information about the malicious application that the user wanted to analyze and provides information related to the crash that was previously difficult to provide.

4.1 Agent

The agent has 4 functions which is described as follows.

Core. This function first checks the kernel panic at run time and then sends the information to the server if panic occurs. After that, it transmits CPU information, Linux version information, IP information and initial process information, and then continuously transmits process and logcat information.

TcpTransmitter. Periodically, use the tcpdump command to send network packet information to the monitoring server, and use ftp to send a pcap file to help analyze network packets.

TcpdumpManager. Periodically, the container is checked to see if the container has been launched, and when the container is launched, the tcpdump command is executed to create a pcap file containing network packet information from the Ethernet card, and It stops the tcpdump process that is being executed using the kill command when the Android container is shut down.

Memdump. A program that dumps a process running in an Android container to */proc/pid/maps* file and */proc/pid/mem* file using a call to the ptrace system.

4.2 Server

The server consists of a function to receive the information transmitted by the agent and a function to show the received information to the user. Users can log in and join the site by entering their ID and password. If a user joins a membership, they can't access it immediately and need to approve the administrator. The administrator must map the IP of the platform and the user and approve it before the user can access the site.

When an authorized user connects, the user can view the CPU, Linux version, and IP information of the platform on which the analysis is performed. Process information provides real-time process change information and memory dump information for malicious applications, as shown in Fig. 3.

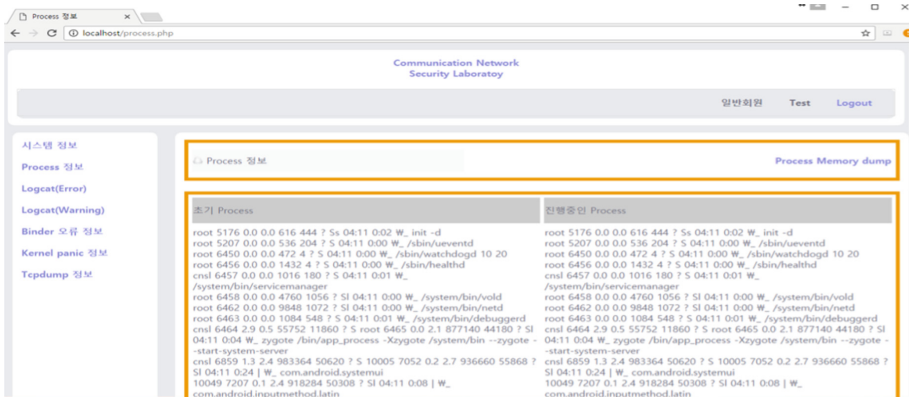


Fig. 3. Pages for process information

Logcat information periodically updates and displays log error information generated when a malicious application runs in the Android container, and can download the entire log information, as shown in Fig. 4. Binder information provides kernel-level binder error information and transaction failure log information that occurred when using binder, as shown in Fig. 5.

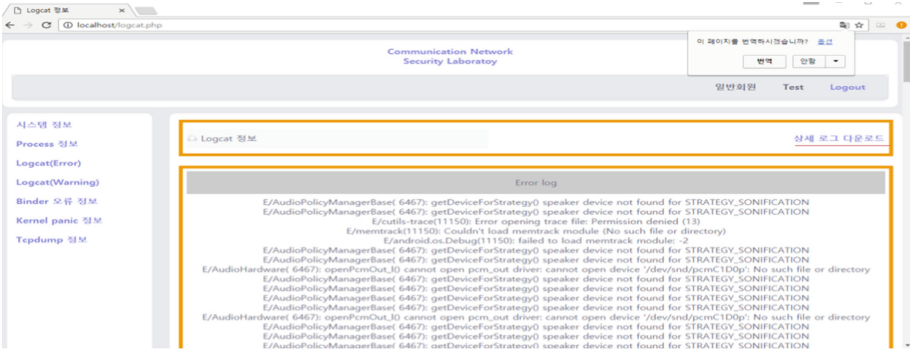


Fig. 4. Page for error log

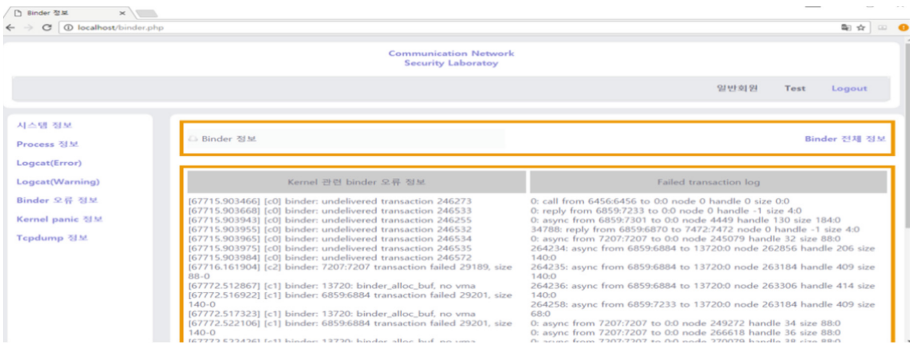


Fig. 5. Page for binder information

If a kernel-related error occurs, you can see information about it in the kernel panic information, as shown in Fig. 6. In addition, Tcpcdump information can be used to check network packet information in the Android container, as shown in Fig. 7. The packet file can be downloaded in the form of Pcap.

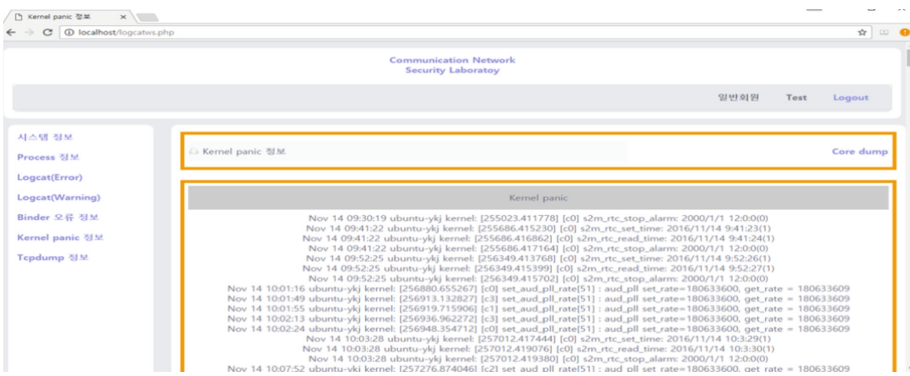


Fig. 6. Page for kernel panic information

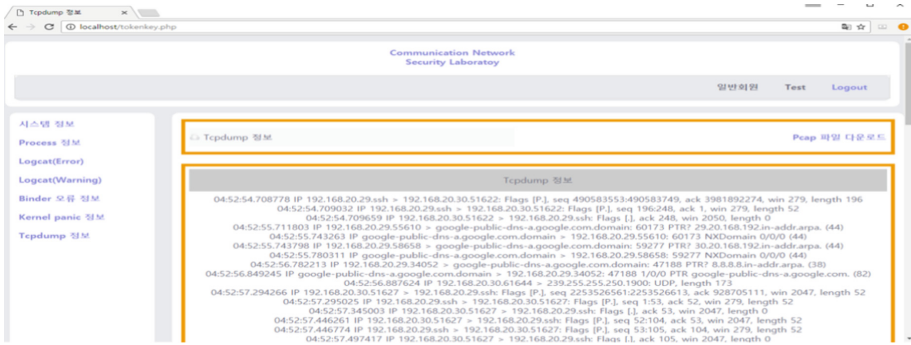


Fig. 7. Page for network packet information

As described above, the server stores the information sent by the agent and provides the mapped information when the user logs in, thereby allowing the user to monitor the cause and information of the crash of Android through the server.

5 Conclusion

In this paper, we propose to design and build an Android container monitoring server and agent. We developed a monitoring method that can easily view the dynamic behavior information of malicious application by matching IP with user on monitoring server and providing IP based malicious behavior information to each user. The proposed monitoring takes advantage of the fact that the malicious application, which is an advantage of using the Android container technology, can be monitored in another independent space.

We use Android Container technology to cope with the situation where the latest malicious applications check anti-emulator technology and super-user privilege to make analysis difficult. Malicious applications normally behave maliciously in a container and monitor the behavior of malicious applications in Linux, a separate space. Using this monitoring, you can analyze various apps that were difficult to analyze in a real device or emulator environment. In addition, malicious applications with superuser privileges can't detect the monitoring, and can use several functions that were previously used for analysis in Linux environments such as Tcpcdump. Finally, by constructing a server, malicious information can be easily accessed by users, and real-time change provides malicious information.

Acknowledgement. This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2012-0-00646) supervised by the IITP (Institute for Information & communications Technology Promotion).

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. 2016-0-00078, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning).

References

1. Disterer, G., Kleiner, C.: BYOD bring your own device. *Proc. Technol.* **9**, 43–53 (2013)
2. Gartner Group: Gartner Says Five of Top 10 Worldwide Mobile Phone Vendors Increased Sales in Second Quarter of 2016. <http://www.gartner.com/newsroom/id/3415117>
3. Felt, A.P., Finifter, M., Chin, E., Hanna, S., Wagner, D.: A survey of mobile malware in the wild. In: *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, pp. 3–14 (2011)
4. Alazab, M., Moonsamy, V., Batten, L.: Analysis of malicious and benign android applications. In: *2012 32nd International Conference on Distributed Computing Systems Workshops, ICDCSW (2012)*
5. Rastogi, V., Chen, Y., Jiang, X.: DroidChameleon: evaluating Android anti-malware against transformation attacks. In: *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, pp. 329–334 (2013)
6. Petsas, T., Voyatzis, G., Athanasopoulos, E., Polychronakis, M., Ioannidis, S.: Rage against the virtual machine: hindering dynamic analysis of Android malware. In: *Proceedings of the Seventh European Workshop on System Security (2014)*
7. Goadrich, M.H., Rogers, M.P.: Smart smartphone development: iOS versus Android. In: *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, pp. 607–612 (2011)
8. Afonso, V.M., de Amorim, M.F., Gregio, A.R.A., Junquera, G.B., de Geus, P.L.: Identifying Android malware using dynamically obtained features. *J. Comput. Virol. Hacking Tech.* **11** (1), 9–17 (2015)
9. Isohara, T., Takemori, K., Kubota, A.: Kernel-based behavior analysis for Android malware detection. In: *2011 Seventh International Conference on Computational Intelligence and Security, CIS (2012)*
10. Miao, Q.-G., Yun-Wang, Cao, Y.: API capture—a tool for monitoring the behavior of malware. In: *3rd International Conference on Advanced Computer Theory and Engineering*, pp. 390–394 (2010)
11. Yan, L.K., Yin, H.: DroidScope: seamlessly reconstructing the OS and Dalvik semantic views for dynamic Android malware analysis. In: *Proceedings of the 21st USENIX Conference on Security Symposium*, p. 29 (2012)
12. Blasing, T., Batyuk, L., Schmidt, A.D.: An Android application sandbox system for suspicious software detection. In: *Malicious and Unwanted Software, MALWARE (2010)*
13. Burguera, I., Zurutuza, U., Nadjm-Tehrani, S.: Crowdroid: behavior-based malware detection system for Android. In: *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, pp. 15–26 (2011)
14. Hofmeyr, S.A., Forrest, S., Somayaji, A.: Intrusion detection using sequences of system calls. *J. Comput. Secur.* **6**, 151–180 (1998)
15. Dey, S., Roy, N., Xu, W., Nelakuditi, S.: ACM HotMobile 2013 poster: leveraging imperfections of sensors for fingerprinting smartphones. *SIGMOBILE Mob. Comput. Commun. Rev.* **17**(3), 21–22 (2013)
16. Dash, S.K., Suarez-Tangil, G., Khan, S., Tam, K., Ahmadi, M., Kinder, J., Cavallaro, L.: DroidScribe: classifying android malware based on runtime behavior. In: *Security and Privacy Workshops, SPW*, pp. 252–261. IEEE, May 2016
17. Tam, K., Khan, S.J., Fattori, A., Cavallaro, L.: CopperDroid: automatic reconstruction of Android malware behaviors. In: *NDSS*, February 2015
18. Saracino, A., Sgandurra, D., Dini, G., Martinelli, F.: MADAM: effective and efficient behavior-based android malware detection and prevention. *IEEE Trans. Dependable Secur. Comput.* (2016)



Improved EM Side-Channel Authentication Using Profile-Based XOR Model

Momoka Kasuya^(✉) and Kazuo Sakiyama

Department of Informatics, The University of Electro-Communications,
1-5-1 Chofugaoka, Chofu, Tokyo 182-8585, Japan
{m.kasuya,sakiyama}@uec.ac.jp

Abstract. A new approach for authentication, side-channel authentication, has been proposed. In side-channel authentication, the authenticity of the device is confirmed with high accuracy by using electromagnetic radiation from the device and response in the conventional challenge–response authentication. The side-channel model or profiled template is used as one of the inputs of the distinguisher when authenticated. The performance of side-channel authentication is greatly affected by the precision of the model or template. In this paper, we evaluate the authentication performance when using profile- and non-profile-based HD models and a profile-based XOR model. We report the results of the experiment in detail using FPGA.

Keywords: Side-channel information · Profiling model
Authentication

1 Introduction

Authentication technology is used in many systems in order to identify users. Authentication approaches can be classified into three types: knowledge-based authentication, possession-based authentication, and biometric-based authentication. Security is improved by combining various authentication technologies, e.g., when withdrawing money from an account, password- and possession-based authentication technology are used. In electronic money or vehicle keyless entry systems, the identification of the holder or owner is verified through possession-based authentication. By utilizing the characteristics of possession-based authentication method, various issues such as car theft [1] and increase in fake products [2] have occurred. Therefore, it is necessary to provide high security by a possession-based authentication system.

As a technology for improving security, Physical Unclonable Function (PUF) has been proposed in [3, 4]. This technology can be applied to an authentication system by utilizing individual differences arising at the time of manufacture. PUF is used to confirm whether or not users own legitimate products. The

manufacturer pre-registers the output for the input data. Then, consumers can confirm the legitimacy of products based on whether the input data has the same output as the registered data.

Another method that has been proposed to improve security is an authentication method using side-channel information, e.g., power consumption and ElectroMagnetic (EM) radiation during cryptographic processing, called side-channel authentication. In side-channel authentication, side-channel information is handled as a device fingerprint. Devices are identified by using the property that the side-channel information leaks different information depending on secret keys of encryption hardware. The previous work [5] was the first paper that experimentally confirmed that a device can be identified with high accuracy using only leaked EM radiation. Side-channel information is generally known to be used in the key-recovery attacks [6–8]. On the other hand, a constructive use of side-channel information is Trojan detection on IC chips [9]. Side-channel information is not just for attacks.

In the side-channel attack, an analysis method known as Correlation Power Analysis (CPA) uses a correlation coefficient when the secret key is guessed [10]. One of the most important factors in calculating the correlation coefficient is a side-channel model. The model is an abstraction of the operation of the encryption hardware. In general, the non-profile-based Hamming Distance (HD) model has been used in the derivation of correlation coefficients. Meanwhile, a profile-based XOR model has been proposed in previous work [11]. This model shows good performance compared to the HD model because it is possible to carry out attacks with a small amount of information. The XOR model was proposed in the 2nd DPA contest, where attacker algorithms using power consumption during Advances Encryption Standard (AES) encryption processing competed.

The acquired side-channel information varies significantly depending on the position of the EM probe in the case of using leaked EM radiation. Therefore, in side-channel analysis using a profile-based XOR model, it is assumed that the attacker can obtain side-channel information at the same position during profiling and attack. However, when considering an actual attack scenario, this assumption is not always feasible. In contrast, in the side-channel authentication system, the system designer can determine the acquisition position of side-channel information with high accuracy. In this paper, we quantitatively evaluate the amount of information leakage of EM radiation from the AES encryption hardware using profile- and non-profile-based HD models and profile-based XOR model.

2 Preliminaries

2.1 Notations

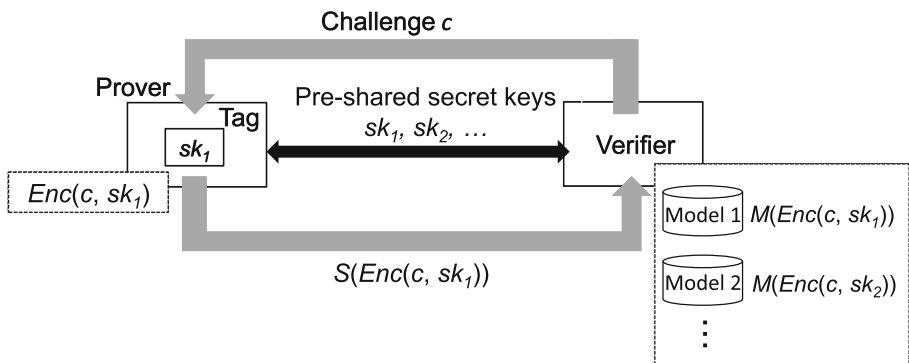
The notations used in this paper are summarized as follows (Table 1).

Table 1. Notations

n	The number of rounds in AES encryption process
N	The intermediate value number
b	S-box number ($1 \leq b \leq 16$)
i	The intermediate value XORed result for 8 bit
\mathbf{A}	The model value of EM radiation for profile-based XOR model
\mathbf{A}'	The model value of EM radiation for profile-based HD model
a	The element of \mathbf{A}
\mathbf{S}	The side-channel information
\mathbf{X}	The set of intermediate value (matrix consisted of 0 and 1)
x	The element of \mathbf{X}

2.2 Overview of Side-Channel Authentication

In previous work [5], four types of side-channel authentication methods have been proposed: Challenge-S-Response, Challenge-S, S-response, and Only-S authentication. In this paper, we focus on the Challenge-S authentication method in order to confirm that the prover can be correctly identified only by side-channel information. The measured side-channel information consists of the signal depending on the secret key and noises. It is difficult for an attacker to distinguish between signal and noise. Meanwhile, as the verifier shares the secret key, the signal can be extracted. Therefore, it is possible to establish a communications path between information even under a noisy environment. The flow of the Challenge-S authentication method is shown in Fig. 1. The secret keys is pre-shared between the prover and verifier as well as the conventional challenge-response authentication. After that, the prover performs AES¹ encryption using the secret key sk_1

**Fig. 1.** The overview of the Challenge-S authentication method

¹ AES can be replaced with other symmetric-key ciphers.

and challenge c supplied by the verifier. The verifier acquires the side-channel information that leaks during the encryption process, and identifies the prover by deriving the value of the correlation coefficient. The correlation coefficient is calculated from the side-channel information and model. When the system exceeds a pre-determined threshold, the authentication is successful.

In the security model we deal with in this paper, the relay attack should be prevented cost-efficiently for practical authentication systems. Side-channel authentication could be used to limit the distance between the prover and verifier. For example, this authentication can be a countermeasure against relay attacks such as [4], where the attacker uses Internet communications. The verifier detects the relay attack by measuring and restricting the authentication time, i.e., the timed exchange of challenges and responses between the prover and verifier [12, 13]. The difference in authentication time is affected by the amount of transmitted data. In a keyless entry system, we believe that authentication security is improved by using side-channel information instead of response of challenge–response authentication. When transmitting side-channel information from the prover, the elapsed time is in the nanosecond order, whereas the response is in the microsecond order [1]. Therefore, in order to relay and reproduce side-channel information, the attacker must measure the data with finer sampling than conventional challenge–response authentication. Some keyless entry systems use 128-bit AES, i.e., communication of several hundred bits. As side-channel information is analog data, the amount of data when digitized becomes larger. Thus, it takes attackers longer to transmit data compared to challenge–response authentication. If an attacker uses equipment that can measure side-channel information with the same accuracy as the verifier, then side-channel information can be measured. Furthermore, it is difficult to prevent attacks, where data can be transmitted.

Meanwhile, the risk of the side-channel attack must be considered when using side-channel information for authentication. One way to reduce the risk of the side-channel attacks is to reduce leaked side-channel information. In this regard, it would be ideal to use the side-channel information leaked by one encryption process. In the authentication system, the intermediate values of AES encryption can be obtained as the prover and verifier share the secret key in advance. Therefore, it is desirable to use all the round information in order to perform authentication more efficiently.

We perform authentication using leaked EM radiation from the n -round AES proposed in previous work [5]. The n represents the number of round functions in the AES encryption process, e.g., $n = 10$ for a standard 128-bit AES. By using multiple rounds, it is possible to authenticate with fewer numbers of encryption when attacked. Side-channel attacks are key-recovery attacks performed by measuring numerous combinations of side-channel information and ciphertext or plaintext. If authentication is performed with EM radiation leaked from one encryption process using n -round AES, then the side-channel information that an attacker can eavesdrop on is reduced, i.e., it becomes difficult to guess the secret key.

Table 2. Comparison between proposed HD and XOR model

	HD model	XOR model
Proposer	Brier et al. [10]	Li et al. [14]
Classification	9	256
Profiling	× (Unnecessary)	○ (Necessary)
Side-channel information	Power consumption	Power consumption

2.3 The Model for Side-Channel Information

In general, the HD model is frequently used in side-channel attack as a non-profile-based model. Non-profile-based models are highly versatile because they are created from a mathematical model, whereas profile-based models are derived from the characteristics of the target prover. The profiling model shows good performance overall compared to non-profiling model.

There is a profile-based model using XOR called the XOR profiling model. In the XOR profiling model of AES encryption hardware, attack accuracy was improved by utilizing the following two properties in previous work.

- The power consumption varied based on changes in the intermediate value that was stored in the register before and after the clock.
- Since the implemented space of individual S-box is limited, each S-box module has different power consumption characteristics when implementing multiple S-boxes.

EM radiation is assumed to have the two properties similar to power consumption.

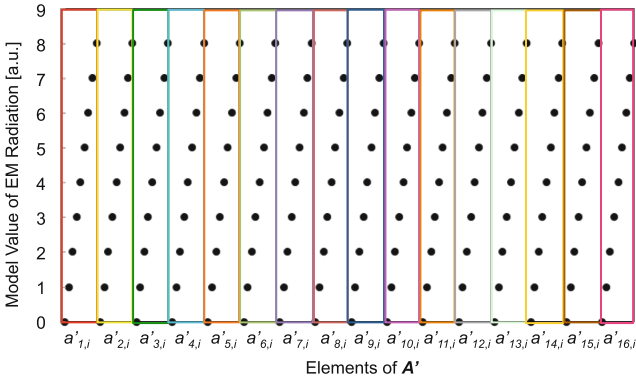
Table 2 shows the comparison between the HD and XOR models at the time of proposal. The 8-bit HD model was classified into nine classes, from zero to eight, according to HD intermediate values between $n - 1$ and n rounds. In the profile-based XOR model, the intermediate value of each round is XORed. As XORed results for 8-bit take 255 from 0, these can be classified into 256 groups. Therefore, more detailed classifications can be realized compared to the HD model. When deriving profile-based XOR model for 128-bit, it is necessary to take into account the difference between 16 S-box instances. The EM radiation is classified into 256×16 groups according to the XOR values and individual S-box, as the characteristic of each S-box varies. Here, assuming that the model $a_{b,i}$ such that the XOR values of the b -th S-box value is i , then the model \mathbf{A} satisfies

$$\mathbf{A} = (a_{1,0} \ a_{1,1} \ a_{1,2} \ \dots \ a_{16,254} \ a_{16,255}) .$$

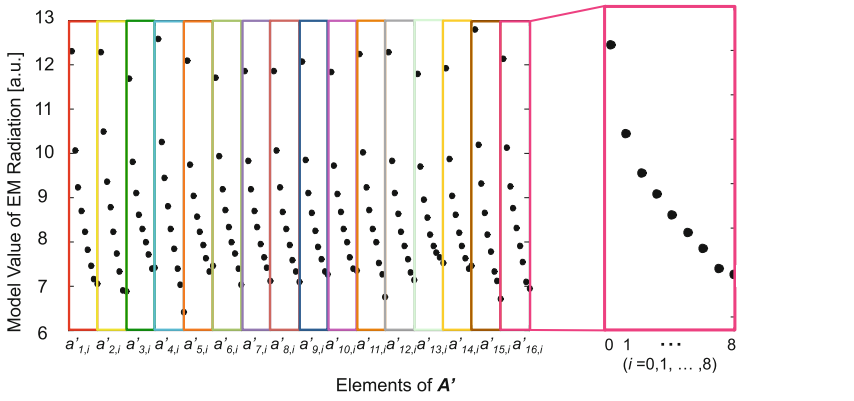
When expressing the HD model as a profiling model, the model \mathbf{A}' for individual S-box is represented as

$$\mathbf{A}' = (a'_{1,0} \ a'_{1,1} \ a'_{1,2} \ \dots \ a'_{16,7} \ a'_{16,8}) ,$$

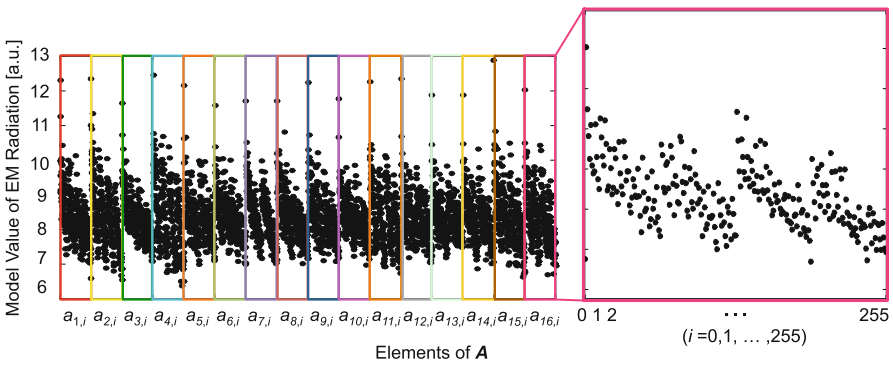
where $a'_{b,i}$ means that the HD values of the b -th S-box value is i .



(a) Non-Profile-Based HD Model



(b) Profile-Based HD Model



(c) Profile-Based XOR Model

Fig. 2. Elements of A and A'

3 Evaluation of the Amount of Information Leakage Using XOR Profiling Model

3.1 Profiling Phase

As pre-processing, EM radiation is observed for 10,000 secret keys with fixed plaintext. A profiling phase is carried out using the corresponding intermediate values of 30 rounds in each EM trace, i.e., a total of 300,000 rounds of information. For the profiling phase, we estimate matrix \mathbf{A} using the sparse linear equations and least squares method, i.e., the `lsqr` instruction in the MATLAB. The 4 to 33 rounds in 1000-round AES are used in the profiling phase in this experiment².

Here, \mathbf{x}_n^b represents the HD value or XORed value between $n - 1$ -th and n rounds. When deriving the profiling data based on the HD and XOR models for 8 bits, \mathbf{x}_n^b is denoted as matrix (9×1) and (256×1) , respectively. If the 8-bit HD model is 0, then the matrix element $(1, 1)$ is 1 and the others are 0. The \mathbf{X} is composed of a set \mathbf{x}_n^b and represented by a matrix $(144 \times n)$ and $(4096 \times n)$. \mathbf{X} can be expressed as

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^1 & \mathbf{x}_2^1 & \dots & \mathbf{x}_n^1 \\ \mathbf{x}_1^2 & \mathbf{x}_2^2 & \dots & \mathbf{x}_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_1^{16} & \mathbf{x}_2^{16} & \dots & \mathbf{x}_n^{16} \end{pmatrix}.$$

The side-channel information can be denoted as

$$\mathbf{S} = (s_1^t \ s_2^t \ s_3^t \ \dots \ s_n^t),$$

where s is the indicated observed EM radiation data and t is the sampling point at which the correlation coefficient is the highest. Matrix \mathbf{A} is derived by utilizing the following relational expression of

$$\mathbf{A} \cdot \mathbf{X} = \mathbf{S}.$$

Figure 2 shows the model values of EM radiation for each S-box of the HD model and XOR profiling model in the experiment. Figure 2(a) shows the model values of EM radiation assumed for a general non-profile-based HD model. This model was created based on the assumption that the leaked side-channel information increased as the HD value increased. Figures 2(b) and (c) show the model values of profiled results based on the HD and XOR models, respectively. From these figures, the observed EM radiation can be predicted from the intermediate values for each round.

² The intermediate values influenced by a biased plaintext are not used.

3.2 Authentication Phase

In this section, the correlation coefficients are derived from the prepared matrix \mathbf{A} , and 965-round information is not used in the profiling phase. The final round of AES is not used to derive the correlation coefficients because the round function process differs from other rounds. The correlation coefficients are calculated from $\mathbf{A}' \cdot \mathbf{X}'$ and EM radiation \mathbf{S}' . Furthermore, \mathbf{X}' is calculated from the challenge and a secret key. Therefore, the correlation coefficient is expressed as follows:

$$\begin{aligned} \text{Profile-based HD} &: (\mathbf{A}' \cdot \mathbf{X}', \mathbf{S}'), \\ \text{Profile-based XOR} &: (\mathbf{A} \cdot \mathbf{X}', \mathbf{S}'). \end{aligned}$$

Here, \mathbf{X}' and \mathbf{S}' can be expressed as

$$\mathbf{X}' = \begin{pmatrix} \mathbf{x}'_1^1 & \mathbf{x}'_2^1 & \dots & \mathbf{x}'_{n'}^1 \\ \mathbf{x}'_1^2 & \mathbf{x}'_2^2 & \dots & \mathbf{x}'_{n'}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}'_1^{16} & \mathbf{x}'_2^{16} & \dots & \mathbf{x}'_{n'}^{16} \end{pmatrix},$$

$$\mathbf{S}' = (s'_1{}^t \ s'_2{}^t \ s'_3{}^t \ \dots \ s'_{n'}{}^t).$$

4 Evaluation of the Amount of Information Leakage for Each Model

4.1 Experimental Setup

Figure 3 shows the framework for the performance evaluation of the models. The DE0-Nano Field Programmable Gate Array (FPGA) board [15] equipped with ALTERA Cyclone IV FPGA is used as the prover. The 1000-round AES is implemented in the FPGA. In the verifier, \mathbf{A} and \mathbf{A}' are derived in advance according to the method written in Sect. 3. The profile-based models $\mathbf{A} \cdot \mathbf{X}'$ and $\mathbf{A}' \cdot \mathbf{X}'$ are created from the challenge and all the pre-registered secret keys. The correlation coefficients are calculated with the profile-based model and observed side-channel information. Then, from the set of correlation coefficients, the average and variance are derived. The model with large average and small variance is determined to be a good performance model.

In this paper, 10,000 secret keys are pre-registered in the verifier. We assume a situation where each of the 10,000 legitimate provers are authenticated only once. For one authentication, 10,000 models are created. In order to authenticate the prover, the authentication system needs to calculate the correlations for 10,000 trials. The derived correlation coefficients for one authentication were classified into one acceptance trial and 9999 rejection trials. The acceptance trial is defined as the derivation of the correlation coefficient when it is equal to the secret key implemented in the prover and used to create the model. In contrast, the rejection trial is defined as the derivation of the correlation coefficient in the case of different secret keys. Therefore, in the authentication of 10,000 provers, it is necessary to derive the correlation coefficients for 10,000 · 10,000 trials.

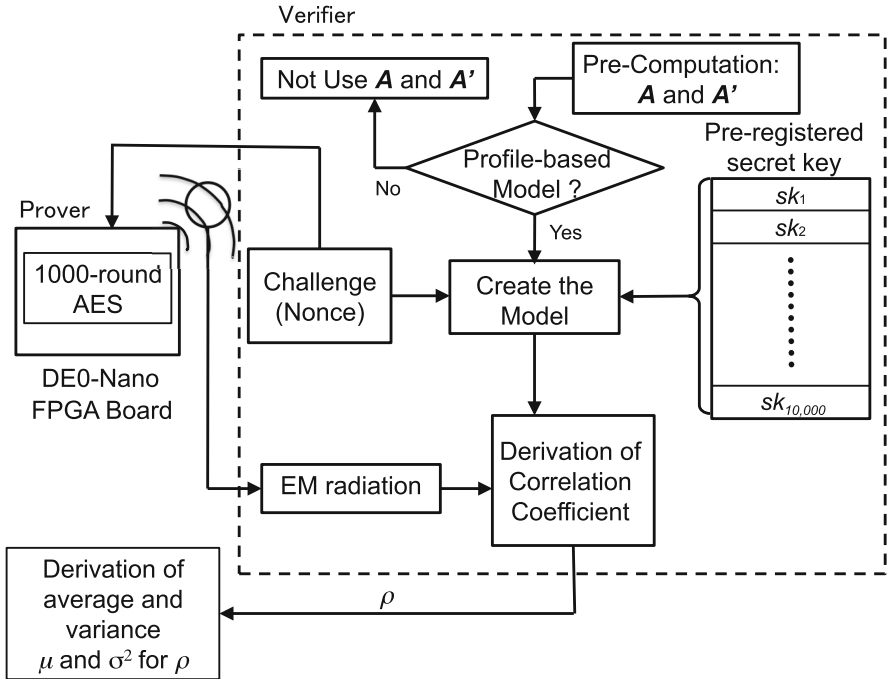


Fig. 3. Framework of performance evaluation for the models

4.2 Performance Evaluation of Each Model

The performance of the models is evaluated using non-profile- and profile-based HD models and profile-based XOR model. From the calculated correlation coefficients, the average and variance are derived for each trial. The results are listed in Table 3. The average and variance for acceptance and rejection trials are denoted as μ_1 , σ_1^2 , μ_2 , and σ_2^2 , respectively. In the rejection trial, the μ_2 and σ_2^2 were nearly the same for each model. In addition, the histogram of correlation coefficient in the rejection trial can approximate a normal distribution. Using the Jarque–Bera test, we confirm that the correlation coefficient can be normalized. Meanwhile, in the acceptance trial, μ_1 and σ_1^2 show different results. Compared to the non-profile-based model, the average and variance of the profile-based model were large and small, respectively. The average of the profile-based XOR model was 0.08 higher than that of the profile-based HD model, and the variance was 1.3×10^{-4} smaller. Therefore, when authentication is performed with the XOR model compared to the HD model, the probability of false positives and/or negatives decrease. Figure 4 shows the comparison between the histograms of the HD model and the XOR model. The model with the best performance is the profile-based XOR model.

Table 3. Average and variance for each model

	μ_1	σ_1^2	μ_2	σ_2^2
Non-profile-based HD model	0.487	6.13×10^{-4}	0.000	0.001
Profile-based HD model	0.537	5.63×10^{-4}	0.000	0.001
Profile-based XOR model	0.615	4.37×10^{-4}	0.000	0.001

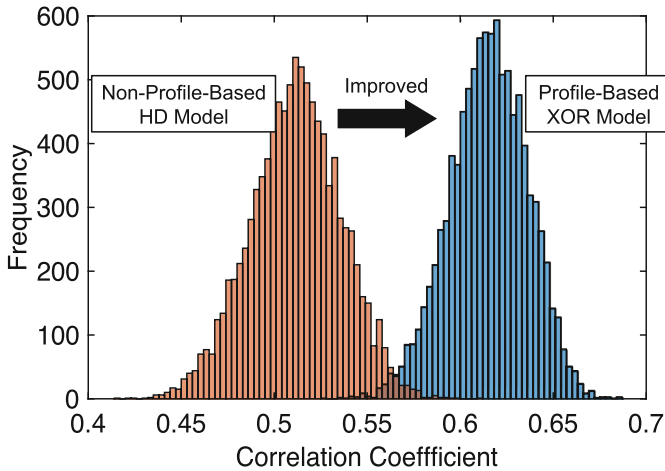


Fig. 4. Comparison of histogram between HD model and XOR model in acceptance trial

5 Conclusion

We evaluated the performance of profile- and non-profile-based HD models and a profile-based XOR model. The performance of the models was evaluated by comparing the average and variance in correlation coefficients calculated from the EM radiation and each model. The XOR model showed the best performance among the three models. The results confirm that more finely classified class and the profile-based model worked effectively in the authentication system. In addition, the profile-based model in the authentication system can be used in a realistic scenario.

Acknowledgments. This work was supported by Japan Society for the Promotion of Science (JSPS) Grants-in-Aid for Scientific Research (KAKENHI) Grant Numbers 15K12035.

References

1. Francillon, A., Danev, B., Capkun, S.: Relay attacks on passive keyless entry and start systems in modern cars. In: Network & Distributed System Security, NDSS. The Internet Society (2011)
2. Global trade in fake goods worth nearly half a trillion dollars a year - OECD & EUIPO. <http://www.oecd.org/industry/global-trade-in-fake-goods-worth-nearly-half-a-trillion-dollars-a-year.htm>. Accessed 8 Aug 2017
3. Ravikanth, P.S.: Physical one-way functions. Ph.D. thesis, Cambridge, MA, USA (2001). AAI0803255
4. Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: Proceedings of the 44th Annual Design Automation Conference, DAC 2007, pp. 9–14. ACM (2007)
5. Sakiyama, K., Kasuya, M., Machida, T., Matsubara, A., Kuai, Y., Hayashi, Y., Mizuki, T., Miura, N., Nagata, M.: Physical authentication using side-channel information. In: Proceedings of the 4th International Conference on Information and Communication Technology, ICoICT 2016 (2016)
6. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44709-1_21
7. De Mulder, E., Örs, S.B., Preneel, B., Verbauwhede, I.: Differential power and electromagnetic attacks on a FPGA implementation of elliptic curve cryptosystems. *Comput. Electr. Eng.* **33**, 367–382 (2007)
8. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
9. Agrawal, D., Baktir, S., Karakoyunlu, D., Rohatgi, P., Sunar, B.: Trojan detection using IC fingerprinting. In: 2007 IEEE Symposium on Security and Privacy, SP 2007, pp. 296–310 (2007)
10. Brier, E., Clavier, C., Olivier, 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). https://doi.org/10.1007/978-3-540-28632-5_2
11. Clavier, C., Danger, J.-L., Duc, G., Elaabid, M.A., Gérard, B., Guilley, S., Heuser, A., Kasper, M., Li, Y., Lomné, V., Nakatsu, D., Ohta, K., Sakiyama, K., Sauvage, L., Schindler, W., Stöttinger, M., Veyrat-Charvillon, N., Walle, M., Wurcker, A.: Practical improvements of side-channel attacks on AES: feedback from the 2nd DPA contest. *J. Cryptogr. Eng.* **4**(4), 259–274 (2014)
12. Brands, S., Chaum, D.: Distance-bounding protocols. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 344–359. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48285-7_30
13. Drimer, S., Murdoch, S.J.: Keep your enemies close: distance bounding against smartcard relay attacks. In: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, SS 2007, pp. 7:1–7:16 (2007)
14. DPA Contest v2 Hall of Fame. <http://www.dpacontest.org/v2/>. Accessed 8 Aug 2017
15. DE0-Nano Development and Education Board. <http://www.terasic.com.tw/en>. Accessed 8 Aug 2017



Holistic Tracking of Products on the Blockchain Using NFC and Verified Users

Vanesco A. J. Boehm^(✉), Jong Kim, and James Won-Ki Hong

Department of Computer Science and Engineering,
Pohang University of Science and Technology, 77 Cheongam-ro, Nam-gu,
Pohang 790-784, Korea
{boehmv, jkim, jwkhong}@postech.ac.kr

Abstract. Tracking the history of products and its parts has become a common way to detect many incidents of counterfeit. However, once a product leaves the manufacturing process, reliably capturing changes that a product undergoes becomes even more challenging. Anti-counterfeit solutions using blockchain technology promise several benefits. For example, updating the blockchain is only possible if a transaction is signed with the correct private key. To assure the confidentiality of such a key, it can be stored on an NFC tag in a way that only the tag can read it. By incorporating such a tag into a product, anyone possessing the product can connect with it and transactions for the blockchain can be signed by the tag. To regulate which kind of updates a user can perform, we suggest that users must be verified and to update a product's history on the blockchain valid user credentials must be provided. This way various actors such as service providers or authorities can be enabled to report changes of a product to the blockchain and the capabilities of other user groups can be restricted at the same time. As a result, holistic tracking of products throughout their lifespan can be achieved.

Keywords: Blockchain · Counterfeit detection · NFC · Supply chain
Product history tracking

1 Introduction

The complexity of the manufacturing processes of electronic goods with supply chains spanning numerous countries makes detecting counterfeits a major challenge. As a consequence, tracking goods by using RFID tags has become a common practice. However, as RFID-based systems are also limited to resolve the counterfeit issue [1]. For example, RFID tags can be cloned and it is still difficult to prevent that counterfeit parts are introduced in the supply chain.

Recently, blockchain technology is regarded by many researchers as a promising mean to achieve high data security and it proves also suitability for systems to track the provenance of goods on a supply chain. Block Verify is an example of a commercial blockchain-based anti-counterfeit solution and realized a proof-of-concept of a tracking solution running on the Ethereum blockchain [2, 3].

Once a product leaves its production process, the difficulty of tracking its status increases and reliably checking the authenticity of a product is very hard to achieve for a customer. For example, tracking IDs can be reused for clone products or an attacker may lead the verifier to a fake website which would wrongly declare a counterfeit to be genuine.

An interesting NFC-based counterfeit detection scheme was introduced, which allows anyone to verify the authenticity of a product without requiring access to a central database [4] and recently, some companies started using NFC together with blockchain technology. For example, the fashion brand, Babyghost, has incorporated NFC chips in some of their garments [5]. By using an NFC capable device and connecting with the NFC tag, people can query information about the product. Furthermore, customers can store further data on the chip to personalize the item when giving it to someone as a gift.

In this paper, we propose a system that allows holistic tracking of products from their production until their disposal. We suggest tracking the provenance of products on the blockchain and to allow certain authorized entities to make further updates even once a product has left the realm of the manufacturer. We aim to achieve this by managing the privileges of relevant actors in a separate database and incorporating NFC tags in products to enable the creation of signatures for transactions to update the blockchain.

The paper is organized as follows. In Sect. 2, some background and relevant technologies as well as related research work are introduced. In Sect. 3, our proposed scheme is explained along with requirements and design. In Sect. 4, a discussion of practicability related aspects of our proposed system will be given. Finally, concluding remarks and future work will be given in Sect. 5.

2 Background

2.1 Blockchain and Smart Contracts

In this subsection, the basic concepts of blockchain technology and smart contracts are described.

A blockchain can be described as distributed database with a transparent and immutable state [6]. A ledger being a register of transactions is the basis of a blockchain. These transactions are grouped as blocks being added consecutively like a chain. Only transactions being approved by a majority of the network are valid.

The state of the ledger is replicated on many network nodes. The first block is called genesis and does not refer to a previous block. All other blocks however know the cryptographic hash of their predecessor. Users wanting to make transactions require a private and public key pair. A transaction is only valid if it is signed with the correct private key. Using public keys, user accounts are uniquely identifiable. This allows transferring to other accounts. Due to asymmetric cryptography authentication, integrity and non-repudiation are achieved for transactions.

The creation of new blocks is an infinitely repeating process: first signed transactions are communicated to one-hop neighbors via broadcast. Only valid transactions are

communicated further. Transactions are then processed into a timestamped candidate block. This packaging process is called mining. Finally, the block is added to the chain if it passes verification. Otherwise it is discarded.

Logic to determine the validity of transactions runs on every node. Security is achieved since every transaction refers to a public key and only the owner of the corresponding private key can commit it.

The mechanism to reach consensus about the validity of transactions requires special arrangements. In the case of the bitcoin blockchain nodes have to solve mathematical complex problems to be eligible to take part in the verification process. This proof-of-work method limits the power of single nodes, so that faking transactions is unlikely to be possible as opposed to a centralized database owned by only one party. Since only the fastest growing chain is accepted as the valid version of the blockchain, chances that an attacker successfully modifies a block diminish with every new block [7].

A smart contract is a digital transaction protocol performing the contents defined in the contract [6]. Procedures are linked to unique addresses and only authorized entities can trigger them. Smart contracts are especially suitable to trade digital assets. It is crucial that business logic to handle all possible outcomes of a smart contract is implemented.

Ideally blockchain-based solutions are tolerant to node failures, have a single, universally accepted state of events, provide transparency, verifiability, and auditability. Moreover, smart contracts require no central authority and are a mean that allows interaction of non-trusting participants in a predefined, secure way.

2.2 Near Field Communication

Near Field Communication (NFC) is interoperable with contactless smart cards of ISO/IEC 14443 standard [4]. NFC can transmit information wirelessly within a distance of four centimeters. Although NFC is easy to use because reading from a tag only requires touching it with a reading device, its applicability is rather limited in the context of supply chain. If a product is wrapped in packaging, connecting with the NFC tag might not be possible any more. NFC technology is a common feature in most modern mobile phones.

2.3 Risk of Counterfeit on the Consumer Electronics Supply Chain

According to the study on counterfeit electronics conducted by the U.S. Department of Commerce, *'a counterfeit is an electronic part that is not genuine because it:*

- *is an unauthorized copy*
- *does not conform to original manufacturer's design, model, and/or performance standards*
- *is not produced by the original manufacturer or is produced by unauthorized contractors*
- *is an off-specification, defective, or used product sold as "new" or working*
- *has incorrect or false markings and/or documentation' [8].*

In line with this definition, the survey [9] done by the European Commission introduces a taxonomy of counterfeit electronic products differentiating between cloned, overproduced, out of spec/defective, recycled, and tampered goods.

According to [10], the consumer electronics supply chain starts with the manufacturing of wafers, which are required for producing chips. These chips can be incorporated in boards being part of final systems. Products reaching the end consumer are in the “*after-market, sales and refurbishment*” stage until they get disposed or recycled.

Design stages of a product or its parts are also important to consider when examining the risk of counterfeit. For example, [11] describes hardware Trojans being altered chips with additional behavior which will be triggered under certain conditions to leak information or damage a system. It is especially difficult to detect counterfeit when it is introduced already at a product’s design phase since the counterfeit will become part of the signature of the original product [12]. As mitigation only signatures provided by trusted institutions (e.g., standardization organizations) should be used to develop counterfeit-prone products. Moreover, there are many incidents of products reaching their end-of-life stage being reintroduced in the supply chain as counterfeit and there is a high risk that components get tampered [13].

In general detection of counterfeit in early stages of a product’s manufacturing process is usually simpler than detecting in later stages. Especially, methods to detect counterfeit components in assembled products require costly equipment and specially-trained people being able to use it [9]. The downside of testing early is that modifications happening in later stages are not covered.

3 Proposed Tracking Solution

3.1 Requirements

To mitigate the risk of counterfeit for consumer electronics, a mechanism allowing to thoroughly track a product’s history throughout its lifespan is required. Ideally, anyone should be able to determine whether a given product is genuine or not in an easy manner.

To achieve broad acceptance high usability, hiding of any technological complexity and granting anonymity of users are crucial factors. Further, the solution should address the needs of different stakeholders such as consumers, dealers, and authorities. At the same time, only authorized entities should be able to update information of a product.

There should be full transparency regarding the data of products and censorship or illegitimate modifications should be impossible. Overall, the system should be secure, reliable, and easily accessible.

3.2 Analysis

Blockchain-Based Tracking. Many incidents of counterfeit can be eliminated by using a system to track products. Furthermore, many of the aforementioned requirements can be met by employing blockchain technology. In this work, the blockchain-based

approach of [3] shall exemplify a solution allowing to trace the complete manufacturing history of products and their components. The basic idea of [3] is to represent the provenance of goods by following an ontology concept.

The goal when employing blockchain technology to realize such a system is to have one central, distributed database serving all participants of the supply chain. However, this is only possible when all partners interpret the data in a common way.

[3] implemented a proof-of-concept for their suggestion as smart contract for the Ethereum blockchain. Using an HTML 5 browser interface, it is possible to query traces of a product. Their solution seems to be practical and effective to detect counterfeit while products are in the production process. However, once produced goods are not possessed by the manufacturer any more e.g. when being shipped to distributors, stores or customers continuance of reliable tracking may not be feasible and the risk for abuse increases.

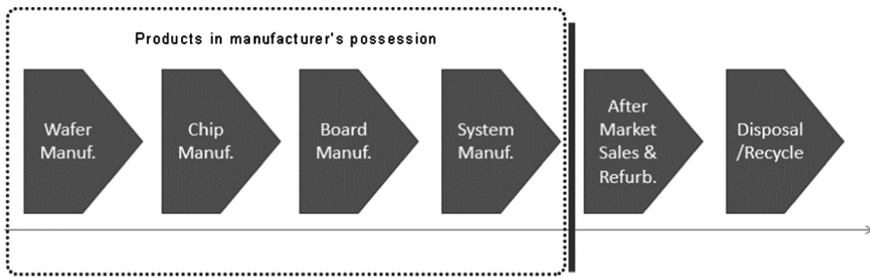


Fig. 1. Manufacturer's sphere of influence on the supply chain.

Figure 1 shows that in the last two supply chain stages the manufacturer's influence to track products can be expected to be rather low, since products are not in their possession any more. The difficulty of tracking a product's status and verifying its authenticity becomes then very high with the solution proposed by [3]. For example, if a distributor or merchant does not update a product's history any more, it may be infeasible for an end consumer to determine the authenticity of a good. In such a case, a clone using the serial number of a genuine product would show a valid production history on the verification website. This issue can be mitigated if customers register their products. Then the manufacturer can detect double registrations. However, in case the serial number of a product gets registered more than once, it would still not be clear which is the original and which are clones. Furthermore, counterfeit products may lead to a fake website which would falsely verify any product to be genuine.

To achieve complete tracking of products even after their production, reliable contribution of further actors seems to be required. As a consequence, integration of such entities is necessary to enable them to update the history of products on the blockchain. However, making a blockchain transaction requires the transaction to be signed with the correct private key. Sharing private keys of products with all relevant parties such as merchants, customers etc., entails an enormous risk that keys come into

property of an adversary. Furthermore, since different people would use the same key to sign transactions, traceability of who did an update would get lost. On the other hand, the privacy of individuals should nonetheless be protected.

NFC-Based Verification Scheme. To help solving the previously described dilemma this research suggests enhancing the blockchain-based approach with an NFC-based counterfeit detection solution such as introduced by [4].

The idea of [4] is to incorporate in addition to an electronic product code (EPC) tag used for RFID-based tracking of products during the manufacturing process an NFC tag in final goods. Their scheme allows checking the genuineness of a product with the help of an NFC capable smart device without the need to query any other system.

An independent trusted entity is required whose public key will allow to verify the authenticity of a product. This entity is called the tag initiator. Initially, the manufacturer is supposed to send the serial number, EPC, and product specification of a good being manufactured to the tag initiator. The tag initiator generates a public/private key pair for the product, stores the EPC in an EPC tag and creates a string by concatenating the EPC, the product's serial number, the product specification, and the generated public key. The tag initiator's signature of this string created with her/his private key gets stored together with the string on the NFC tag. This data is accessible for any NFC reader. The assigned private key, however, is stored at a secure location of the tag which can only be read by the tag's processor. Once the values are stored, the tag is set to read-only. Finally, the tag gets physically integrated in the final product. Any attempt to remove the tag shall obviously damage the product.

Verifying the genuineness of such an equipped product is done in two steps. First, the verifier compares the identity of the product with its product description. Further, with the help of a smart device the verifier reads the information stored on the NFC tag to check whether it matches with the product at hand and the smart device assesses the validity of the signature with the help of the tag initiator's public key.

The second step consists of a challenge-response protocol. The device for reading the NFC tag sends a random challenge r to the tag. The tag signs r with the secret key stored at a location being inaccessible for reading devices and returns the information. Finally, the verifying device can evaluate the correctness of the signature with the public key which it learned from the NFC tag.

3.3 Proposed Design

To fulfill the requirements described in 3.1, we propose blockchain-based tracking of products with the mechanism of NFC tags carrying a product's private key to allow entities possessing the product to trigger further updates and to verify the authenticity of the product offline. Additionally, verified user accounts to grant permissions depending on the kind of entity are proposed to complement the solution.

Similar as in the NFC-based scheme described in [4], manufactured products must be equipped with an NFC tag storing a private key that only the processor of the tag can read. In addition to verifying the authenticity of a product, this key shall serve to sign transactions to update information of this product on the blockchain. However, to only allow authorized entities to perform certain updates, a user account verified by a trusted

third party is required. To assign a privileged role to an account, some verification must be done. For example, an entity could be verified as a dealer by conveying a verification code to the applicant via the store’s officially registered phone number. Verifying users via a phone number is common on many social network applications and it is also a popular method to recover an account in case of a lost password.

In any case, just reading data of a product should be always possible without any special permission as it is transparent on the blockchain anyway. To register a product by an end consumer a registration code should be shipped with the product. This way only the owner of the registration code can register the product as it is common, for example, for software products.

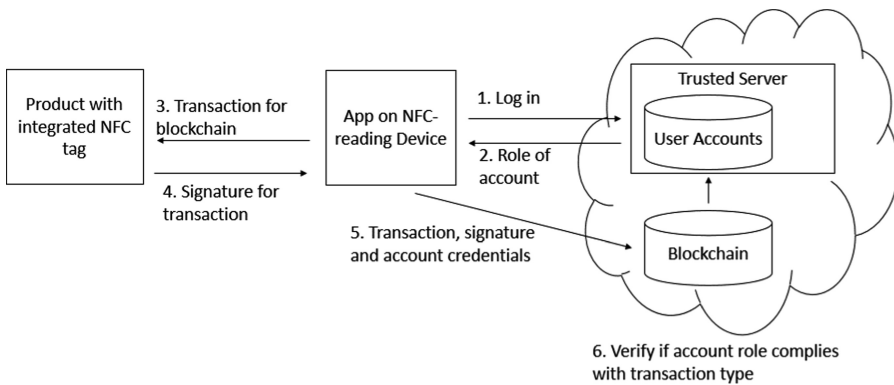


Fig. 2. Proposed design of a blockchain based tracking solution using NFC technology and a user account database.

Figure 2 illustrates the procedure to update data of a product on the blockchain. Initially users are supposed to log into the corresponding app on an NFC capable device. The available actions in the app that a user can perform depend on the user’s role being inquired from the user account database. When a user selects an action to update data on the blockchain, the necessary transaction gets signed with the private key being stored on the NFC tag. Then the transaction, its signature as well as the credentials of the user’s account get transmitted to a blockchain client. The blockchain client will verify whether the transaction’s signature is valid and if the credentials of the user match with an account being authorized to initiate the desired action.

The log-in procedure could be omitted and a user could just choose among all possible actions. The eventual success of a transaction is determined by the verification of the blockchain client which expects credentials of an account with the necessary privilege. To improve the security only a hash value of a user’s account password should be sent to the blockchain node and compared with the hash value of the account’s password being stored in the account database.

To achieve anonymity of users only a unique account number shall appear in the data record on the blockchain. This account number could however be used by authorized bodies to investigate who triggered a transaction for example when misuse is suspected.

As mentioned earlier, different kind of users shall be able to perform different actions depending on their role.

Table 1. Examples of possible roles and corresponding capabilities.

Type of role	Examples	Capabilities
Owner	End consumer	Register product Inquire product history
Service provider	Certification body; repair shop; store	Describe product status, e.g. damaged, repaired, recycled Describe product’s quality e.g. by providing test results Inquire product history
Authority	Customs clearance; law enforcement	Describe product status, e.g. damaged, recovered, kept Inquire product history
Carrier	Shipping company	Describe delivery status Inquire product history

Table 1 shows different roles which can be relevant for the proposed system. Independent of the role, anyone can read the product history without any special permission. The rightful owner of the product is most probably the end consumer. The owner can register the product with the registration code. A service provider such as a dealer or certifier can update information related to the state of the product. For example, if a product got repaired or is damaged such information should be stored on the blockchain. Furthermore, test results of a product as well as the fact that a product is supposed to get recycled marking the end of a product’s lifespan should be recorded. Also, certain authorities should have the power to make statements about the state of a product. It might be that a product was stolen and could be recovered or damage due to vandalism might have occurred. Additionally, it may be useful to track on the blockchain that a product is kept at the customs clearance and carriers should be able to add information about a product’s delivery status. In this case, however, the packaging of the product must permit connecting with the NFC tag.

The assignment of privileged roles to accounts should have an expiry date, so that renewal of user roles can be enforced in certain intervals. In this way, higher confidence can be achieved that a registered entity still meets the requirements qualifying her/him to make certain updates. The owner role is an exception, since it should be the default role which does not need any special verification procedure. Also, the owner role does not allow a user to perform any updates on the blockchain except registering the product which requires the registration code having been shipped with the product.

User Accounts Associated with Registered Products. Recording that a product got lost or stolen is a use case requiring special consideration. Since the actual product is needed to sign transactions to update its data on the blockchain, this is impossible if the product is absent. To settle this issue, the user account of the owner who registered the product could be queried to obtain information about the product's status. This data can simply be maintained by the owner. However, it should be made evident for people checking the status of a product that this piece of information is neither verified nor stored on the blockchain and just a statement of the owner which may be false.

To assure that the rightful owner of a product is always capable to take advantage of this privilege a mechanism is required that in case the owner of an already registered products changes, the product registration can be transferred to the user account of the new owner.

4 Practicality Aspects

The proposed system enables authorized entities to update a product's history on the blockchain. This solution does not require to share a private key to sign a transaction with anyone. Instead, to obtain the signature for a transaction, the actual product which functions as a token is needed. Additionally, the complexity of the blockchain technology is completely hidden from users as they only have to use an application on an NFC capable device, select their desired option and provide credentials of an eligible user account. Initially, creating a new account to use the system is a procedure with which any smartphone user should be familiar.

The proposed solution assures confidentiality of personal information due to the separation of product related data being stored on the blockchain and user information stored in a proprietary database. The security of personal data however depends on the security and trustworthiness of the server where the database is hosted.

Especially, the additional overhead and costs are drawbacks of the suggested system. NFC chips must be bought and built in every product. This requires a strong partnership with the NFC tag provider, since the tag provider's certificate is required when someone wants to check the authenticity of a product by using the scheme introduced in [4]. Further, the entity managing the user accounts has to be a reliable partner of the manufacturer. Consequently, the manufacturer's dependency of other service providers increases. Updates on the blockchain are only possible if the account database is online. Thus, this database can be a bottleneck and single-point-of-failure of this system. This risk however is slightly mitigated since writing to the blockchain can be expected to happen less often than reading its data and the reading procedure does not require the account database, except if one wants to check the product status set by the owner who has registered the product.

The greatest weakness of this solution is probably that the effectiveness of the system relies on the correctness of the updates that people perform. People being authorized to record certain information of a product on the blockchain can either accidentally or intentionally make wrong updates. In any case, if a false statement appears on the blockchain, the related user account is of course identifiable and the account owner can be held liable. If an attacker gets possession of account credentials

the risk that she/he can make fraudulent updates on the blockchain is comparatively low, since she/he also requires an original product to sign a transaction with the help of the integrated NFC tag.

If people reliably use the suggested system and continuously update the history of a product throughout its lifespan plus if the authenticity of products can be checked with the scheme of [4], many types of counterfeit can be mitigated. Reuse of product ids for clones or overproduced products is not possible anymore. If products are expected to get recycled, but resold with false labeling, it can be detected as long as the corresponding product record is up-to-date. Similarly remarked and out of spec or defective products can be identified when checking their product history on the blockchain. Furthermore, products which were obtained illegally can be identified easily. Especially when receiving a product being already registered and whose status is set to stolen or lost by the owner some criminal activity may have occurred.

However, with the proposed system, it is impossible to discover whether a product got tampered. This requires other methods which could be either build-in mechanisms allowing an electronic device to autonomously detect any modifications of it or it may be an evaluation done by experts with specialized tools and methods. Of course, any insights gained due to such an investigation should ultimately be captured on the blockchain.

To obscure data of a product stored on the blockchain an attacker can still try to lead a victim to a false verification website or make her/him use a fake app which would show a wrong product history. Due to the NFC-based verification scheme of [4] the verifier would get assured that she/he possesses an original product. As a result, she/he may not detect fraud related to the product information, if she/he assumes that this data is queried from the blockchain and thus should be trustworthy.

Although it is possible to employ such a proposed system for many kind of different products, this research focuses on the special case of consumer electronics. Electronic devices are assembled of many parts. So, it may be comparatively easy to consider the additional integration of an NFC tag when designing the product. Especially for high-priced premium goods the extra costs may be acceptable, since the relative increase is rather neglectable and consumers in this segment might be also willing to pay a little surplus if this gives them greater assurance that they own an original product.

Similarly, the usage of NFC can be justified. Since NFC is common in many modern portable smart devices, the proposed approach may face broad acceptance. A shortcoming of NFC is however its short reach. Therefore, its applicability in automated scenarios is limited, because machines attempting to connect with an NFC tag would have to be positioned very precisely.

5 Conclusion and Future Work

This research suggests a system based on NFC tags incorporated in products to allow updating the history of produced goods on the blockchain, even once products have left their manufacturing process. This concept settles the difficulty of distributing keys to enable eligible entities to sign transactions, since every NFC tag knows the necessary

secret key and can sign transactions with it. The combination of blockchain technology and separately hosted user accounts guarantees both anonymity of users and that only authorized entities can make certain updates. For a transaction to become valid credentials of a user with the necessary permission are required. Initially, an owner of a user account must be verified to grant any special permissions.

All in all, such a tracking solution is capable to capture a product's provenance including any changes holistically. As a result, many types of counterfeit can be combated. Only the detection of tampering requires other mechanisms such as self-inspection of a product or expert assessment. Furthermore, due to the NFC-based verification adopted from [4] high confidence about the genuineness of a product at hand can be achieved.

The benefits of the described solution come with additional overhead and costs. NFC tags must be incorporated in every product and an infrastructure to manage user accounts, verify entities and monitor that no user having special permissions abuses her/his power are required. Assessment of the feasibility of such a system by implementing a prototype has not been done yet and remains an open task.

Since the overall effectiveness of the solution depends on the reliability of its users, concepts to limit the power of authorized entities might further improve the introduced system. For example, certifiers or stores could use test devices with an integrated secret user account password similar to NFC tag's secret key. When such a test device checks a product, it could automatically trigger a transaction to update the blockchain with the test results by authorizing itself with its account password. The ultimate solution might be products which can autonomously track their history and report any changes they undergo to the blockchain.

References

1. Bilal, Z., Martin, K.: A hierarchical anti-counterfeit mechanism: securing the supply chain using RFIDs. In: Danger, J.-L., Debbabi, M., Marion, J.-Y., Garcia-Alfaro, J., Zincir Heywood, N. (eds.) FPS 2013. LNCS, vol. 8352, pp. 291–305. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05302-8_18
2. Nofer, M., Gomber, P., Hinz, O., Schiereck, D.: Blockchain. *Bus. Inf. Syst. Eng.* **59**(3), 183–187 (2017). <https://doi.org/10.1007/s12599-017-0467-3>
3. Kim, H.M., Laskowski, M.: Towards an ontology-driven blockchain design for supply chain provenance. *SSRN Electron. J.* (2016). <https://doi.org/10.2139/ssrn.2828369>
4. Saeed, M.Q., Bilal, Z., Walter, C.D.: An NFC based consumer-level counterfeit detection framework. In: 2013 Eleventh Annual Conference on Privacy, Security and Trust (2013). <https://doi.org/10.1109/pst.2013.6596047>
5. Brown, C.: Fashion brand adds NFC and blockchain for authentic personal clothing (2016). <https://www.nfcworld.com/2016/11/14/348460/fashion-brand-adds-nfc-blockchain-authentic-personal-clothing/>
6. Christidis, K., Devetsikiotis, M.: Blockchains and smart contracts for the internet of things. *IEEE Access* **4**, 2292–2303 (2016). <https://doi.org/10.1109/access.2016.2566339>
7. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008)
8. U.S. Department of Commerce – Bureau of Industry and Security (2010). Defense Industrial Base Assessment: Counterfeit Electronics, January 2010

9. Baldini, G., Fovino, I.N., Satta, R., et al.: Survey of techniques for the fight against counterfeit goods and Intellectual Property Rights (IPR) infringement. European Commission. Joint Research Centre (2015)
10. Huehne, M., Lee, J.C., Miles, H., Schaffer, M.: Methodology development for counterfeit component mitigation. In: 2013 14th International Conference on Electronic Packaging Technology (2013). <https://doi.org/10.1109/icept.2013.6756645>
11. Tehranipoor, M., Salmani, H., Zhang, X.: Integrated Circuit Authentication: Hardware Trojans and Counterfeit Detection. Springer, Cham (2014). <https://doi.org/10.1007/978-3-319-00816-5>, ISBN 978-3-319-00815-8
12. McFadden, F.E., Arnold, R.D.: Supply chain risk mitigation for IT electronics. In: 2010 IEEE International Conference on Technologies for Homeland Security (HST) (2010). <https://doi.org/10.1109/ths.2010.5655094>
13. Tehranipoor, M.M., Guin, U., Forte, D.: Counterfeit Integrated Circuits: Detection and Avoidance (2015). ISBN: 978-3-319-11823-9

Attack and Defense II and Network Security



Abusing TCP Retransmission for DoS Attack Inside Virtual Network

Son Duc Nguyen^(✉), Mamoru Mimura, and Hidema Tanaka

National Defense Academy of Japan, Yokosuka, Japan
{ed56037,mim,hidema}@nda.ac.jp

Abstract. Among DoS attack techniques, abusing UDP-based public servers like DNS or NTP for reflective amplification attack is continued to pose a great threat. Recent studies show that attacker can also use TCP retransmission before the three-way-handshake completion to perform this kind of attack. In this paper, we focus on the virtual environment, in which we evaluate the potential of abusing the virtual switch system to perform amplification attack. We created a virtual network that able to connect to an external network and observed the virtual switch system's behavior while receiving TCP packets from outside the network. We show that the virtual switch system itself can retransmit TCP packets and therefore can be abused for amplification attack by an internal attacker. In other words, he/she can make amplification using TCP hosts from outside the network and the virtual switch system's retransmission ability. Furthermore, we test the endurance of different OS and show that Windows OS family and macOS are more vulnerable than Linux Ubuntu OS against this kind of attack.

Keywords: Virtual network · Amplification attack
TCP retransmission

1 Introduction

1.1 Background

Virtualization. Recently, virtualization has become a mainstream IT architecture with high abilities in memory, storage and processing power management. It provides the capability to easily move virtual servers between different physical servers to balance demand for resources. This can dramatically reduce the number of physical servers in the data center. Furthermore, virtual machines are much easier to set up and break down. If an application needs a new server, an administrator can provide the resources much faster than setting up a physical server. With these benefits, many companies and institutions are adopting virtual server in their system and deploying all new applications in the virtualized environment. Virtualization has opened the gate to *Cloud computing*, in which users and companies can process and store their data in virtual servers of third-party data centers. In 2013, it was reported that cloud computing had become a highly demanded service and a promising business [10].

An obvious downside to virtualization is the fact that multiple virtual machines are run on a host machine called a *hypervisor*. Therefore when the hypervisor falls down, all applications running on virtual machines will become unavailable. In a virtual network, all virtual machines Ethernet connection are managed by virtual network adapters and virtual switch. Virtual network adapters are software constructs that are responsible for receiving and transmitting Ethernet frames into and out of their assigned virtual machine or the management operating system. They are provided by a virtual switch, a software construct, which operates within the active memory of the hypervisor and performs Ethernet frame switching functionality. A virtual switch can use single or multiple physical network adapters to serve as uplinks to a physical switch in order to communicate with other computers on the physical network.

In this paper, we show a new threat scenario in which the virtual network is overloaded by making the virtual switch retransmit and amplify TCP packets that come from outside to virtual machines. Therefore, all virtual machines under that switch will be disconnected to the Internet as the result of the attack.

TCP Reflective Amplification Attack. TCP/IP (Transmission Control Protocol/Internet Protocol) protocol suite is a communication rule, which specifies how data should be transmitted [5]. TCP/IP uses a client/server model in which a computer (client) requests and is provided a service from another computer (server) in the network. In order to manage the data transmission, TCP/IP uses two main types of protocol, which are TCP and UDP (User Datagram Protocol). TCP ensures a reliable and ordered delivery of a load of packets from a client to server or vice versa, while UDP just sends them to another side of the connection. These features make TCP more reliable than UDP since it manages message acknowledgment and retransmissions in case of packet loss. However, UDP is suitable for applications that need fast transmissions.

Recently, UDP-based public servers such as DNS and NTP are known to be abused for reflective amplification attack. The attacker sends relatively small requests with a spoofed source address to the hosts that reflect significantly larger responses to the victim. As a result, it overloads and exhausts the capacity of the victim's network and makes normal client unable to access to the victim's server. This kind of attack is posing a serious problem because of its consequences, which cost the victim a very high cost to remediate the vulnerable systems.

On the other hand, TCP-based public servers are not known to be used for DDoS amplification attacks [3]. However, Kühner, Hupperich, Rossow and Holz show that a lot of TCP servers that can actually allow amplification [1,2]. This discovery opens a new threat to the Internet security since using TCP for transmitting data is more common in TCP/IP. In order to mitigate this kind of attack, it is important to identify hosts and devices that have amplification factor beforehand and make changes in their configuration.

Besides the threat coming from outside, cyber attacks can also happen from inside. In fact, IBM has pointed out that 60% of cyber attacks in 2015 were caused by insider threats [4]. As already shown above, in this paper, we con-

sider the threat of TCP-based reflective amplification attack inside a virtual environment, in which the virtual switch system itself can retransmitting TCP packets, that come from external TCP hosts, to virtual machines. An attacker can use this kind of attack to sabotage the entire virtual network or overload the virtual system from inside. To evaluate the potential of this attack, we used VMware Workstation [9] to create a virtual network environment that connects with our physical subnet through a virtual switch system. We observed different retransmitting behaviors of the virtual switch and enumerated any kind of TCP packets retransmitted. In most cases of our experiments, the virtual switch sent up to 300 SYN/ACK packets in an interval time of 30 s. In addition, we also identified that in some cases, the virtual switch retransmitted FIN/SYN/ACK packet, in which did not stop after the retransmission time-out. In order to evaluate the impact of this kind of attack on the virtual network, we did some experiments in which we could control the attack volume. The results show that this kind of attack can easily overload the virtual switch system, posing a new threat to any virtual network.

This paper has the following contributions:

- Discovery of a new type of TCP retransmission that can become an advantage for amplification attack.
- Evaluation of the ability of real world TCP-retransmission attack in the virtual network.
- Evaluation of the immunity of different operating systems from this kind of attack, showing Windows OS and macOS on default settings is more vulnerable than Linux Ubuntu.
- Introduction of a threat scenario, in which adversary can sabotage a virtual network from inside by abusing the host’s virtual switch system.

1.2 Related Work

Our work is based on the evaluation of TCP-based reflective amplification attack. Kühner *et al.* performed scans in IPv4 space for potential DDoS amplification hosts, which they labeled as *amplifiers*, and found out that some TCP-based hosts can be abused for amplification attack [1]. As the results, they focused on thirteen common TCP-based protocol and performed other scans to find those TCP amplifiers. They analyzed the amplifiers’ features and identified three main types among others [1].

- (a) SYN/ACK packets retransmission
- (b) Payload data transmission prior the handshake through PSH packets
- (c) Aggressive RST segment storms

During the scans, they did not respond any packets from hosts in order to make it easier to trigger the retransmission due to packet loss. Furthermore, they performed real-world TCP-based amplification attacks using a randomly chosen subset of hosts and gave an overview of countermeasures.

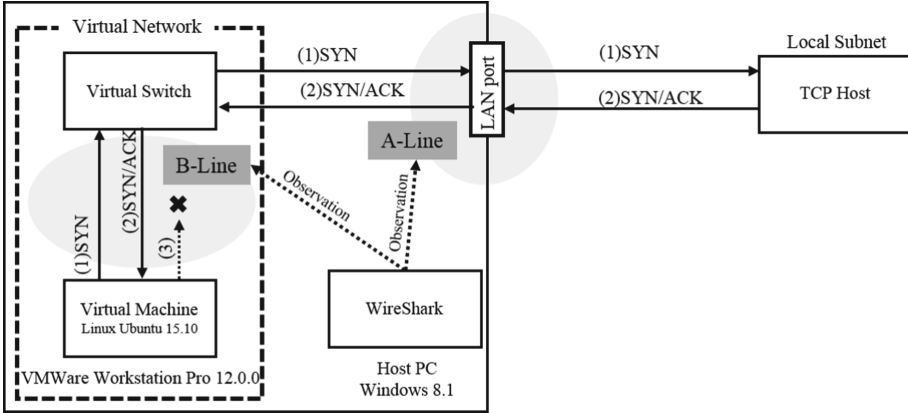


Fig. 1. TCP retransmitting packets capture experiment

2 TCP Retransmission from the Virtual Switch

2.1 Experimental Environment

First of all, we observe TCP retransmission in the virtual environment. Theoretically, there are nearly no restrictions between the virtual switch system and virtual machines. All the following experiments are performed using VMWare Workstation Pro 12.0.0 [9] with NAT setting for external connection and Linux Ubuntu 15.10 as a virtual machine (Fig. 1).

In order to make the virtual switch retransmit TCP packets, the internal attacker has to search for external TCP hosts that connect to the virtual network. In this experiment, we use Advanced IP Scanner and NMap’s TCP SYN scan on our local subnet to identify connecting TCP hosts [6, 7]. As a result, we found hundreds of open port with 124 types of protocol. In summarize, 579 pairs of ports/devices in our subnet that can be abused for our experimental attack. In this way, we confirmed that port scan from a virtual environment to an external network is feasible.

In reflective attack, if the victim responds to an unknown SYN/ACK packet with RST segment, the handshake will be canceled and no amplification will be made. However, Kühner shows some conditions for the victim having no responses. Detailed condition and analysis are shown in [1], we follow their condition to measure the best effectiveness of amplifiers (Fig. 1).

Step-1. From virtual machine, we send a single SYN packet ((1) SYN in Fig. 1) to each open port of TCP hosts in the local subnet.

Step-2. TCP hosts reply by sending SYN/ACK packet ((2) SYN/ACK in Fig. 1).

Step-3. In order to active TCP retransmission in the three-way-handshake, we set that virtual machine does not reply RST to TCP hosts ((3) in Fig. 1).

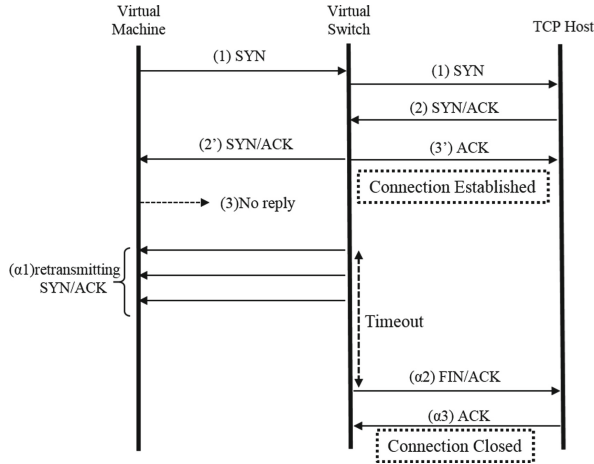


Fig. 2. SYN/ACK retransmitting communication

We observe communication situations in external (“A-line” in Fig. 1) and internal (“B-line” in Fig. 1) of virtual network in Fig. 1 using Wireshark [8] on the host PC. From some experiments, we can find followings:

- In A-line, the three-way-handshake is established between host PC and TCP hosts on external network. This three-way-handshake is ordinal, therefore no retransmissions are found in this communication.
- In B-line, we can observe many retransmitting SYN/ACK packets from virtual switch. This retransmission executes in a time-out interval.

2.2 Analysis of Transmission

Figures 2 and 3 summarizes the virtual switch behaviors that we could observe from the experiment. In A-line, we observed that the virtual switch sent the SYN packet ((1) in Fig. 2) using its own ports. After it received a SYN/ACK packet ((2) in Fig. 2) from the host, it automatically responded with its own ACK packet ((3') in Fig. 2). This procedure finishes the three-way-handshake and establishes the TCP connection between the virtual switch and the host. On the other hands, in B-line, the virtual switch redirect this SYN/ACK packet ((2') in Fig. 2) to the virtual machines. However, since the virtual machine did not respond to the SYN/ACK packet ((3) in Fig. 2), the three-way-handshake could not finish and the virtual switch retransmitted the SYN/ACK packet according to the TCP retransmission rule ((α1) in Fig. 2). In most cases, the retransmission stops after 30s since the first SYN/ACK packet was sent. We deduce that this is VMWare Workstation’s retransmission default setting. After stopping retransmit SYN/ACK packet to the virtual machine, we observed in A-line that the virtual switch sent a FIN/ACK packet to the hosts in order to end the established TCP connection

(($\alpha 2$) in Fig. 2). 119 out of 124 types of the experimental protocol have been found with this behavior.

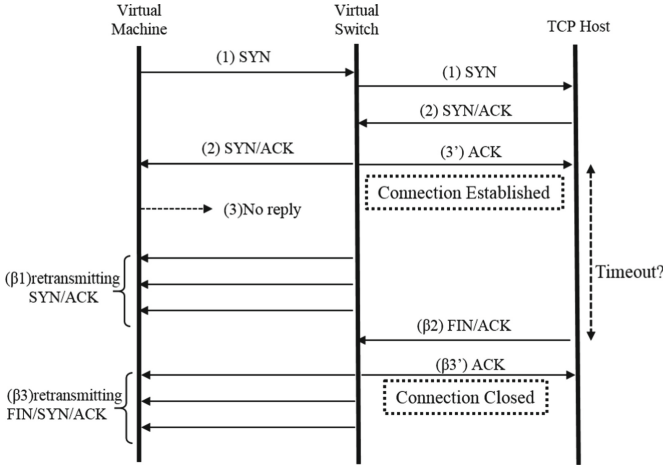


Fig. 3. FIN/SYN/ACK retransmitting communication

We did not find any behavior of payload data transmission before the handshake or RST segment storms from the virtual switch. On the other hand, in some cases, we observed that the virtual switch’s retransmission change from SYN/ACK packets to FIN/SYN/ACK packets. Such FIN/SYN/ACK is an abnormal packet which has the FIN, SYN and ACK flag set at the same time in the TCP header. Instead of stopping after 30s, this kind of retransmission does not stop until we sent an RST segment from the virtual machine to abort the connection. This event only occurs when the host decides to perform an active close and sends a FIN/ACK packet (($\beta 2$) in Fig. 3) before the virtual switch ends the TCP connection. The virtual switch subsequently inserts the FIN flag to the retransmitting packet, thus makes a SYN/ACK packet becomes a FIN/SYN/ACK packet (($\beta 3$) in Fig. 3). The FIN/SYN/ACK packet retransmission might not be set a time-out interval, which makes it continuously retransmit until receiving respond from the virtual machine. On the other hand, in A-line, the virtual switch responds to the host with an ACK packet (($\beta 3'$) in Fig. 3) and ends the connection. We found this behavior when performing experiments on the following ports: 1218, 1947, 3306, 3493 and 60002. Furthermore, some specified pairs of port/device show the same behavior. For example, port 21 and 80 of Canon’s Printer send a FIN/ACK packet before the virtual switch.

2.3 Analysis of Data Size

From the analysis of transmission (Sect. 2.2), the total amount of received data by virtual machine in estimated by retransmitting of SYN/ACK (($\alpha 1$) in Fig. 2 and

(β_1) in Fig. 3) and of FIN/SYN/ACK ((β_3) in Fig. 3). We define the amplification factor (AF) as follows:

$$AF = \frac{SA \times t_1 + FSA \times t_2}{S} \quad (1)$$

where each variable is shown in Table 1. In all cases, SYN packet size is fixed to 60 bytes. On the other hand, the sizes of SYN/ACK and FIN/SYN/ACK are various following the protocol. In most cases, 300 times of retransmitting of SYN/ACK packet with 58 bytes is found in our analysis. Already shown above, in some cases, retransmitting of FIN/SYN/ACK is occurred. Since retransmitting FIN/SYN/ACK continues infinite, we observed only within 60s since the first retransmission in our experiments. Table 2 shows only significant results (13 out of 579 ports/devices).

Table 1. Notations

AF	Amplification factor
S	SYN packet data size [byte]
SA	SYN/ACK packet data size [byte]
FSA	FIN/SYN/ACK packet data size [byte]
t_1	Number of retransmitted SYN/ACK packet ((α_1) in Fig. 2, (β_1) in Fig. 3)
t_2	Number of retransmitted FIN/SYN/ACK packet ((β_3) in Fig. 3)

Because there is a large difference between our experiment (virtual network) and real network, it is obvious that our result is different far from Kuhrer [1]. The differences are summarized as follows:

- An abnormal FIN/SYN/ACK packet retransmission behavior
- Retransmission that endlessly continues without a time-out interval
- In the real network, in most cases, the SYN/ACK retransmitting frequency is only 5 packets in 30s. The virtual network, on the other hand, can have a larger retransmitting frequency depends on their virtual switch’s setting. For example, in real network, FTP protocol retransmits with $t_1 = 5$ in 30s [1]. However, in our virtual network, it retransmits with $t_1 = 300$ (see Table 2).

As the results, we can conclude that, since there are big differences in behavior of retransmitting manner between outside and inside of the virtual switch system, this situation becomes a great vulnerability for virtual machine users.

2.4 Analysis of Different OS Default Response

In previous experiments, we used Linux Ubuntu 15.10 and set it not to reply any TCP packet. In order to use this attack on various virtual machines, we verify the response to TCP packet of different OS in their own default setting. In other

Table 2. Maximum amplification factor from protocols/hosts in 60s

Port	Protocol	Device	SA	t_1	FSA	t_2	AF
21	FTP	printer	137	300	0	0	685.0
22	SSH	server	101	300	0	0	505.0
23	Telnet	SX-1000U	165	300	0	0	825.0
25	SMTP	PC	156	300	0	0	780.0
80	HTTP	Printer	58	30	248	570	2385.0
515	printer	Switch	58	1	117	599	1169.0
587	submission	PC	156	300	0	0	780.0
902	iss-realsecure	PC	58	1	193	599	1927.8
903	iss-console-mgr	PC	193	300	0	0	965.0
912	apex-mesh	PC	160	300	0	0	800.0
3306	mysql	Server	136	92	136	508	1360.0
8080	http-proxy	Router	58	150	426	450	3340.0
9100	jetdirect	printer	410	300	0	0	2050.0

words, the virtual machine OS can reply `ACK` or `RST` to the retransmitted `SYN/ACK` packets. We repeat same analysis on default Linux Ubuntu 15.10, Windows 10, 8.1, Server 2016 and macOS 10.12 Sierra.

We found out that, in a virtual environment, Linux Ubuntu 15.10 and macOS Sierra will respond to the incoming `SYN/ACK` packet and therefore do not create amplification. However, all three kinds of Windows OS did not respond to the `SYN/ACK` packet. This allows retransmission from the virtual switch, we can evaluate that Windows OS in a virtual network is more vulnerable to TCP amplification attack.

3 Internal TCP Retransmission Attack

3.1 Experimental Model

We estimate the effectiveness of TCP retransmission attack under the virtual network environment with external TCP hosts. In order to execute attack simulation, we use the same experimental environment shown in Fig. 1 using all of 579 pairs of protocol/host in our local subnet.

First, the attacker sends `SYN` packets to 579 pairs of protocol/host. In this step, the attacker can send a multiple number of `SYN` packets using different port number simultaneously. In our first attack simulation, the attacker sends `SYN` packets from #1234 port to each # of port for TCP host. We define such condition as single-type attack. We expand the single-type attack by increasing the number of sender ports. For example, in 10-`SYN`-type, the attacker sends 10 `SYN` packets from ten kinds of port (#1234–#1243) to each TCP host at

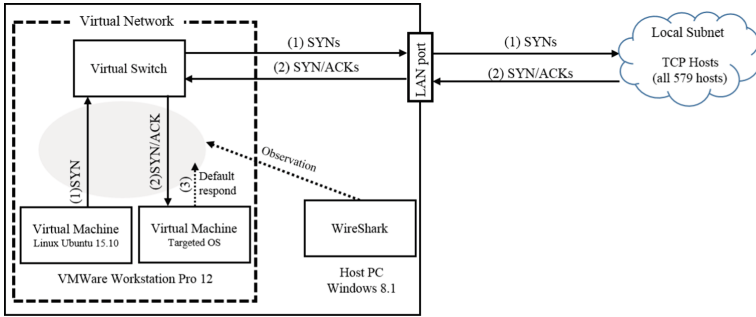


Fig. 4. Four experimental attack settings

the same time. In the same way, we also set 50-SYN-type (#1234-#1283) and 100-SYN-type (#1234-#1333) as shown in Fig. 5.

Since the size of SYN packet is 60 bytes, the amount of data E [byte] which is sent from attacker is calculated as follow

$$E = 60 \times 579 \times n \quad (2)$$

where n is the number of used source ports from the attacker.

In our attack simulation, we used TCP Hosts in our real subnet. These TCP hosts are actually used for office works and research activity and those management is entrusted to each section. To simulate the realistic attack environment, we did not open our attack experiment to the users of those TCP hosts beforehand. Therefore, we could not make exactly all of TCP hosts respond because some hosts were off-line or were powered off at the experiment time. From these conditions, we observed the following factors in 60 s:

- Number of respond TCP Hosts (Responded Port)
- Total amount of Received Data (Received Data)
- Maximum bandwidth of TCP retransmission (Max Bandwidth)

Using these results, we can consider the relation between

Relation-1. Number of SYN packet from attacker and responded TCP Hosts

Relation-2. Number of SYN packet from attacker and total amount of received data and maximum bandwidth, and

Relation-3. Number of SYN packet from attacker and retransmission type.

3.2 Results

Table 3 shows attack experiment results. From the viewpoint of Relation-1, the number of responded hosts decrease significantly when sending more SYN packets. We expect that the virtual switch was suffered from overload and it could not respond to every TCP query. On the other hand, on Relation-2, the data

Table 3. The attack experiment results (in 60 s)

SYN sent	Responded ports	Received data	Max bandwidth	AF
Single	540	11.5 MB	364 kbps	346.4
10-SYN	324	51.5 MB	1.91 Mbps	155.1
50-SYN	119	54.2 MB	1.96 Mbps	31.9
100-SYN	125	66.1 MB	2.12 Mbps	19.4

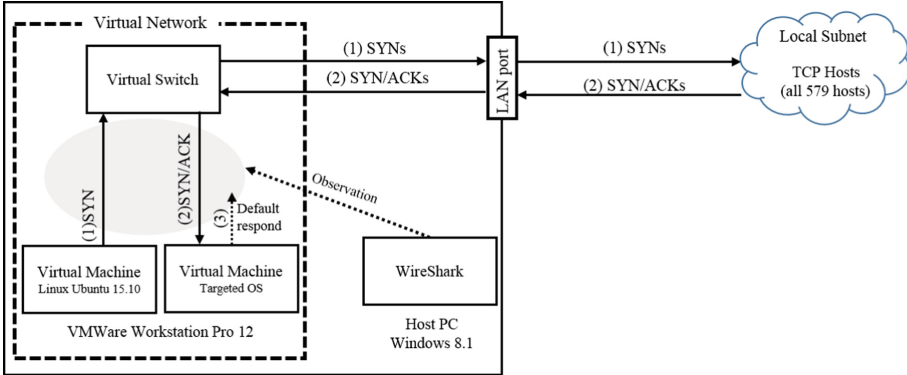


Fig. 5. Experimental model with different virtual machine OS

volume and bandwidth is increased when increasing the SYN packets. More specifically, the hosts retransmitted 11.5 MB with a maximum bandwidth of 364 kbps in the case of single-type. The attack of 10-SYN-type resulted in 51.5 MB with a maximum bandwidth of 1.91 Mbps, which is 5 times increased comparing to single-type. The retransmitted volume slightly increased when raising the SYN packets to 50 and 100, with 54.2 MB and 66.1 MB respectively. From the viewpoint of Relation-3, AF is lower when increasing the number of SYN packet. However, the larger reflected packet size the victim received, the virtual switch suffered more from the effectiveness of DoS and fall into an unstable state, which in some cases we cannot observe the transmission by Wireshark.

When receiving 10 or more SYN packets, some FTP, SSH, Telnet hosts changed from SYN/ACK packet retransmission to FIN/SYN/ACK retransmission, as mentioned in Sect. 2.2. In addition, we found new retransmission type using PSH packet. In other words, some TCP hosts send FIN/ACK ((β_2) in Fig. 3) or PSH when receiving a lot of SYN packet. We did not find these kinds of behavior from those hosts in the single-type attack. As a result, we consider that the hosts' behavior has some relations with the number of SYN packet it received. This result benefits attacker when they want to start FIN/SYN/ACK retransmission for this kind of attack.

Table 4. Received data on different OS (60 s)

SYN sent	Linux Ubuntu			macOS			Windows OS		
	Responded ports	Received data	AF	Responded ports	Received data	AF	Responded ports	Received data	AF
Single	496	31 kB	0.9	325	1.1 MB	33.1	504	10.8 MB	325.3
10-SYN	314	703 kB	2.1	436	23.3 MB	70.1	333	52.9 MB	159.3

3.3 The Virtual Machine’s OS Default Response

We repeated the single-type and 10-SYN-type SYN packets attack and tested it to victims using Linux Ubuntu 15.10, Windows 10, 8.1, Server 2016 and macOS 10.12 Sierra. Figure 5 shows our experimental model. Table 4 shows received traffic volume with each kind of OS. As the default setting, Linux Ubuntu responds to unknown TCP packets with RST segments. However, macOS cannot respond to the FIN/SYN/ACK packets and all tested Windows OS continued to receive every reflected TCP packets with no response. Furthermore, in the 10-SYN-type attack, while Linux Ubuntu can still handle all of the reflected packets with RST segment, macOS can only response about half of the incoming packets. Therefore, the data volume macOS received is much larger compared to Ubuntu. This indicates the vulnerability of Windows and macOS default settings under this kind of attack. We further test the 100-SYN-type attack on Windows OS victim. As we expected, although all of the virtual machines are still alive, the virtual switch is down and the virtual machines become unable to connect to the external network. In order to recover the virtual switch, we had to restart the virtual environment. As a consequence, all of the running virtual machines are also interrupted and restarted.

4 Threat Scenario

We derive new threat scenario based on our experimental results. We assume an internal attacker who is also a user of the virtual network and his purpose is to sabotage the other virtual machines’ activities. We show two types of scenario:

Scenario-1. The attacker sends SYN packets to TCP Hosts and does not respond to any SYN/ACK with ACK or RST packets. This ensures TCP retransmissions from the virtual switch, which makes the virtual switch becomes slow and eventually down. The effect of the attack is critical and easy to execute. On the other hand, retransmitted TCP packets focus on attacker’s virtual machines. It will expose attacker’s identity and the hypervisor can terminate his/her virtual machine to stop the attack.

Scenario-2. If the attacker has known other virtual machines’ IP address, the attacker can redirect retransmitted TCP packets to other virtual machines by IP spoofing. As a result, the hypervisor can not identify the attacker’s identity and can not determine which virtual machine to terminate in order to stop the attack.

However, the effect of attack depends on the number and setting of victim's OS (Sect. 3.3), in some cases, the risk will not become so serious.

Furthermore, the attacker can create an environmental risk to the virtual network by using local TCP host types that have a high amplification factor (Sect. 2.3). The attacker can also target Windows OS virtual machines only to leverage the attack (Sect. 3.3). For any organizations using a virtual system in their infrastructure (e.g. thin client machine), this kind of attack realistically create an opportunity loss threat.

5 Conclusion

In this paper, we found a case that the virtual switch itself can retransmit TCP packets and this behavior can be abused for TCP retransmission DoS attack, in particular, VMWare Workstation Pro 12.0.0. The majority of TCP retransmission type is SYN/ACK type, which is limited in an interval time-out. Besides that, we also find the FIN/SYN/ACK retransmission type that continues endlessly until the victim responds with RST packet. We further show that the host's behavior also depends on the number of received SYN packets. In addition, we also prove that Windows and macOS virtual machines at default setting are vulnerable to this kind of attack. In conclusion, TCP-based retransmission attack poses a new threat of an internal attack to any virtual network environment.

Acknowledgment. This work was supported by JSPS KAKENHI Grant Number 17K06455.

References

1. Kührer, M., Hupperich, T., Rossow, C., Holz, T.: Hell of a handshake: abusing TCP for reflective amplification DDoS attacks. In: 8th USENIX Workshop on Offensive Technologies (WOOT 2014), August 2014
2. Kührer, M., Hupperich, T., Rossow, C., Holz, T.: Exit from hell? Reducing the impact of amplification DDoS attacks. In: 23rd USENIX Security Symposium (USENIX Security 2014), August 2014
3. Rossow, C.: Amplification hell: revisiting network protocols for DDoS abuse. In: Symposium on Network and Distributed System Security (NDSS), February 2014. https://www.internetsociety.org/sites/default/files/01_5.pdf. Accessed 30 May 2017
4. IBM: Reviewing a year of serious data breaches, major attacks and new vulnerabilities, April 2016. <https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=SEP03394USEN>. Accessed 30 May 2017
5. IETF: Transmission Control Protocol, DARPA Internet Program Protocol Specification RFC793 (1981). <https://tools.ietf.org/html/rfc793>. Accessed 30 May 2017
6. Nmap ("Network Mapper") v7.12. <http://nmap.org>
7. Famatech: Advanced IP Scanner v2.4. <http://www.advanced-ip-scanner.com/>
8. Wireshark: Network protocol analyzer, Wireshark v2.2.6. <https://www.wireshark.org/>

9. VMWare: Workstation for Windows, VMWare Workstation Pro 12. <https://www.vmware.com/products/workstation.html>
10. FSN: The economy is flat so why are financials cloud vendors growing at more than 90 percent per annum?, March 2013. http://www.fsn.co.uk/channel_outsourcing/. Accessed 30 May 2017



Improving Detection of Wi-Fi Impersonation by Fully Unsupervised Deep Learning

Muhamad Erza Aminanto and Kwangjo Kim^(✉)

Cryptology and Information Security Lab, School of Computing,
Korea Advanced Institute of Science and Technology (KAIST),
Daejeon, Republic of Korea
{aminanto, kkj}@kaist.ac.kr

Abstract. Intrusion Detection System (IDS) has been becoming a vital measure in any networks, especially Wi-Fi networks. Wi-Fi networks growth is undeniable due to a huge amount of tiny devices connected via Wi-Fi networks. Regrettably, adversaries may take advantage by launching an impersonation attack, a common wireless network attack. Any IDS usually depends on classification capabilities of machine learning, which supervised learning approaches give the best performance to distinguish benign and malicious data. However, due to massive traffic, it is difficult to collect labeled data in Wi-Fi networks. Therefore, we propose a novel fully unsupervised method which can detect attacks without prior information on data label. Our method is equipped by an unsupervised stacked autoencoder for extracting features and a k -means clustering algorithm for clustering task. We validate our method using a comprehensive Wi-Fi network dataset, Aegean Wi-Fi Intrusion Dataset (AWID). Our experiments show that by using fully unsupervised approach, our method is able to classify impersonation attack in Wi-Fi networks with 92% detection rate without any label needed during training.

1 Introduction

The experts have already anticipated the growth of wireless network traffics [1]. Mobile traffics, including mobile 5G and Wi-Fi traffic are believed to increase tremendously, in the next 10 years [1]. Unfortunately, as the traffic increases, a number of malicious attacks by adversaries are have jumped accordingly [2]. An impersonation attack is one of common Wi-Fi attacks [3]. In this attack, adversaries can impersonate themselves as legitimate clients to gain unauthorized access. The impersonation attack also has a severe impact due to allowing unauthorized users to access the network as a security breach [4].

Intrusion Detection System (IDS) become a promising countermeasure for network attacks by leveraging machine learning application occasionally. Machine learning, based on data label availability, can be divided into two approaches: supervised and unsupervised learning. A supervised learning needs

prior information about the class label data. The supervised learning fits for the classification task, including attack detection. In the latter case, an unsupervised learning learns without any prior information about the class label of raw data. Therefore, the unsupervised learning fits for clustering task, which makes an efficient way to group similar data. In terms of attack detection, we may leverage unsupervised approach by claiming the outlier data from big clusters as attacks, since benign data usually form a big cluster. Besides that, unsupervised learning is suitable for huge Wi-Fi networks as labeling training data may be infeasible.

There are numerous famous unsupervised learning methods such as k -means clustering [5], Principal Component Analysis (PCA) [6] and Independent Component Analysis (ICA) [7]. The key characteristics of the three methods are: k number of class partitioning, orthogonal transformation and reveal hidden independent factors, respectively [8]. However, since we are facing huge and complex Wi-Fi data, the three traditional unsupervised learning methods are insufficient because the data might be not well distributed [9]. In order to overcome this problem, we venture to transform raw data into another form of data, which can lead to better unsupervised learning result.

One acceptable candidate for the transformations is Stacked Autoencoder (SAE) which transforms original features into more meaningful representation by reconstructing its input with the decoder. It provides an efficient way to validate that the important information in the data has been captured. The SAE as a deep learning method, can be efficiently used for unsupervised learning on a complex dataset. By stacking several unsupervised feature learning layers, and greedy method training for each layer, we can consider extracted features on each hidden layer as a new space with better form for clustering task. However, SAE is originally designed for capturing complex information in lower-dimensional features than the original features, not for clustering task. Therefore, we see SAE for assisting traditional clustering algorithm to achieve better clustering result. We then forward the newly formed features from non-linear SAE transformation into k -means clustering algorithm to improve k -means clustering performance.

We implement and test our work using a comprehensive Wi-Fi network benchmark dataset, called AWID dataset [3]. Besides this dataset, Koliadis *et al.* [3] also tested a series of existing machine learning models on the dataset in a heuristic manner. The lowest detection rate is observed on impersonation attack by detection rate of 22% only while our proposed approach outperforms on impersonation attack detection achieving a detection rate of 92%. Clearly, the novel way of combining deep learning transformation and traditional k -means clustering method improves the performance of impersonation attack detector and can be further generalized for different attack types in large scale Wi-Fi networks.

This paper is organized as follows: Sect. 2 reviews several related work. We provide our proposed approach along preliminaries in Sect. 3. Section 4 gives our experimental results and analysis. Conclusion and future work of this paper will be suggested in Sect. 5.

2 Related Work

There are several previous work which leverages deep learning techniques as a clustering method. Song *et al.* [9] proposed an autoencoder-based data clustering. They mapped original data space to a new space using autoencoder, which is more suitable for clustering, and claimed that by applying a non-linear transformation, the data become compact with respect to their corresponding cluster center in the new space. They modified original autoencoder by adding two new objective functions during training: minimize reconstruction error and distance. Comparing with Song *et al.* [9] which needs to modify the original autoencoder, our proposed method does not need to modify the original autoencoder and improves traditional clustering algorithm. While Saito and Tan [10] proposed similar work by mapping the input data to an embedded space, using autoencoder. They concatenated the learned representations of all intermediate layers. All features learned by each neural network layer were used to generate a combined representation which is useful for effective cluster analysis. Different from Saito and Tan [10], we leverage unsupervised k -means clustering algorithm instead of supervised k -Nearest Neighbor (k -NN) classification algorithm. We also use a single hidden layer only without concatenation of all hidden layers to maintain the process still lightweight.

A lot of proposed approaches for detecting impersonation attacks [11–14]. [11–13] were designed to detect one particular impersonation attack by adding or modifying specific protocols. However, in particular, Aminanto and Kim [14] proposed one general model that can detect an impersonation attack by reducing the features dimensionalities and adopting SAE at final stage. Unfortunately, their approach needs data labels which is supervised learning algorithm. Therefore, we propose a fully unsupervised approach for coping with huge and complex Wi-Fi network traffics.

3 Our Approach

In this Section, we briefly describe our novel fully unsupervised deep learning-based Wi-Fi impersonation attack detector. For clarity, we firstly introduce preliminaries about SAE and k -means clustering, and then explain how the overall scheme works for detecting attacks.

3.1 Stacked Autoencoder (SAE)

An autoencoder is a symmetric neural network model as shown in Fig. 1, which belongs to unsupervised learning in the sense that a model could be built from non-labeled data. To extract new-lower dimensional features, autoencoder uses an encoder-decoder paradigm as shown in Fig. 1, which can capture relevant data from the original data. The encoder is a function that maps an input X to a representation layer H as expressed by Eq. (1).

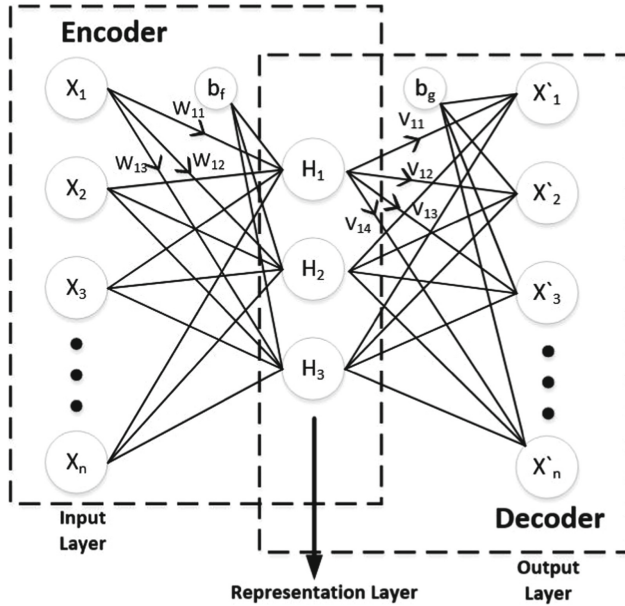


Fig. 1. Autoencoder network with symmetric input-output layers

$$H = s_f (WX + b_f), \tag{1}$$

where s_f is a non-linear activation function, in this case of a logistic sigmoid, $s_f(t) = \frac{1}{1 + e^{-t}}$, where t is the function input. The W and b_f denote a weight matrix for features and a bias vector for encoding, respectively. The decoder function expressed in Eq. (2) maps representation layer H back to a reconstruction X' as an output.

$$X' = s_g (VH + b_g), \tag{2}$$

where s_g is the activation function of the decoder, which is a sigmoid function too. The V and b_g denote a weight matrix for features and a bias vector for decoding, respectively. Autoencoder training phase finds optimal parameters $\theta = \{W, V, b_f, b_g\}$ which minimize the reconstruction error E between the input data X and its reconstruction output X' on a training set as shown in Eq. (3).

$$E = \frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K (X'_{kn} - X_{kn})^2 + \lambda \cdot \Omega_{weights} + \beta \cdot \Omega_{sparsity}, \tag{3}$$

where N and K denote the total number of training data and the number of variables for each data, respectively. $\Omega_{weights}$ represents L_2 regularization, while $\Omega_{sparsity}$ denotes sparsity regularization, which evaluates how close the average output activation value and the desired value. The coefficient of L_2 regularization

term λ and the coefficient of sparsity regularization term β are specified during autoencoder training.

Autoencoder can be used as a deep learning technique by unsupervised greedy layer wise pre-training algorithm as depicted in Fig. 2, which is called Stacked Autoencoder (SAE). In this algorithm, all layers except the last layer are initialized in a multi-layer neural network. Each layer is then trained in an unsupervised manner as autoencoder which constructs new representations of the input.

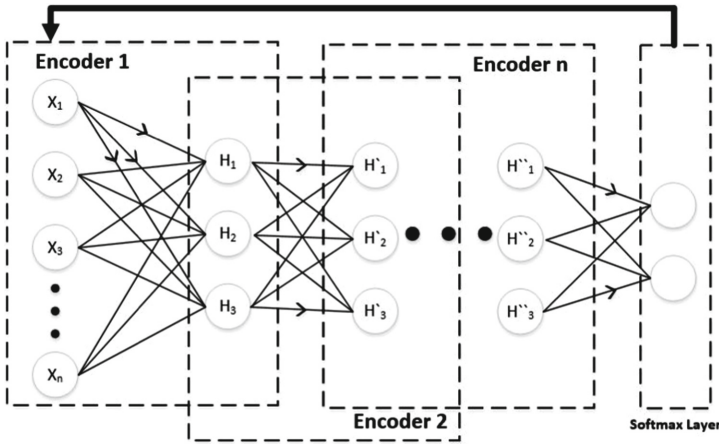


Fig. 2. Stacked autoencoder (SAE) network with three hidden layers

The final layer implements the softmax for the classification the deep neural network. Softmax function is a generalized term of the logistic function that squashes the K -dimensional vector $\mathbf{v} \in \mathbb{R}^K$ into K -dimensional vector $\mathbf{v}^* \in (0, 1)^K$ which adds up to 1. The softmax layer minimizes the loss function, which is the cross entropy function.

3.2 K -means Clustering

K -means clustering algorithm groups all observations data into k clusters iteratively until convergence will be reached. In the end, one cluster contains similar data since each data enters to the nearest cluster. K -means algorithm assigns a mean value of the cluster members as a cluster centroid. In every iteration, it calculates the shortest Euclidean distance from an observation data into any cluster centroid. Besides that, the intra-variances inside the cluster are also minimized by updating the cluster centroid iteratively. The algorithm would terminate when convergence is achieved, which the recent clusters are the same as the previous iteration clusters [8].

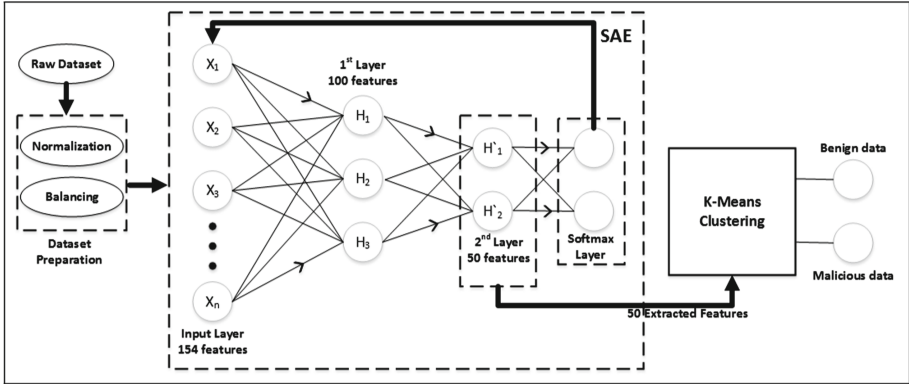


Fig. 3. Our proposed scheme contains feature extraction and clustering tasks

Algorithm 1. Pseudocode of Fully Unsupervised Deep Learning

```

1: procedure START
2:   function DATASET PREPARATION(Raw Dataset)
3:     for each data instance do
4:       Convert into integer value
5:       Normalization  $z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$ 
6:     end for
7:     Balance the normalized dataset
8:     return InputDataset
9:   end function
10:  function SAE(InputDataset)
11:    for  $i=1$  to  $h$  do ▷  $h=2$ ; number of hidden layers
12:      for each data instance do
13:        Compute  $H = s_f(WX + b_f)$ 
14:        Compute  $X' = s_g(VH + b_g)$ 
15:        Minimize  $E = \frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K (X'_{kn} - X_{kn})^2 + \lambda \cdot \Omega_{weights} + \beta \cdot \Omega_{sparsity}$ 
16:         $\theta_i = \{W_i, V_i, b_{f_i}, b_{g_i}\}$ 
17:      end for
18:       $InputFeatures \leftarrow W_2$  ▷ 2nd layer, 50 extracted features
19:    end for
20:    return InputFeatures
21:  end function
22:  Initialize clusters and  $k=2$  ▷ two clusters: benign and malicious
23:  function k-MEANS CLUSTERING(InputFeatures)
24:    return Clusters
25:  end function
26:  Plot confusion between Clusters and target classes
27: end procedure
    
```

3.3 Fully Unsupervised Deep Learning

In this subsection, we describe our novel fully unsupervised deep learning-based IDS for detecting impersonation attacks. There are two main tasks, feature extraction and clustering tasks. Figure 3 shows our proposed scheme which contains two main tasks in cascade. We use a real Wi-Fi networks-trace, AWID dataset [3], which contains 154 original features. Before the scheme starts, normalizing and balancing process should be done in order to achieve best training performance. Algorithm 1 explains the procedure of the proposed scheme in detail.

The scheme starts with two cascading encoders, and the output features from the second layer then forwarded to the clustering algorithm. The first encoder has 100 neurons as the first hidden layer while the second encoder comes with 50 neurons only. We follow a common rule for choosing the number of neurons in a hidden layer by using 70% to 90% of the previous layer. In this paper, we define $k = 2$ since we consider two classes only. The scheme ends by two clusters formed by k -means clustering algorithm. These clusters represent benign and malicious data.

4 Evaluation

We evaluate the proposed scheme on AWID dataset. We firstly show the effectiveness of leveraging second layer representation of SAE training result compared to original data. We implement SAE and k -means clustering algorithm using MATLAB R2016b running on an Intel Xeon E-3-1230v3 CPU @3.30 GHz with 32 GB RAM. We verify our proposed scheme by comparing the proposed scheme with the previous work. We introduce the dataset has been used and evaluation metrics in the next subsections.

4.1 Dataset

We use AWID dataset as a benchmark dataset since the dataset might become a common benchmark dataset for wireless network research due to its comprehensiveness and real world-alike characteristics. Regarding the number of classes, the dataset has two types of attack classes: “ATK” and “CLS”. The “ATK” dataset consists of 16 classes including one benign class, while the “CLS” data contains four classes only. The 16 classes of the “ATK” dataset can be classified to four attack categories in the “CLS” dataset. In this paper, we use the “CLS” dataset which contains benign, impersonation, injection and flooding classes. However, we consider two classes only among four classes. Besides that, the AWID dataset is also divided into two types based on the size of data instances included, namely, full and reduced datasets. There are 1,795,595 data instances in the full dataset, with 1,633,190 and 162,385 benign and attack instances, respectively. While the reduced dataset contains only 575,643 instances, with 530,785 and 44,858 benign and attack instances, respectively. In this paper, we used the reduced “CLS” AWID dataset for the sake of simplicity.

The dataset expresses the nature of a network, where the number of benign instances is larger than attack instances [3]. The ratio between benign and attack instances are 10:1 and 11:1 for training and test datasets, respectively. This situation might cause a bias during training, and infer machine learning performance. Therefore, we balance the dataset for training purpose. The ratio between benign and attack instances then become 1:1 for both balanced training and test datasets. The benign instances are randomly reduced into 163,319 data instances for training dataset while 53,078 data instances for test dataset.

The AWID dataset not only consists of discrete data but also continuous and symbolic data types, with flexible value ranges. This situation might confuse any machine learning during training. The dataset preparation should be done in advance, which contains two main tasks: mapping symbolic-valued attributes to integer values and normalizing tasks. First, target classes would be mapped to integer type: 1 for benign instances and 2 for impersonation attack. Second, symbolic attributes, such as a receiver, destination, transmitter, and source address, would be mapped to integer values with a minimum value of 1 and a maximum value of i , where i is the number of all symbols. Third, some attributes that have a hexadecimal data type, such as WEP Initialization Vector (IV) and Integrity Check Value (ICV), need to be cast into integer values too. Also, there are some attributes left with a continue data type, like timestamps. Last, the dataset also contains a question mark (“?”) for unavailable values for the corresponding attributes. The question marks are assigned to zero value. After all attribute values are cast into integer values, each of the attributes is linearly normalized between zero and one. Equation (4) shows the normalizing formula:

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}, \quad (4)$$

where z_i denotes the normalized value, x_i refers to the corresponding attribute value, and $\min(x)$ and $\max(x)$ are the minimum and maximum values of the attribute, respectively.

4.2 Evaluation Metrics

We use several metrics that commonly used for measuring IDS performance [15]: classification accuracy (Acc), Detection Rate (DR), False Alarm Rate (FAR). Acc shows the overall effectiveness of an algorithm. DR , also known as $Recall$, refers to the number of attacks detected divided by the total number of attack instances in the test dataset. FAR is the number of normal instances classified as an attack divided by the total number of normal instances in the test dataset. F_1 score measures a harmonic mean of precision and recall, where $Precision$ shows the number of attacks compared to the total of classified instances as an attack. Intuitively, our goal is to achieve a high Acc , DR , $Precision$ and F_1 score, and at the same time, maintain low FAR . The above measures can be defined as shown in Eqs. (5), (6), (7), (8) and (9):

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

$$DR = Recall = \frac{TP}{TP + FN} \quad (6)$$

$$FAR = \frac{FP}{TN + FP} \quad (7)$$

$$Precision = \frac{TP}{TP + FP} \quad (8)$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (9)$$

where True Positive (TP) is the number of intrusions correctly classified as an attack. True Negative (TN) is the number of normal instances correctly classified as a benign packet. False Negative (FN) is the number of intrusions incorrectly classified as a benign packet. False Positive (FP) is the number of normal instances incorrectly classified as an attack.

4.3 Experimental Results

We implement our proposed scheme as shown in Algorithm 1. There are two hidden layers in the SAE network with 100 and 50 neurons accordingly. The encoder in the second layer fed with features formed by the first layer of encoder. The softmax activation function is implemented in the final stage of the SAE in order to optimize the SAE training. The 50 features extracted from the SAE are then forwarded to k -means clustering algorithm as an input. We use random initialization for k -means clustering algorithm. However, we set a certain value as a random number seed for reproducibility purpose. We compare clustering results from three inputs: original data, features from the first hidden layer of the SAE and features from the second hidden layer of the SAE as shown in Table 1.

Table 1. The evaluation of our proposed scheme

Input	DR (%)	FAR (%)	Acc (%)	$Precision$ (%)	F_1 (%)
Original data	100.00	57.17	55.93	34.20	50.97
1 st hidden layer	100.00	57.48	55.68	34.08	50.83
2 nd hidden layer	92.18	4.40	94.81	86.15	89.06

We observe the limitation of a traditional k -means algorithm, which unable to clusters complex and high dimensional data of AWID dataset, as expressed by 55.93% of accuracy only. Although 100 features coming from the 1st hidden layer achieved 100% of detection rate, the false alarm rate is still unacceptable with 57.48%. The k -means algorithm fed by 50 features from the 2nd hidden layer achieved the best performance among all as shown by the highest F_1 score (89.06%) and Acc (94.81%), also the lowest FAR (4.40%). Despite a bit lower

detection rate, our proposed scheme improves the traditional k -means algorithm in overall by almost twice F_1 score and accuracy.

Figure 4 shows cluster assignment result in Euclidean space, by our proposed scheme. Black dots represent attack instances, while gray dots represent benign instances. The location of cluster centroid for each cluster is expressed by X mark.

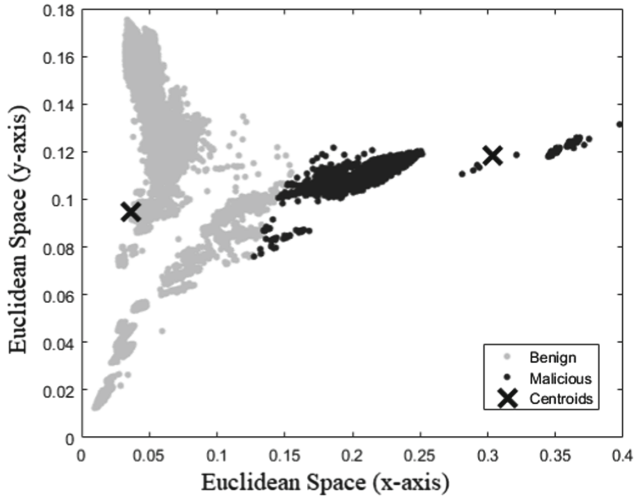


Fig. 4. Cluster assignment result in Euclidean space by our proposed scheme

We also compare the performance of our proposed scheme against two related previous work by Kolia *et al.* [3] and Aminanto and Kim [14] as shown in Table 2. Our proposed scheme is able to classify impersonation attack instances with a detection rate of 92.18% while maintaining low FAR , 4.40%. Kolia *et al.* [3] tested various classification algorithms such as Random Tree, Random Forest, J48, Naive Bayes, *etc.*, on AWID dataset. Among all methods, Naive Bayes algorithm showed the best performance by correctly classifying 4,419 out of 20,079 impersonation instances. It achieved approximately 22% DR only, which is unsatisfactory. Aminanto and Kim [14] proposed another impersonation detector by combining Artificial Neural Network (ANN) with SAE. They successfully improved the IDS model for impersonation attack detection task by achieving a DR of 65.18% and a FAR of 0.14%. In this study, we leverage SAE for assisting traditional k -means clustering with extracted features. We still have a high false alarm rate, which leads to a severe impact of IDS [16]. However, we can accept false alarm rate value about 4% since we use fully unsupervised approach here. We can adjust the parameters and cut the FAR down, but, less FAR or high DR remains a tradeoff for users and will be discussed in further work. We observe the advantage of SAE for abstracting a complex and high dimensional data to

assist traditional clustering algorithm which is shown by reliable DR and F_1 score achieved by our proposed scheme.

Table 2. Comparison with previous work

Method	DR (%)	FAR (%)	Acc (%)	$Precision$ (%)	F_1 (%)
Kolias <i>et al.</i> [3]	22.01	0.02	97.14	97.57	35.92
Aminanto and Kim [14]	65.18	0.14	98.59	94.53	77.16
Our proposed scheme	92.18	4.40	94.81	86.15	89.06

5 Conclusion and Future Work

In this paper, we improve traditional k -means clustering algorithm by proposing a novel fully unsupervised-based intrusion detection system incorporating deep learning technique, a stacked autoencoder. We implement SAE to achieve high level abstraction of complex and huge Wi-Fi network data. The SAE has important features: model-free and learnability on large-scale data, which is suitable for the open nature of Wi-Fi networks where attackers can easily impersonate as legitimate users. We believe that the extracted features by SAE are in the new space that can improve clustering algorithm performance. Our experiments show significant improvements compared to previous work with notably 94.81% of accuracy.

In the near future, we will further investigate and propose a method to reduce false alarm rate in order to achieve a reliable IDS. In addition, we will discuss using deep learning techniques, especially stacked autoencoder as an outlier detection for detecting unknown attacks.

Acknowledgment. This work was partly supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (B0101-16-1270, Research on Communication Technology using Bio-Inspired Algorithm) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. NRF-2015R1A2A2A01006812).

References

1. Osseiran, A., Boccardi, F., Braun, V., Kusume, K., Marsch, P., Maternia, M., Tullberg, H.: Scenarios for 5G mobile and wireless communications: the vision of the METIS project. *IEEE Commun. Mag.* **52**(5), 26–35 (2014)
2. Kolias, C., Stavrou, A., Voas, J., Bojanova, I., Kuhn, R.: Learning internet-of-things security hands-on. *IEEE Secur. Priv.* **14**(1), 37–46 (2016)
3. Kolias, C., Kambourakis, G., Stavrou, A., Gritzalis, S.: Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset. *IEEE Commun. Surv. Tutor.* **18**(1), 184–208 (2015)

4. Beyah, R., Kangude, S., Yu, G., Strickland, B., Copeland, J.: Rogue access point detection using temporal traffic characteristics. In: Global Telecommunications Conference, 2004 GLOBECOM 2004, vol. 4, pp. 2271–2275. IEEE (2004)
5. Jain, A.K.: Data clustering: 50 years beyond k -means. *Pattern Recogn. Lett.* **31**(8), 651–666 (2010)
6. Abdi, H., Williams, L.J.: Principal component analysis. *Wiley Interdisc. Rev. Comput. Stat.* **2**(4), 433–459 (2010)
7. Lee, T.W.: Independent Component Analysis, pp. 27–66. Springer, Heidelberg (1998). https://doi.org/10.1007/978-1-4757-2851-4_2
8. Jiang, C., Zhang, H., Ren, Y., Han, Z., Chen, K.C., Hanzo, L.: Machine learning paradigms for next-generation wireless networks. *IEEE Wirel. Commun.* **24**(2), 98–105 (2016)
9. Song, C., Liu, F., Huang, Y., Wang, L., Tan, T.: Auto-encoder based data clustering. In: Ruiz-Shulcloper, J., Sanniti di Baja, G. (eds.) CIARP 2013. LNCS, vol. 8258, pp. 117–124. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41822-8_15
10. Saito, S., Tan, R.T.: Neural clustering: concatenating layers for better projections. In: Workshop Track of International Conference on Learning Representations (ICLR) (2017)
11. Shang, T., Gui, L.Y.: Identification and prevention of impersonation attack based on a new flag byte. In: 2015 4th International Conference on Computer Science and Network Technology (ICCSNT), vol. 1, pp. 972–976. IEEE (2015)
12. Yilmaz, M.H., Arslan, H.: Impersonation attack identification for secure communication. In: 2013 IEEE Globecom Workshops (GC Wkshps), pp. 1275–1279. IEEE (2013)
13. Laksmi, B., Sanmuga, L., Karthikeyan, R.: Detection and prevention of impersonation attack in wireless networks. *Int. J. Adv. Res. Comput. Sci. Technol. (IJARCST)* **2**(1), 267–270 (2014)
14. Aminanto, M.E., Kim, K.: Detecting impersonation attack in WiFi networks using deep learning approach. In: Choi, D., Guilley, S. (eds.) WISA 2016. LNCS, vol. 10144, pp. 136–147. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56549-1_12
15. Al-Jarrah, O.Y., Alhussain, O., Yoo, P.D., Muhaidat, S., Taha, K., Kim, K.: Data randomization and cluster-based partitioning for Botnet intrusion detection. *IEEE Trans. Cybern.* **46**(8), 1796–1806 (2016)
16. Sommer, R., Paxson, V.: Outside the closed world: on using machine learning for network intrusion detection. In: 2010 IEEE Symposium on Security and Privacy (S&P), pp. 305–316. IEEE (2010)



Cyber Influence Attack: Changes in Cyber Threats Seen in the Russian Hacking Incident

Mookyu Park¹, Moosung Park², and Kyungho Lee¹(✉)

¹ Institute of Cyber Security and Privacy (ICSP), Korea University, Anam-ro, Seongbuk-Gu, Seoul, Republic of Korea
kevinlee@korea.ac.kr

² Agency for Defense Development, Ogeum-ro, Songpa-Gu, Seoul, Republic of Korea

Abstract. As the Russian government is revealed that it had intervened in the US presidential election by hacking, the social confusion caused by cyber attacks increased. This incident has led to the impeachment of the president by the dismissal of FBI director James Comey. In the French presidential election held in 2017, the social confusion created because of fake news during the period of election silence. The past cyber attacks were used as deception tactics, like the Georgian war. Nowadays, these attacks are concentrated in the period of social issues. In other words, these recent changes in cyber attacks have begun to have an adverse effect on systems in the real world, such as hybrid battlefields. This attack is called cyber influence attacks. This paper identifies the weaknesses of the basic democratic election system and classifies it against cyber influence attacks. In addition, we analyze the cyber influence attacks in Russia during the US presidential election in 2016 as a case study.

1 Introduction

Cyber attacks have been made in the form of attacks on SW, HW, and network objects in cyberspace. For example, traditional cyber attacks such as hacking, DDoS, and APT attacks have affected cyber space. However, cyber attacks are expanding to the real world as the relationship between cyberspace, such as social networks, and the real world gets closer.

Recent cyber attacks are changing into forms that can harm the real world. The attacks such as WannaCry Ransomware have caused social disruption by encrypting ordinary citizens' files as well as infrastructure paralysis [1]. This social confusion is threatening when there are big issues of the state. The confusion has not ceased until the Trump administration was launched [2]. In particular, Russia's cyberattack on the US presidential election affected democratic elections, which emphasized one vote each. In other words, it was confirmed that the development of cyber weapons for cyber attacks could be extended from systematic damage, which could damage the political system of each country such as information leakage, fake news production, and democracy.

James Comey, former FBI director in charge of the Russian hacking incident, said in his keynote speech at the Boston Conference on Cyber Security 2017 that cyber security should be a top priority for all companies in the United States at all levels. He also argued that it is important to build trust between the government and the private sector to counter the threat. His remarks include concerns about the impact of the recent Russian hacking incident on the private sector and fake news in cyberspace [3]. In RSA 2017, US Department of Homeland Security (DHS) chief counsel Michael McCaul criticized the US government for responding to the 21st century threat with 20th century technology and 19th century institutions [4]. In addition, Weeping Angel, which is aimed at attacking and monitoring IoT devices such as CIA smart TVs, has been released [5]. Considering the development of cyber weapons and the social impact of cyber attacks, the impact of cyber threats is significant.

This paper presents the vulnerabilities of the democracy system and its possible attack method, focusing on the cyber influence attack, which is the secondary damage caused by the cyber attack. In addition, Twitter data confirms whether the cyber influence attack that occurred during the US presidential election was not an independent act of Russia, and that it is related to the Trump Administration.

2 Background and Related Works

This chapter explains the concepts used in this study and related research. The first part explains the definition of cyberspace and each layer, and confirms the range of influence of cyber influence attack. Second, I will explain the related research that will be used in social media and analysis. This study confirms the effect of social media on the real world.

2.1 Concept of Cyberspace Layer

Cyberspace is composed of interdependent environments including the Internet, communications, networks, and embedded processes. According to Joint Publication 3-12R Cyber Operation, cyberspace consists of three layers: physical layer, logical layer and persona layer. The physical layer is a layer composed of geographical elements and physical network elements. A logical layer is a network composed of logical connections between nodes established in the physical layer. The persona layer is a layer that includes human and cognitive aspects and includes personally identifiable information such as e-mail address and IP address [6].

Clark described Cyberspace as four layers: Physical Layer, Logical Layer, Information Layer, and Top Layer. Logical Layer and Physical Layer have the same meaning and role as JP 3-12R Layers. However, in JP 3-12R, the persona layer composed of one is divided into the information layer and the top layer-people 2 layer. The information layer is a layer in which information stored, transmitted, and converted exists in the cyberspace. Top Layer-People is a group

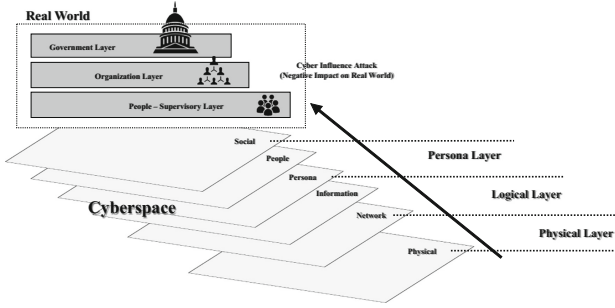


Fig. 1. Cyberspace layer

of people who participate in the cyber-experience, communicating using services and functions, making decisions and making plans using information [7].

The Cyber Situational Awareness (Cyber SA) model divides the above three and four layers of cyberspace into 6 layers of social, people, persona, information, network and real world. Cyber SA consists of three components: Persona layer, which David Clark claimed, and Top layer, Social, People, and Persona. In addition, Cyber SA explained that each layer is connected and dependent, and that the social component at the top level ultimately influences the real world [8].

According to a report released by the US Department of Homeland Security in 2011, *Enabling Distributed Security in Cyberspace* explains that the cyberspace layer is extended to cyber ecosystem. In this report, the cyber ecosystem explained that private enterprises, nonprofit organizations, governments, individuals, processes and cyber devices (computer, software and communication technologies) interacted for many purposes as well as natural ecosystems. The Cyber ecosystem consists of fifteen layers that are connected to hardware, cables, people, buildings, all physical and physical items to solve the complex interactions of people, processes and technology [9]. This is shown in Fig. 1.

The cyber influence attack presented in this study is a phenomenon that occurs because each layer interacts in the above cyber ecosystem. That is, there is a possibility that the cyber attack can have a negative impact on the real world. The following section describes related research on social media at the contact points of cyberspace and physical space.

2.2 Research of Social Media

This section describes the study of the representative social media of the persona class, which can influence the cyber attack from the perspective of the cyberspace layer. The reasons for explaining the research related to the social media are that the fake news and the information generated by the outbreak spread through social media such as Facebook and Twitter.

Herrick’s social media and cyber operations, published in CyCon 2016, argued that operations in social media could be an important factor in organizing

cyberspace operations. In this study, we propose three methods of collecting, attacking and defending information on social media for effective operation of cyber space [10].

The study of the social media according to the threat of the real world is the study of the relationship between terror media such as ISIS (Islamic State of Iraq and Syria) or Al-Qaeda and social media. Golan and Lim have investigated the social impact of social media activism on ISIS online recruitment through differences in third-party effects and self-perception [11]. Weimann explained through research that ISIS is recruiting online through the ability of social media to measure direct contact through interactive communication and to enable sophisticated profiling of recruiters [12]. In other words, these studies show that the negative effects of the real world can be reflected in cyberspace.

Amato et al. conducted a study to detect malicious behavior, that is, Lone Wolf, that could affect the Real World among the information that occurred in Twitter. These results show that the malicious influence on cyberspace affects the physical space [13]. The next chapter categorizes the types of cyber influence attacks proposed in this study. This study also categorizes the weaknesses of democracy and the corresponding media.

3 Democracy Vulnerabilities and Cyber Influence Attacks

Cyber attacks must intervene in human decision making to confuse public opinion. This intervention in decision making affects democratic elections. This chapter explains the weaknesses of human decision making in the democratic electoral system that can occur as the effects of cyberspace expand. In addition, this study attempts to classify attacks against cyber influence attack through vulnerability.

3.1 Vulnerabilities of the Democratic System

The current democracy system is called representative democracy. The majority rule in this democratic system is essential to securing national sovereignty and democratic legitimacy. The principle of majority rule is one of the democratic devices operating by the majority rule. Schumpeter's theory of democracy claimed the equivalence of "the equal vote" and "the equal value", which had a great impact on the democratic electoral system [14, 15]. As a result, one-person ordinary electoral system and equality vote have been established.

The basic framework of representative democracy is not the will of the ruler but the will of the people or will of people. The voting method of one elector who elects a national representative appointed by Schumpeter is to form a government and a parliament by collecting the will of the majority. Will of people lets people make a rational decision and takes a precious vote. The rational decision stage of human beings is divided into five stages: problem recognition stage, information gathering stage, alternative setting stage, alternative evaluation stage, and alternative selection stage [16].

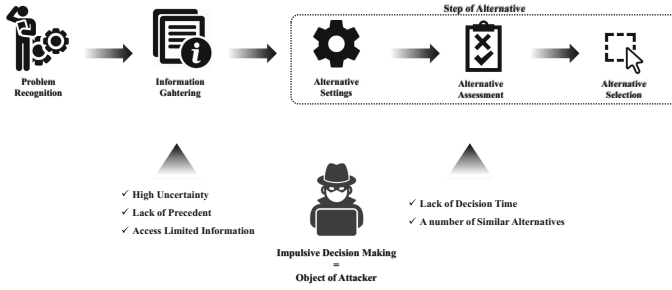


Fig. 2. Object of attacker in democratic system

However, there are vulnerable situations in which individual decisions are made by an impulsive or intuitive decision-making environment. According to Deacon and Firebaugh, impulsive or intuitive decision-making situations include increased uncertainty, lack of cases, limited access to information, lack of decision-making time, and similar alternatives [17]. These vulnerabilities act as an impediment to each reasonable step. In this case, high uncertainty, lack of precedence, and access limited information can interfere with rational decision making of voters by acting on alternatives, alternative evaluations, and alternative choices in the information gathering stage. This is shown in Fig. 2.

If cyber influence attacks attack these five vulnerabilities in a humanitarian situation, it creates a situation that hinders voters’ decision making. An attacker who chooses Cyber influence attacks as an attack method should focus on spreading information for attack. At this point, there are mediators necessary for information dissemination. These are social media, online news, and community websites that have high diffusion power in cyberspace. The common feature of these mediators is that there is more informal information by individuals or groups than official information through government or certified bodies. The corresponding decision steps for each vulnerabilities are shown in Table 1 below.

Unofficial information increases during large event periods such as presidential elections in a situation where cyberspace and connections are present. Information produced or disseminated through these informal media can interfere with human decision making. The following sections describe cyber influence attacks and classify attack methods that can attack reasonable decisions.

3.2 Types of Cyber Influence Attacks in Democratic Systems

The cyber influence attack utilizes informal media with many contact points, such as social media, community website etc. Cyber influence attack utilizes many informal agents with many of these contact points. Cyber influence attacks are common in that they are attacked through the spread of information, though depending on the attacker’s capabilities and the active layer. In this study, logical influence attacks (LIA) starting from the Logical Layer and persona influence attacks (PIA) starting from the Persona Layer are defined.

Table 1. Vulnerabilities of voter’s decision making

Type	Vulnerabilities	Role	Phase of decision-making
A	High uncertainty	Increase uncertainty about the selection of voters for the future	Information gathering
B	Lack of precedent	When voters make decisions, there is a lack of additional relevant information	Information gathering
C	Access limited information	Absence of the official announcement of the government or the organization when the voter has limited access to the information	Information gathering
D	Lack of decision-making time	When a voter makes a decision, there is a limited amount of time	Alternative setting
			Alternative assessment
			Alternative selection
E	Number of similar alternatives	If the voter is unable to select a priority in his decision due to the excess or shortage of information	Alternative setting
			Alternative assessment
			Alternative selection

A logical influence attack (LIA) is an attack that starts at the logical layer. Most of these attacks are general cyber attacks, but they are aimed at information leakage or capture. The LIA is a cyber influence attack that ultimately affects the real world through the persona layer. Such an LIA is similar to the existing cyber attack, but the target of the attack is in information, such as confidential information, personal information, and the like. The cyber attacks used in LIA are common attacks of common types of network attacks: eavesdropping, identity spoofing, password-based attacks, man-in-middle attacks, compromised-key attacks, sniffer attacks, and application-layer attacks [18,19]. Confidential information obtained through this process spreads through social media, community site, which is a mediator of persona layer. At this time, the information that is spread is based on facts. LIA has the disadvantage that the attacker must have the expertise to attack, but if the information is successfully leaked through the attack, the damage is large because the information itself is confidential information of the individual or the organization. The attack flow for the LIA is shown in Fig. 3 below.

A persona influence attack (PIA) has a negative impact on the real world through attacks initiated at the persona layer due to the influence attack

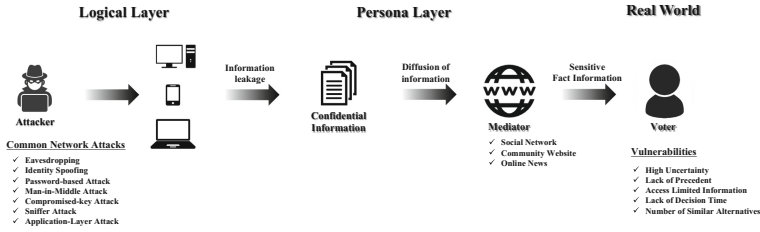


Fig. 3. Logical influence attack flow

occurring in the persona layer. The attack method of the PIA is based on the production and diffusion of information. The production of information is the deception of illegal information, the production of illegal and false information. Information dissemination uses social media such as Facebook or Twitter, community websites such as WikiLeaks, and online news. PIAs are characterized by their ability to attack without expert knowledge. Also, it can be used for information dissemination quickly through a user who has a high degree of centrality, such as social media, which can serve as a hub. These attacks include malicious rumors, illegal information, and fake news. The attack flow for the PIA is shown in Fig. 4 below.

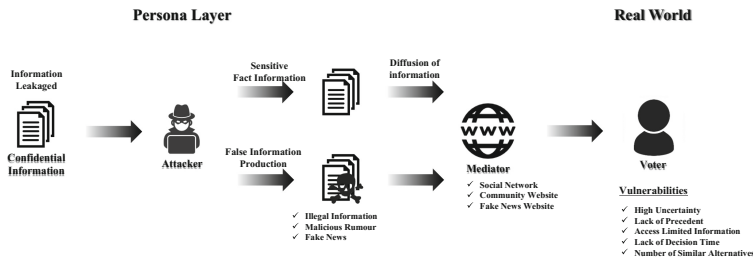


Fig. 4. Persona influence attack flow

Cyber influence attacks are the methods of obtaining information from producing information or exploiting information directly. However, the medium used to spread the information is the same. Also, the choice of mediator with high spreading power increases the damage to cyber influence attacks. The following chapter will analyze the cyber influence attack resulting from the Russian scandal through the US presidential election in 2016. In addition, through the trend analysis of Donald Trump and Hillary Clinton during the election period, the influence of the cyber influence attack is checked.

4 Case Study: Russian Scandal

This chapter analyzes the Russian hacking incident during the 2016 US presidential election. Through this case analysis, we will explain the influence attack in each cyberspace layer. Using sentimental analysis and cosine similarity, we analyze the negative impacts on the trend during the presidential election. This study uses fake news data provided by Kaggle and Donald Trump (@realDonaldTrump) tweet.

4.1 Sentimental Analysis and Cosine Similarity

This section uses the sentimental analysis of Donald Trump’s tweets using Naive Bayes. Sentimental analysis through Naïve Bayes assumes that the probability of each word in the sentence is independent of each other, and the emotion of the sentence is predicted through the frequency of each word. The reason for this assumption is that the relationships between words are independent because the number of parameters needed to calculate the Bayesian probability can be reduced.

In this study, the sentimental analysis for this tweet is set as follows. X means the word contained in Tweet. In this case, X_i is the i th word in Tweet. Y is a positive and negative value. This is expressed by the following equation.

$$h_{NB}(x) = \operatorname{argmax}_y P(y) \prod_{i=1}^{\text{Length of Tweet}} P(x_i|y) \quad (1)$$

In the case of sentimental analysis through Naïve Bayes, as described above, it is not related to the position because it assumes the relationship between the words in the tweet independently [20]. The formula for this is as follows.

$$\prod_{i=1}^{\text{Length of Tweet}} P(x_i|y) = \prod_{w=1}^W P(w|y)^{\text{count}_w} \quad (2)$$

Through this, Naive Bayes judges whether each tweet is positive or negative through the training phase. The tweets used here are tweets that include the keywords for Hillary Clinton, the rival during the presidential election. However, instead of using the name “Hillary Clinton” on the tweet, I used “She” and “her” to filter the tweet. When the compound was negative, the negative expression was strong, but in the positive case, there was a negative tweet for Hillary Clinton. The reason for this is that there are cases where the use of irony is used.

In addition, the relationship between Donald Trump’s tweet and fake news was analyzed using cosine similarity. The reason for using cosine similarity is that the distribution of frequency such as term and word extracted from tweet is exponential scale. In addition, it is possible to check whether the direction of the text vectors is similar to each other. To use this, the content of each tweet and fake news was converted into a vector and the similarity was measured [21]. In order to vectorize each word, we used tokenize and *sklearn* in the python library *NLTK*. At this time, the cosine similarity used is as follows.

$$\text{Similarity} = \frac{T \cdot F}{|T||F|} = \frac{\sum_{i=1}^n T_i \times F_i}{\sqrt{\sum_{i=1}^n (T_i)^2} \times \sqrt{\sum_{i=1}^n (F_i)^2}} \quad (3)$$

In this equation, T represents the tweet of Donald Trump, and F represents the content of fake news. Through these results, the articles related to fake news and Donald Trump’s tweet are as follows. The results confirmed that Donald Trump tweeted to some extent on the spread of fake news. That’s because Twitter can spread its tweet content through a feature called retweet. At this time, cosine similarity is low because the contents of tweet and fake news are different. Therefore, this study compared only 0.3 or more through relative comparison.

The next section compares the sentimental analysis and cosine similarity with the Google trends to see the process of the Russian hacking case and the resulting cyber influence attacks. In particular, check how one of these attacks, the fake news attack, affected Hillary Clinton camp.

4.2 Result

This section analyzes what kind of cyber influence attacks occurred as a case study of the US presidential election in 2016. In addition, we discuss whether Trump’s comments influenced cyber influence attack through sentimental analysis and cosine similarity of Trump tweets.

This study examines how the PIA, LIA, Cyber Influence Attacks, which attack the rational decision-making of voters defined above in connection with changes in the support of Google Trends data during the US presidential election. The US presidential election schedule began with the Republican National Convention starting from the Republican National Convention on July 21, 2016 to the election on December 19, 2016, and ultimately elected the president. The related schedule is shown in Fig. 5 below.

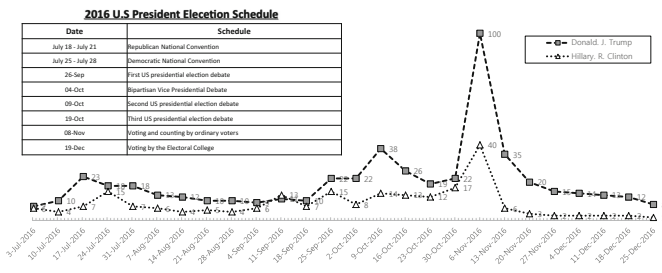


Fig. 5. 2016 U.S. president election schedule and Google trend

Trump is predominantly on the graph, but Hillary was dominant on July 24, 2016 and September 11, 2016, 2016. However, since then, it has shown a decline. From October 2 to October 23, trends for Trump increased while trends for Hillary Clinton decreased. During this period, we can confirm that there is a fake news for Hillary Clinton. At this time, fake news such as “Clinton sold weapons to ISIS” was reproduced, with information on e-mails leaked from online news and community sites [22]. During the same period, information from

Clinton's chief of staff, leaked through the Russian hacking group, was disseminated through Wikileaks. Also, as the presidential election period, December 19, was approaching, Wikileaks not only leaked information from hacking but also increased frequency to fake news.

Table 2. Sentimental analysis of Donald Trump tweet

Type	Date	Tweet	# of Retweet	Compound
A	2016.10.19	'Hillary Clinton Deleted Emails With Her Email Server Technician'	7874	0
B	2016.10.22	Crooked Hillary Clinton Tops Middle East Forums Islamist Money List	9800	0.51
C	2016.11.1	WikiLeaks emails reveal Podesta urging Clinton camp to 'dump' emails.	14312	0
D	2016.10.19	EXCLUSIVE: FBI Agents Say Comey Stood In The Way Of Clinton Email Investigation:	15871	0.303

Type	Publish Date	Fake News Title	Similarity
A	2016.10.29	FBI FOUND "TENS OF THOUSANDS OF EMAILS" BELONGING TO HUMA ABEDIN ON WEINER'S LAPTOP	0.409
B	2016.10.27	Hillary Clinton Tops "Islamist Money in Politics" List	0.3836
C	2016.11.2	Here's your intent! Hillary Intentionally Erased Emails - Wikileaks	0.368
C	2016.11.21	ASSISTANT ATTORNEY GENERAL TIPPED OFF CLINTON CAMP ABOUT DOJ INVESTIGATION	0.349
D	2016.10.31	MUTINY AT THE FBI: COMEY WARNED BY HIS OWN AGENTS TO INDICT CLINTON OR WATCH THE FBI REPUTATION GO DOWN IN FLAMES	0.367

Also, it is possible to confirm that the number of retweets of fake news that show similarity between trump tweets at the same time increases. The unusual point is that instead of negative statements to Hillary Clinton, the number of neutral expressions has a high number of Retweet and favorite. Trump's tweet was updated before the opening of fake news. The relationship between a typical trump tweet and fake news is shown in the following Tables 2 and 3. Fake news, one of the information generated by the Russian hacking, spread through rival Donald Trump. This result has helped Donald Trump to some extent to the cyber influence attack that Russia did not intend or intend. The point is clear.

5 Conclusion

The recent hacking in Russia during the US presidential election has led to the realization that cyber attacks can affect the real world beyond cyberspace.

This study suggests the five vulnerabilities (high uncertainty, lack of precedence, access limited information, and number of similar alternative) of voter's rational decision - making in the democracy electoral system of one vote per person. Through this, this study defined the kinds of cyber influence attacks. As a result, the influence of the Russian hacking incident was confirmed by the change of Google Trend.

In addition, this study analyzed Donald. R. Trump's tweets and measured how Trump's tweet helped cyber influence attack. Sentimental analysis was used for this. In addition, we used cosine similarity to analyze the relationship between fake news and Trump tweet. The reason for using this is because Trump's tweet activity can check whether there is an activity using fake news. As a result, it was confirmed that Trump's tweet behavior helped spread the fake news, one of the cyber influence attacks.

There is a limitation to assess damage of cyber influence attack with an objective indicators, because it depends on the subjective judgment of voters' decision. However, the fact that there is a change in the trend means that cyber influence attacks can affect. However, since the attacks occurred at a politically sensitive time, the response of the US government was passive and, as a result, the possibility of impeachment of the president and the social disorder was reported. Therefore, a de-politicization of cyber security policy is needed to protect the nation's basic system from cyber influence attacks.

Acknowledgment. This work was supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract (UD060048AD).

References

1. Willis, M.: WannaCry: the ransomware attack on the NHS and what we can learn from it, June 2016. <http://blogs.lse.ac.uk/politicsandpolicy/wannacry-the-ransomware-attack-on-the-nhs-and-what-we-can-learn-from-it/>
2. Entous, A., Nakashima, E.: FBI in agreement with CIA that Russia aimed to help Trump win White House, The Washington Post (2016). https://www.washingtonpost.com/politics/clinton-blames-putins-personal-grudge-against-her-for-election-interference/2016/12/16/12f36250-c3be-11e6-8422-eac61c0ef74d_story.html?utm_term=.381f14d62e3d
3. Clauss, K.S.: FBI director James Comey to speak at Boston college, March 2017. <http://www.bostonmagazine.com/news/blog/2017/03/02/fbi-director-james-comey-boston-college-cybersecurity/>
4. Spring, T.: DHS chairman paints bleak US cyber security picture, February 2017. <https://threatpost.com/dhs-chairman-paints-bleak-us-cybersecurity-picture/123739/>
5. WikiLeaks: Vault 7: CIA Hacking Tools Revealed (2017). <https://wikileaks.org/ciav7p1/>
6. Cyberspace Operation JP 3-12R, vol. 3-12R. Joint Chiefs of Staff. Department of Defence (2013)
7. Clark, D.: Characterizing cyberspace: past, present and future. MIT CSAIL, Version 1, 2016-2028 (2010)

8. Barford, P., et al.: Cyber SA: situational awareness for cyber defense. In: Jajodia, S., Liu, P., Swarup, V., Wang, C. (eds.) *Cyber Situational Awareness*. ADIS, vol. 46, pp. 3–13. Springer, Boston (2010). https://doi.org/10.1007/978-1-4419-0140-8_1
9. Philip, R.: *Enabling distributed security in cyberspace*. Department of Homeland Security (2011)
10. Herrick, D.: The social side of ‘cyber power’? Social media and cyber operations. In: 2016 8th International Conference on Cyber Conflict (CyCon), pp. 99–111. IEEE, May 2016
11. Golan, G.J., Lim, J.S.: Third-person effect of ISIS’s recruitment propaganda: online political self-efficacy and social media activism. *Int. J. Commun.* **10**, 21 (2016)
12. Weimann, G.: The emerging role of social media in the recruitment of foreign fighters. In: de Guttery, A., Capone, F., Paulussen, C. (eds.) *Foreign Fighters under International Law and Beyond*, pp. 77–95. T.M.C. Asser Press, The Hague (2016). https://doi.org/10.1007/978-94-6265-099-2_6
13. Amato, F., Cozzolino, G., Mazzeo, A., Romano, S.: Detecting anomalies in Twitter stream for public security issues. In: 2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a Better Tomorrow (RTSI), pp. 1–4. IEEE, September 2016
14. Schumpeter, J.A.: *Capitalism Socialism and Democracy*. Routledge, Abingdon (2013)
15. Pateman, C.: *Participation and Democratic Theory*. Cambridge University Press, Cambridge (1970)
16. Deacon, R.E., Firebaugh, F.M.: *Family Resource Management: Principles and Applications*. Allyn and Bacon, Boston (1981)
17. Doyle, J.: Rational decision making. In: *MIT Encyclopedia of the Cognitive Sciences*, pp. 701–703 (1999)
18. Marchette, D.J.: *Common network attacks*
19. Hansman, S., Hunt, R.: A taxonomy of network and computer attacks. *Comput. Secur.* **24**(1), 31–43 (2005)
20. Pak, A., Paroubek, P.: Twitter as a corpus for sentiment analysis and opinion mining. In: *LREc*, vol. 10, no. 2010, May 2010
21. Muflikhah, L., Baharudin, B.: Document clustering using concept space and cosine similarity measurement. In: 2009 International Conference on Computer Technology and Development, ICCTD 2009, vol. 1, pp. 58–62. IEEE, November 2009
22. Kopan, T.: Wikileaks Reveals Hillary-World Vexed by ‘Bill Clinton Inc.’. *CNN* (2016). <http://edition.cnn.com/2016/10/27/politics/bill-clinton-inc-foundation-wikileaks-emails/>



A Protection Technique for Screen Image-Based Authentication Protocols Utilizing the SetCursorPos Function

Insu Oh¹, Kyungroul Lee², and Kangbin Yim¹(✉)

¹ Department of Information Security Engineering, Soonchunhyang University, Asan, South Korea

{catalyst32, yim}@sch.ac.kr

² R&BD Center for Security and Safety Industries (SSI), Soonchunhyang University, Asan, South Korea
carpedm@sch.ac.kr

Abstract. This paper focuses on security problems of password-based authentication systems and password exposure by login users following image-based authentication protocols requiring a mouse HID. One of these systems consists of on-screen virtual keyboard authentication protocol, which is commonly utilized by Internet banking services and electronic payment services. Nevertheless, this protocol presents the vulnerability of mouse coordinate data exposure through the GetCursorPos() API. Authentication information involving image-based authentication systems is thus still vulnerable to attacker's attacks and theft. Accordingly, we propose a security protection technique that utilizes the SetCursorPos() function to introduce random irrelevant mouse coordinate data.

Keywords: Mouse · Image-based authentication · Windows API
SetCursorPos function

1 Introduction

A mouse, which is a typical Human Interface Device (HID), has long been used with a keyboard. Input devices such as the keyboard and the mouse deliver user commands to a computer system. For example, when a user moves the mouse HID, a cursor on a screen display moves accordingly. The mouse includes several features, including click features that designate specific actions. For example, a user can click a default button of the mouse (left button) once to select a screen element or double click the same button to execute a function. The mouse also supports a scroll feature that utilizes a wheel button to facilitate viewing screen content. As new features are created, buttons can be added to the mouse to enhance user convenience while working and gaming. In this manner, a mouse can send more information to the host system. The data that is transferred from the HID to the host system, which is generally connected via PS/2 or USB interfaces, can be sent in different formats. Input commands are transferred and processed as data from an HID to an operating system through particular control tools, and finally the data is delivered to an application program [1]. Being that a mouse is a

convenient user input device, screen-based authentication protocols have emerged that authenticate users in relation to images displayed on the screen (for example, a virtual numeric grid) and a required pattern of clicks by a user on the mouse to prevent the bypassing of authentication processes that occur when keyboard information is exposed in password-based authentication protocols [2, 10].

Such image-based authentication protocols have emerged to overcome the problem of exposure of authentication information that occurs when keyboard data is stolen by sight at the level of data entry into the keyboard [11–14]. In such scenarios, several problematic vulnerabilities exist, including high-level security attacks that call Windows APIs, the problem which does not prevent mouse loggers to get on the Internet easily. The act of logging in can thus be utilized as an attack tool by exposing mouse coordinates through a simple program that extracts the coordinates of a mouse cursor by way of the `GetCursorPos()` function, one of Windows's APIs. All of this makes clear the close relationship between screen image displays and mouse coordinates as well as the ways that these can be manipulated to compromise security in screen-based authentications. Clearly, when such information is compromised, serious damages may follow. Despite all this, security breaches that occur by way of image-based authentication systems and in tandem with mouse coordinate data are still insufficiently studied. Accordingly, this paper investigates security breach prevention techniques involving mouse coordinate data in relation to screen-based authentication protocols.

2 Related Works

2.1 Screen Image-Based Authentication

Image-based authentication depends on the user clicking on specific coordinates in order to establish authentication. This means that displayed images and mouse-click coordinates must be matched in a specific affinitous relationship for security restrictions to be bypassed. Authentication can be secured according to various maneuvers depending on the arrangement of displayed images in relation to an on-screen virtual keyboard or keypad. The illustration in Fig. 1 demonstrates how the relationship between keyboard information and screen click locations are utilized to achieve authentication [3].

Even though screen image-based authentication systems have been developed to overcome the problem of stolen authentication information by way of compromised keyboard data, the security assessment of these systems have not been carried out sufficiently. The most pressing issues pertain to (1) vulnerabilities to image and mouse coordinate data by way of screen capture tools, and (2) mouse loggers which are easy to get on the Internet and (3) vulnerability to shoulder surfing attack as a social engineering attack [4]. In particular, an attacker may gather mouse movement data as inputted by a user by recording and then reproducing movement coordinates and clicked entries.



Fig. 1. Image-based authentication using an on-screen virtual keyboard

2.2 Mouse Data Exposure Using Windows API

In the Internet banking service, there exists a malware ZeuS which steals the user's online banking account. It attacks Web browsers such as Internet Explorer to remotely control the victim's PC, causing malicious behavior. The malware hooks the victim's PC's Window API and uses the `GetCursorPos()` function to get the mouse data. When users use Internet banking with image-based authentication, they take a screenshot of the infected PC and take the user's account information through the `GetCursorPos()` [5-7].

While attackers can trace mouse movements, more expert attackers may also implement sniffing programs. The Windows operating system provides various APIs to manage and support mouse coordinate data inputted by a user, including `GetCursorPos()`, which reads coordinates along x and y axes [8]. Accordingly, attackers can collect x and y coordinates by periodically tracking the `GetCursorPos()` API and thereby ascertain the mouse movements enacted by users. This API attack scenario is illustrated in Fig. 2.

There are several assumptions when stealing mouse data through `GetCursorPos()`. The attack program must be installed on the victim's PC in advance. If an image-based authentication is entered, the attack program must be running. And the protection program that monitors the Windows API should not work.

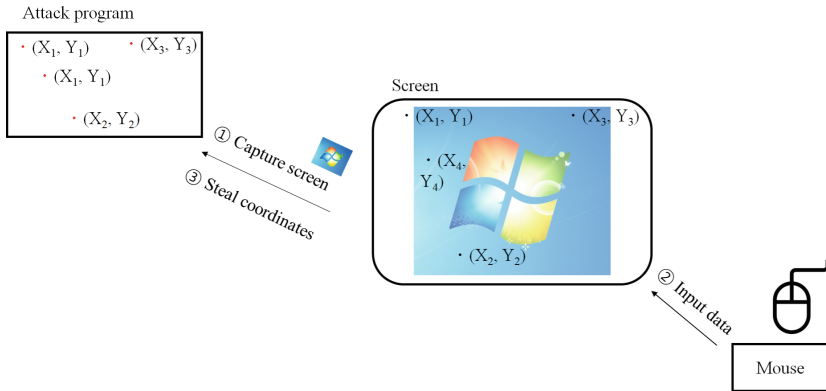


Fig. 2. Attack scenario using the GetCursorPos() function

Step 1. An attack program captures a screen image after obtaining screen size.

Step 2. The attack program extracts mouse periodically draws coordinate data from the GetCursorPos() API to ascertain mouse position as managed by the operating system. Once mouse coordinates are ascertained, linear mouse movements can be determined intuitively.

Step 3. In order to test selected positions on a screen image, the attack program collects click data by utilizing an event handler. Click positions are displayed in red color, and click sequences are displayed in black color. Finally, the attack program determines mouse movement and click data to derive the correct authentication information.

In previous research, we implemented a proof-of-concept tool to perform the operations as illustrated in Fig. 3 and investigated exposure possibilities to mouse data in image-based authentication systems supported by Internet banking services and electronic payment services in South Korea. We found that most mouse data utilized in these services were exposable, meaning that authentication information could also be exposed. Like the illustration in Fig. 3, mouse movement information and the order of inputted password information was obtainable from virtual keyboards, thereby enabling an attacker to steal a real password.

As shown in above Fig. 3, image-based authentications can neutralize the risk of API manipulations as enabled by the operating system. A more serious problem occurs when authentication data is compromised through image-based protocols in the realm of financial services, contributing to financial theft. Unfortunately, countermeasures are not enough to detect and prevent this vulnerability. This paper therefore proposes a prevention technique for exposing mouse data.



Fig. 3. Stealing mouse data by drawing from the GetCursorPos() function (Color figure online)

3 Proposed Prevention Technique

In what follows, we investigate the neutralization of image-based authentications that occur when mouse movements are traced by drawing on the GetCursorPos() API. The problem arises insofar as operating systems can expose the GetCursorPos() function and, thereby, the mouse coordinates. There are two distinct approaches for overcoming this problem. The first method works by preventing mouse movements, and the second consists of tracing the actual mouse coordinates and then generating irrelevant mouse data. This paper expands on the second prevention technique; that is, the generation of random mouse coordinates, as illustrated in Fig. 4.

Since the Windows operating system generates mouse position data, it is possible to generate random irrelevant mouse positions data through the SetCursorPos() function [9]. In generating random x and y coordinates with a security enhancing program, an attack program would extract both the real and the irrelevant coordinates. As far as the attack program is concerned, all of the gathered coordinates have been inputted through real mouse movements. There is no way to distinguish the real x and y coordinates from the irrelevant ones. Nevertheless, the prevention program itself will distinguish the randomly generated coordinates from the real ones, being that it is designed for this very purpose. This distinction-making process is continuously performed to protect the input data.

In scenarios where the security prevention program is generating random coordinates as a user actually moves the mouse, the operating system continues to recognize the actual HID data and moves the screen cursor position accurately. In the case that the coordinate data is compromised by an attack program, the latter can only assume that

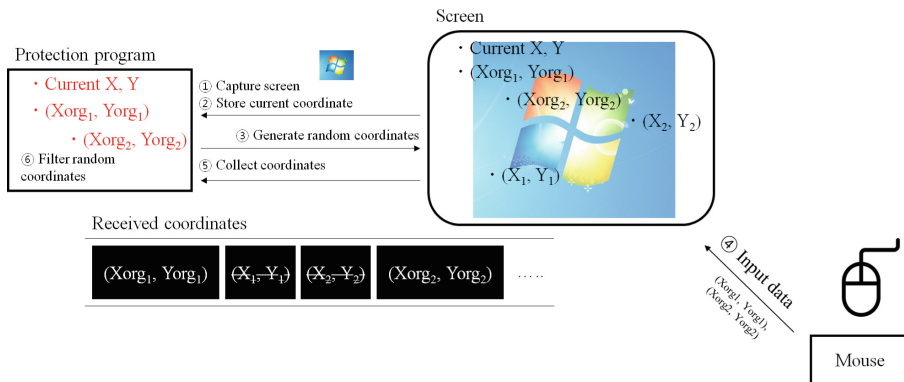


Fig. 4. The operation process of the proposed prevention technique

all of the coordinates (actual and irrelevant) have been inputted in relation to the mouse movements generated by a user. On the other hand, the prevention program distinguishes coordinate positions generated by the user from those generated by the prevention program itself. This means that only the prevention program can determine the randomly generated coordinates and subsequently filter them so that the operating system ultimately responds to the real movements made on the mouse. However, the attack program collects generated random coordinates and inputted coordinates from the user, so it is possible to disturb the authentication information because there is no way to distinguish between collected coordinates. The procedures for this proposed security enhancing technique is detailed in what follows.

Step 1. The designated screen image for user password selection is displayed.

Step 2. The prevention program stores the real mouse coordinates that determine the continuous real cursor positions while ignoring randomly program generated coordinates.

Step 3. The randomly generated coordinate data is mingled with the real authentication data to be provided to any attack program. Ultimately, the actually intended mouse position data is decidedly proffered to enact the SetCursorPos() function. In the case of a GetCursorPos() API data breach, the attack program will extract the randomly generated coordinates, thereby obscuring the real mouse movement coordinates.

Step 4. The prevention program makes use of mouse position data only when it matches the data of actually designated positions. All other data (that which is not generated through the mouse HID) is filtered out. This means that only mouse position coordinates that do not designate randomly generated cursor positions are actually inputted for execution. The prevention program then intuitively generates linear cursor movements. The prevention program then displays click positions in red color and clicked sequences in black color.

Ultimately, the proposed prevention technique protects mouse data from malicious hacker attacks. This process occurs in parallel and is illustrated in Fig. 5.

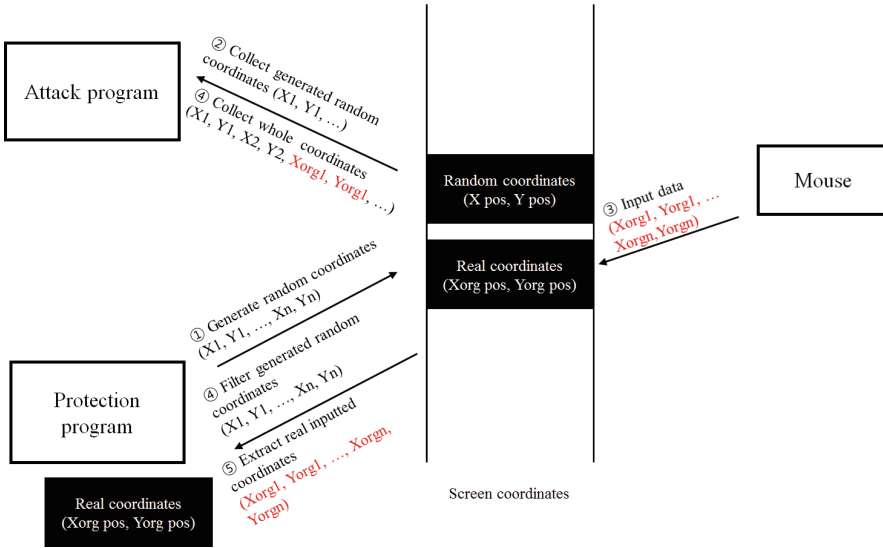


Fig. 5. The mouse data protection process of protection program to prevent GetCursorPos()

4 Experiment Results

In this paper, we summarize the implementation of a proposed prevention technique, investigating the exposure of mouse movements by attack programs that rely on stolen GetCursorPos() API data. Figure 6 shows how this mouse movement data would appear in the case of an infiltration by an attack program but where the prevention program is running. In such instances, hackers will not be able to ascertain real mouse movements and thereby steal authentication information because clicked positions are displayed regardless of the image.

As previously noted, a prevention program can still trace mouse movements as designated by user input because all randomly generated coordinates are filtered out. The random coordinates generated by the protection program are filtered and are less likely to interfere when the user enters the password. Figure 7 illustrates the way that stolen data would appear to a hacker. The left figure shows user-generated mouse movement and click information data while the prevention program is running. The right column demonstrates how this same data would be proffered to an attack program. The user can use the virtual keyboard while the protection program is running. Ultimately, the prevention program securely gathers user-generated authentication information, preventing the exposure of real mouse movements and click information to an attack program. Figure 8 shows the randomly order generated coordinates. The proposed prevention technique thus improves security of image-based authentication systems used by numerous organizations that require secure authentication protocols, including Internet banking services and electronic payment services.

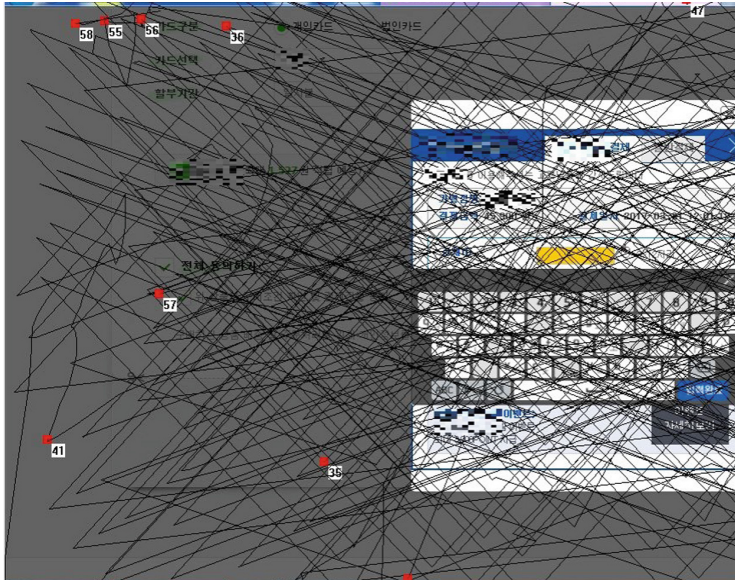


Fig. 6. Attack is not possible with mouse movement and click information data extracted by attack program

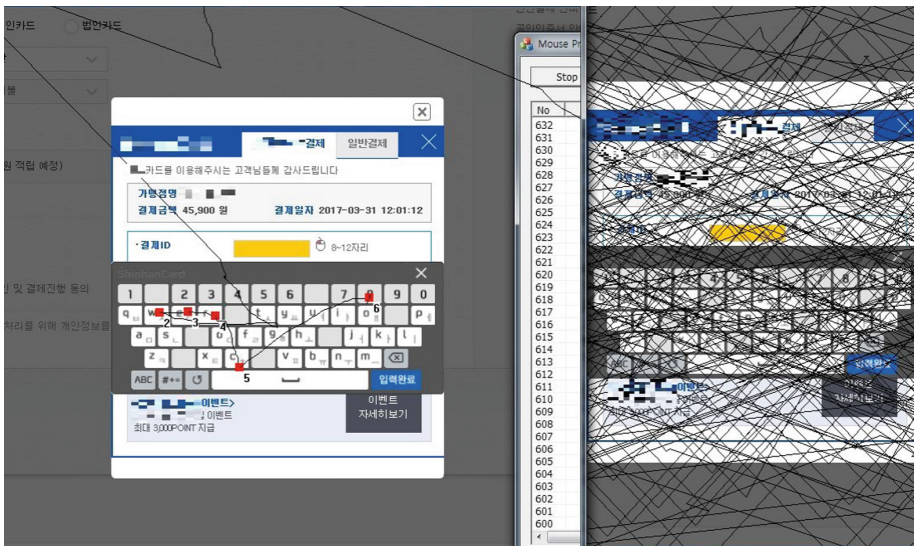


Fig. 7. A user who normally uses a virtual keyboard and an attacker who can't take a password by protection program

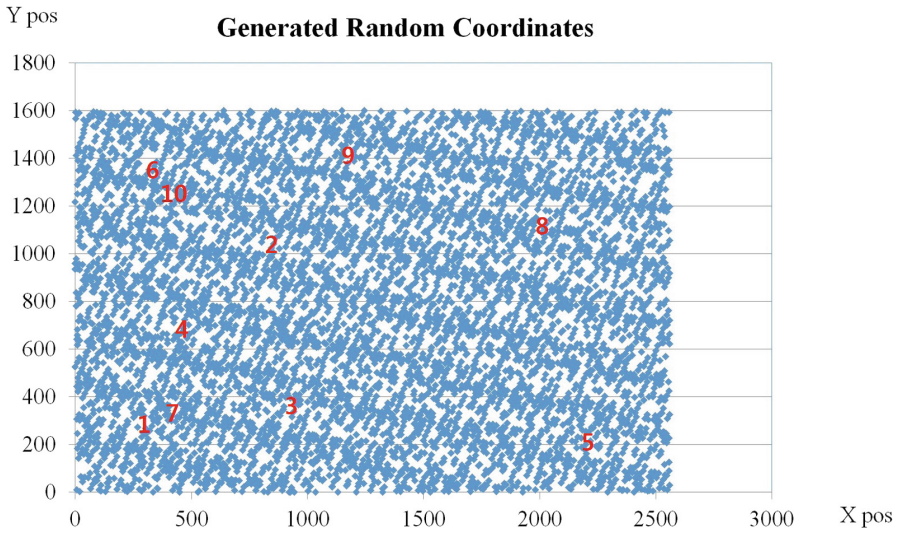


Fig. 8. Generated random order coordinates

5 Conclusion

In this paper, we proposed a prevention technique to improve the security of the image-based authentication method which emerged to overcome a vulnerability of the keyboard-based authentication method. The keyboard data is exposed by key loggers, and the image-based authentication has a vulnerability of exposing the displayed screen information and the mouse data. Moreover, the internet banking service in South Korea utilized the image-based authentication to improve the security. Nevertheless, the authentication information is exposed to several services and verified the vulnerability.

Especially, the mouse data and movements are exposed by mouse loggers and an implemented simple attack program only just calling windows API. For this reason, there is vulnerability that the authentication information is exposed. Hence, prevention and protection techniques are required to counteract this vulnerability, so we proposed a prevention technique by generating random mouse coordinates without exposing real mouse movements. The proposed technique is safe, so we consider that the technique can protect the authentication information safely for various services applied the image-based authentication.

However, the proposed technique is a countermeasure at the application level, so it is possible to expose the mouse data to steal low-level attacks such as operating system level and hardware level. In future works, we will research to counteract this problem and to improve the security.

Acknowledgments. This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) that is funded by the Ministry of Education (NRF-2015R1D1A1A01057300).

References

1. Wikipedia: Computer Mouse. https://en.wikipedia.org/wiki/Computer_mouse
2. Dadkhah, M., Jazi, M.D.: A novel approach to deal with keyloggers. *J. Comput. Sci. Technol.* 7(1), 25–28 (2014)
3. Wikipedia: Virtual Keyboard. https://en.wikipedia.org/wiki/Virtual_keyboard
4. Parekh, A., Pawar, A., Munot, P., Mantri, P.: Secure authentication using anti-screenshot virtual keyboard. *J. Comput. Sci. Issues* 8(5), 534–537 (2011)
5. Braschi, A., Continella, A.: Prometheus: A Web-based Platform for Analyzing Banking Trojans (2014)
6. Aditya, S., Rohit, B.: Prosecting the citadel botnet revealing the dominance of the zeus descendent (2014)
7. O’Murchu, L., Gutierrez, F.P.: The evolution of the fileless clickfraud malware Poweliks (2015)
8. MSDN: GetCursorPos function. [https://msdn.microsoft.com/ko-kr/library/windows/desktop/ms648390\(v=vs.85\).aspx](https://msdn.microsoft.com/ko-kr/library/windows/desktop/ms648390(v=vs.85).aspx)
9. MSDN: SetCursorPos function. [https://msdn.microsoft.com/ko-kr/library/windows/desktop/ms648394\(v=vs.85\).aspx](https://msdn.microsoft.com/ko-kr/library/windows/desktop/ms648394(v=vs.85).aspx)
10. Lee, H., Lee, Y., Lee, K., Yim, K.: Security assessment on the mouse data using mouse loggers. *Advances on Broad-Band Wireless Computing, Communication and Applications. LNDECT*, vol. 2, pp. 387–393. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-49106-6_37
11. Lee, S., Lee, K., Yim, K.: Security assessment of keyboard data based on Kaspersky product. *Advances on Broad-Band Wireless Computing, Communication and Applications. LNDECT*, vol. 2, pp. 395–400. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-49106-6_38
12. Lee, K., Yim, K.: Keyboard security: a technological review. In: *Proceedings of the Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, pp. 9–15, June 2011
13. Lee, K., Choi, Y., Yeuk, H., Yim, K.: Password sniff by forcing the keyboard to replay scan codes. In: *Proceedings of the Joint Workshop on Information Security (JWIS)*, p. 9, August 2010
14. Lee, K., Bae, K., Yim, K.: Hardware approach to solving password exposure problem through keyboard sniff. In: *Proceedings of the Academic Science Research, WASET*, pp. 23–25, October 2009

Crypto Protocols



A General Two-Server Cryptosystem Supporting Complex Queries

Sha Ma^(✉) and Yunhao Ling

College of Mathematics and Informatics, South China Agricultural University,
Guangzhou, Guangdong, China
shamahb@163.com, ftd250785267@163.com

Abstract. In the era of cloud computing, searchable encryption is an essential technology to provide security measure to protect the outsource data security and meanwhile support the desired computation on the ciphertexts. In this paper, we focus on the following cases: if the plaintext messages are considered as integers, given the ciphertexts of M_1 and M_2 , how to enable the server to test (1) whether $aM_1 + bM_2 + c = 0$, (2) whether $M_1^a M_2^b c = 1$, where a, b and c are integers. Under the extension, this equation queries could be used as a building block for range join queries on encrypted data. In order to overcome offline message guessing attack as an inherent vulnerability of searchable encryption, we consider the setting of two non-colluded servers and propose a general public-key cryptosystem based on smooth projective hash function (SPHF) with linear and homomorphic properties. Thanks to the efficient SPHF instantiations without any pairing, our scheme would have many interesting applications.

Keywords: Searchable encryption · Offline message guessing attack
Smooth projective hash function

1 Introduction

In the era of cloud computing, searchable encryption is an essential technology to provide security measure to protect the outsource data security and meanwhile support the desired computation on the ciphertexts. Public key encryption with keyword search (PEKS) [1, 2] is a typical and well-studied searchable encryption to check whether the message under a ciphertext is equal to the keyword hidden in the trapdoor. If the keywords are chosen from a much smaller space, anyone can generate a ciphertext for the guessing keyword and then test whether a given trapdoor contains the keyword using the *Test* algorithm. A few work [3–8] has solved this off-line keyword guessing attack to resist outside adversary. However, it is generally believed that the off-line keyword guessing attack against

This work is supported by the National Natural Science Foundation of China (No. 61402184).

inside adversary is an inherent vulnerability of PEKS. Public key encryption with equality test (PKEET) is another type of searchable encryption, which was first proposed by Yang *et al.* [9] in order to check whether two ciphertexts share the same underlying plaintext. According to the functionality, PEKS could be viewed as a special type of PKEET since the latter supports checking not only the plain keyword but also the encrypted keyword. Similar to the off-line keyword guessing attack in PEKS, the off-line message recover attack is also an inherent vulnerability of PKEET because anyone can totally recover the plaintext under the ciphertext by encrypting the guessing message and then running the **Test** algorithm to see if the ciphertext contains the guessing message. In fact, the traditional searchable encryption in the single-server framework might be impossible to resist this attack.

Recently, Chen *et al.* [8,10] proposed a new framework of DS-PEKS (dual-server PEKS) to overcome this inherent vulnerability of searchable encryption. In the DS-PEKS scheme, the encryption algorithm is the same as the trapdoor generation algorithm, which reminds us of the possibility of comparison between the ciphertexts themselves. Furthermore, if the encryption of the hash of M in DS-PEKS is replaced by the encryption of a function of M , there is manipulation possible for the tester to perform certain functions using the linear and homomorphic properties of SPHF. Inspired by DS-PEKS [8], we also use two servers (a front server and a back server) to solve the problem of offline message attack meanwhile achieving the functionalities. Compared with [8], the main difference in this paper is:

DS-PEKS is for the keyword search on the ciphertexts using a trapdoor. Our construction is for testing some equations on the ciphertexts when the plaintext messages are considered as integers, which includes the equality test on ciphertext supported by PKEET.

In the related work, [11] also provided a PKEET scheme in the two-server setting. We observe that it has the following security and efficiency concerns:

1. For the token generation, a trusted party needs to take the two users' private keys as input, which would increase the risk of key leakage.
2. In the final step of comparison, an interactive protocol is performed between the two servers. It needs either a trusted proxy, which might be unrealistic in some applications, or multiple communications between these two servers, which results in much communication cost.

1.1 Our Contributions

In this paper, based on the work [8,10] we propose a general public-key cryptosystem supporting some complex queries between the ciphertexts. In brief, our contributions are summarized as follows:

1. Under the different user-defined functions, the equation queries on the ciphertexts of M_1 and M_2 at least include the following two cases: (1) $aM_1 + bM_2 + c = 0$ and (2) $M_1^a M_2^b c = 1$, where a, b and c are integers.

2. The predicate $p(M_1, M_2) := (v_1 \leq (aM_1 + bM_2) \leq v_2)$ can be implemented on the equation queries, which is the building block for constructing the range join on encrypted data.
3. We provide the efficient scheme based on the SPHF instantiation from the DDH assumption while most PEKS or PKEET schemes requires pairing operations.

2 Preliminaries

2.1 Smooth Projective Hash Function

Smooth projective hash function (SPHF) was first introduced by Cramer and Shoup [12] for constructing encryption scheme. A projective hashing family is a family of hash functions that can be evaluated in two ways giving the same result: one can compute the function on each value in the domain using the secret hashing key, whereas one can only compute the function on the values in a special subset of the domain using the public projective hash key. The security of an SPHF is defined through two notions:

1. *Smoothness*: The hash value of any point outside the special subset is statistically indistinguishable from a random element.
2. *Pseudo-randomness*: The hash value of any point inside the special subset is computational indistinguishable from a random element without the knowledge of an evidence of the point.

Let these functions are from \mathcal{X} into \mathcal{Y} and \mathcal{L} be a language as a certain subset of the domain \mathcal{X} . An SPHF system on \mathcal{L} is defined by five algorithms:

- $\text{SPHFSetup}(1^k)$: It generates the global parameters param .
- $\text{HashKG}(\mathcal{L}, \text{param})$: It generates a hashing key hk .
- $\text{ProjKG}(\text{hk}, (\mathcal{L}, \text{param}), W)$: It derives the projection key hp , possibly depending on the word W .
- $\text{Hash}(\text{hk}, (\mathcal{L}, \text{param}), W)$: It outputs the hash value from the hashing key on any word $W \in \mathcal{X}$.
- $\text{ProjHash}(\text{hp}, (\mathcal{L}, \text{param}), W, w)$: It outputs the hash value from the projection key hp and the witness w for the word $W \in \mathcal{L}$.

2.2 Linear and Homomorphic Properties

In this paper, we need a variable SPHF with linear and homomorphic properties, which has been proposed in [8]. Let \mathcal{WS} be the witness space of \mathcal{L} . The operations on the set $(\mathcal{L}, \mathcal{Y}, \mathcal{WS})$ are described as follows.

1. $\odot : \mathcal{L} \times \mathcal{L} \rightarrow \mathcal{L}. \forall W_1, W_2 \in \mathcal{L}, W_1 \odot W_2 \in \mathcal{L}$.
2. $\bullet : \mathcal{WS} \times \mathcal{L} \rightarrow \mathcal{L}. \forall w \in \mathcal{WS}, \forall W \in \mathcal{L}, w \bullet W \in \mathcal{L}$.
3. $\boxplus, \boxtimes : \forall w_1, w_2 \in \mathcal{WS}, w_1 \boxplus w_2 \in \mathcal{WS}$ and $w_1 \boxtimes w_2 \in \mathcal{WS}$.

4. $*$: $\mathcal{Y} \times \mathcal{Y} \rightarrow \mathcal{Y}$. $\forall y_1 \in \mathcal{Y}, \forall y_2 \in \mathcal{Y}, y_1 * y_2 \in \mathcal{Y}$.
 5. \circ : $\mathcal{WS} \times \mathcal{Y} \rightarrow \mathcal{Y}$. $\forall w \in \mathcal{WS}, \forall y \in \mathcal{Y}, w \circ y \in \mathcal{Y}$.

Moreover, for any element $y \in \mathcal{Y}$, we define $y * y^{-1} = 1_{\mathcal{Y}}$, which is the identity element of \mathcal{Y} .

Definition 1 (Linear and homomorphic SPHF). *Assume the underlying language is also linear and homomorphic language. A linear and homomorphic SPHF has the following properties:*

1. $\forall \Delta w \in \mathcal{WS}, \forall W \in \mathcal{L}$, we have

$$\text{Hash}(\text{hk}, \Delta w \bullet W) = \Delta w \circ \text{Hash}(\text{hk}, W)$$

In other words, suppose w is the witness of W , we have

$$\text{ProjHash}(\text{hp}, \Delta w \bullet W, \Delta w \boxtimes w) = \Delta w \circ \text{ProjHash}(\text{hp}, W, w)$$

2. $\forall W_1, W_2 \in \mathcal{L}$, we have

$$\text{Hash}(\text{hk}, W_1 \odot W_2) = \text{Hash}(\text{hk}, W_1) * \text{Hash}(\text{hk}, W_2)$$

In other words, suppose w_1 and w_2 are the witnesses of W_1 and W_2 , respectively, we have

$$\text{ProjHash}(\text{hp}, W_1 \odot W_2, w_1 \boxplus w_2) = \text{ProjHash}(\text{hp}, W_1, w_1) * \text{ProjHash}(\text{hp}, W_2, w_2)$$

3 Definition

3.1 Binary-Equation Cryptosystem

In a binary-equation cryptosystem, two non-colluded servers (front server and back server) could check certain binary equation on ciphertexts C_1 and C_2 .

Definition 2 (Binary-Equation Cryptosystem). *A binary-equation cryptosystem (BEC) is defined by the four algorithms (KeyGen, Enc, FSTest, BSTest):*

- **KeyGen**(k): *It takes the security parameter k as input and generates the public/secret key (pk_{FS}, sk_{FS}) of the front server and the public/secret key (pk_{BS}, sk_{BS}) of the back server.*
- **Enc**(pk_{FS}, pk_{BS}, M): *It takes as input the public keys (pk_{FS}, pk_{BS}) of the front server and the back server, and outputs a ciphertext C of the plaintext M .*
- **FSTest**(sk_{FS}, C_1, C_2, f): *It takes as input the secret key sk_{FS} of the front server, two ciphertexts (C_1, C_2) and a certain binary equation $f(x, y) = 0$, and outputs an internal value IV .*
- **BSTest**(sk_{BS}, IV): *It takes as input the secret key sk_{BS} of the back server and an internal value IV from the front server, and outputs a result 1, indicating that C_1 and C_2 share the same message; otherwise 0.*

Correctness: This cryptosystem knows if the plaintexts M_1 and M_2 underneath C_1 and C_2 satisfy the binary equation $f(M_1, M_2) = 0$.

3.2 Security Models

According to the \mathcal{BEC} security, it should satisfy the following properties for the binary equation $f(x, y) = 0$:

- **Semantic security against chosen keyword attack-I (SS-CKA-I):** Semantic security against chosen keyword attack-I guarantees that the adversary (as the role of the back server) cannot distinguish a keyword from another one given the corresponding ciphertext. That is, the BEC ciphertext does not reveal any information about the underlying keyword to this adversary even under such attack.
 - **Setup.** The challenger runs the $\text{KeyGen}(k)$ algorithm using the security parameter k to generate $(pk_{FS}, pk_{BS}, sk_{FS}, sk_{BS})$. It gives $(pk_{FS}, pk_{BS}, sk_{BS})$ to the adversary \mathcal{A} .
 - **Challenge.** The adversary \mathcal{A} sends the challenger two plaintexts M_0 and M_1 on which it wishes to be challenged. The challenger picks a random $b \in \{0, 1\}$ and gives the adversary $C^* = (W^*, U^*) = \text{Enc}(pk_{FS}, pk_{BS}, M_b)$. We refer to C^* as the challenge ciphertext.
 - **Guess.** Eventually, \mathcal{A} outputs $b' \in \{0, 1\}$ and wins the game if $b = b'$.

We refer to such an adversary \mathcal{A} in the above game as an SS-CKA-I adversary and define its advantage as

$$\text{Adv}_{\mathcal{BCE}, \mathcal{A}}^{\text{SS-CKA-I}}(k) = \Pr[b = b'] - 1/2. \tag{1}$$

- **Semantic security against chosen keyword attack-II (SS-CKA-II):** Here the semantic security against chosen keyword attack-II is the same as the one against the back server except that the challenger as the role of the front server sends $(pk_{FS}, pk_{BS}, sk_{FS})$ to the adversary in stead of $(pk_{FS}, pk_{BS}, sk_{BS})$. We omit the details here. We refer to such an adversary \mathcal{A} as an SS-CKA-II adversary and define its advantage as

$$\text{Adv}_{\mathcal{BCE}, \mathcal{A}}^{\text{SS-CKA-II}}(k) = \Pr[b = b'] - 1/2. \tag{2}$$

It should be noted that both SS-CKA-I and SS-CKA-II are weaker than that in [8] since the adversary is no longer provided an oracle of test query, which takes a PEKS ciphertext and a keyword as input and outputs a decision about whether the ciphertext is the encryption of keyword.

- **Semantic security against chosen keyword attack-III (SS-CKA-III):** Here the semantic security against chosen keyword attack-III is defined to capture the requirement that the back server cannot learn any information about the keyword from the intermediate results.
 - **Setup.** The challenger runs the $\text{KeyGen}(k)$ algorithm using the security parameter k to generate $(pk_{FS}, pk_{BS}, sk_{FS}, sk_{BS})$. It gives $(pk_{FS}, pk_{BS}, sk_{BS})$ to the adversary \mathcal{A} .
 - **Challenge.** The adversary \mathcal{A} sends the challenger three plaintexts M_0 , M_1 and M_2 on which it wishes to be challenged. The challenger randomly

picks $b_1, b_2 \in \{0, 1, 2\}$ (It should be noted that b_1 and b_2 might be the same or not) and computes

$$\begin{aligned} C_1^* &= \text{Enc}(pk_{FS}, pk_{BS}, M_{b_1}), \\ C_2^* &= \text{Enc}(pk_{FS}, pk_{BS}, M_{b_2}), \\ IV^* &= \text{FrontTest}(sk_{FS}, C_1^*, C_2^*, f). \end{aligned}$$

The challenger sends IV^* to \mathcal{A} .

- **Guess.** Eventually, \mathcal{A} outputs $b'_1, b'_2 \in \{0, 1, 2\}$ and wins the game if $\{b_1, b_2\} = \{b'_1, b'_2\}$.

We refer to such an adversary \mathcal{A} as the role of the back server in the above game as an SS-CKA-III adversary and define its advantage as

$$\mathbf{Adv}_{\mathcal{BCE}, \mathcal{A}}^{\text{SS-CKA-III}}(k) = \Pr[b = b'] - 1/3. \quad (3)$$

4 Proposed BEC Scheme

4.1 General Construction

Let the language \mathcal{L} be hard-partitioned subset. Let SPHF = (SPHFSetup, HashKG, ProjKG, Hash, ProjHash) be an SPHF defined on $\mathcal{X} \rightarrow \mathcal{Y}$ for the language \mathcal{L} under the security parameter k . We build a general BEC scheme from such SPHF. Let a mapping function h defined on $\mathcal{M} \rightarrow \mathcal{Y}$, where \mathcal{M} is the message space and a binary function $f(x, y)$ defined on $\mathcal{Y} \times \mathcal{Y} \rightarrow \mathcal{Y}$:

$$f(x, y) = (a \circ x) * (b \circ y) * (c \circ h(q_c)), \quad (4)$$

where a, b and c are the values in the witness space \mathcal{WS} of SPHF, \circ and $*$ are two operators defined in Sect. 2.2 and q_c is a constant value in \mathcal{M} . We present a general construction of $\mathcal{BCE} = (\text{Setup}, \text{Enc}, \text{FSTest}, \text{BSTest})$ for the binary equation f :

$$f(x, y) = (a \circ x) * (b \circ y) * (c \circ h(q_c)) = 1_{\mathcal{Y}}, \quad (5)$$

where $1_{\mathcal{Y}}$ is the identity element of \mathcal{Y} .

- **KeyGen(k):** It generates the global parameters param using the SPHFSetup(k) algorithm of SPHF and then the public/secret keys (pk_{FS}, sk_{FS}) and (pk_{BS}, sk_{BS}) using the HashKG and ProjKG algorithms of SPHF for the front server and back server, respectively.

$$\begin{aligned} sk_{FS} &= \text{HashKG}(\mathcal{L}, \text{param}), \quad pk_{FS} = \text{ProjKG}(sk_{FS}, (\mathcal{L}, \text{param})). \\ sk_{BS} &= \text{HashKG}(\mathcal{L}, \text{param}), \quad pk_{BS} = \text{ProjKG}(sk_{BS}, (\mathcal{L}, \text{param})). \end{aligned}$$

- $\text{Enc}(pk_{FS}, pk_{BS}, M)$: It randomly picks a word $W \in \mathcal{X}$ with the witness w and computes

$$C = (W, U) = (W, \text{ProjHash}(pk_{FS}, W, w) * \text{ProjHash}(pk_{BS}, W, w) * h(M)).$$

Finally, it outputs a BEC ciphertext $C = (W, U)$ of the plaintext M . Note that we will omit $(\mathcal{L}, \text{param})$ as input of the algorithm in this paper for simplicity.

- ★ Assume that the ciphertexts C_1 and C_2 are the encryptions of M_1 and M_2 , respectively.

$$\begin{aligned} C_1 &= (W_1, U_1) \\ &= (W_1, \text{ProjHash}(pk_{FS}, W_1, w_1) * \text{ProjHash}(pk_{BS}, W_1, w_1) * h(M_1)), \\ C_2 &= (W_2, U_2) \\ &= (W_2, \text{ProjHash}(pk_{FS}, W_2, w_2) * \text{ProjHash}(pk_{BS}, W_2, w_2) * h(M_2)). \end{aligned}$$

- $\text{FSTest}(sk_{FS}, C_1, C_2, f)$: For $C_1 = (W_1, U_1)$ and $C_2 = (W_2, U_2)$, it randomly picks $\Delta w \in \mathcal{WT}$ and computes for $f(M_1, M_2) = 1$ as follows.

$$\begin{aligned} W &= (a \bullet W_1) \odot (b \bullet W_2) \\ C &= (a \circ U_1) * (b \circ U_2) * (c \circ h(q_c)) * \text{Hash}(sk_{FS}, W)^{-1} \\ W' &= \Delta w \bullet W \\ C' &= \Delta w \circ C \end{aligned}$$

Finally, it sends $IV = (W', C')$ to the back sever.

- $\text{BSTest}(sk_{BS}, IV)$: For $IV = (W', C')$, it uses its secret key sk_{BS} to check

$$C' \stackrel{?}{=} \text{Hash}(sk_{BS}, W'). \quad (6)$$

If Eq. (6) holds, it outputs 1 indicating M_1 and M_2 satisfy f , otherwise outputs 0.

Correctness. It is easy to verify the correctness.

$$\begin{aligned} \text{Hash}(sk_{FS}, W)^{-1} &= \text{Hash}(sk_{FS}, (a \bullet W_1) \odot (b \bullet W_2))^{-1} \\ &= (a \circ \text{Hash}(sk_{FS}, W_1))^{-1} * (b \circ \text{Hash}(sk_{FS}, W_2))^{-1} \end{aligned}$$

$$\begin{aligned} C &= (a \circ U_1) * (b \circ U_2) * (c \circ h(q_c)) * \text{Hash}(sk_{FS}, W)^{-1} \\ &= (a \circ (\text{Hash}(sk_{FS}, W_1) * \text{Hash}(sk_{BS}, W_1)) * h(M_1)) * (b \circ (\text{Hash}(sk_{FS}, W_2) * \\ &\quad \text{Hash}(sk_{BS}, W_2)) * h(M_2)) * (c \circ h(q_c)) * \text{Hash}(sk_{FS}, W)^{-1} \\ &= (a \circ \text{Hash}(sk_{BS}, W_1)) * (b \circ (\text{Hash}(sk_{FS}, W_2))) * (a \circ h(M_1)) * (b \circ h(M_2)) * (c \circ h(q_c)) \end{aligned}$$

If $(a \circ h(M_1)) * (b \circ h(M_2)) * (c \circ h(q_c)) = 1_{\mathcal{Y}}$ then

$$\begin{aligned} C &= (a \circ \text{Hash}(sk_{BS}, W_1)) * (b \circ (\text{Hash}(sk_{BS}, W_2))) \\ &= \text{Hash}(sk_{BS}, a \bullet W_1) * \text{Hash}(sk_{BS}, b \bullet W_2) \\ &= \text{Hash}(sk_{BS}, (a \bullet W_1) \odot (b \bullet W_2)) \end{aligned}$$

Therefore,

$$\begin{aligned}
C' &= \Delta w \circ C \\
&= \Delta w \circ (\text{Hash}(sk_{BS}, (a \bullet W_1) \odot (b \bullet W_2))) \\
&= \text{Hash}(sk_{BS}, \Delta w \bullet ((a \bullet W_1) \odot (b \bullet W_2))) \\
&= \text{Hash}(sk_{BS}, \Delta w \bullet W) \\
&= \text{Hash}(sk_{BS}, W').
\end{aligned}$$

Otherwise, if M_1 and M_2 do not satisfy f , $C' \neq \text{Hash}(sk_{BS}, W')$.

4.2 Security Proof

The following theorems show the \mathcal{BEC} scheme satisfies SS-CKA-I, SS-CKA-II and SS-CKA-III security based on the pseudo-randomness of SPHF. The proofs of Theorems 1 and 2 are similar to the proofs of Lemmas 2 and 1 in [8] except without emulating the oracle of test query. The proof of Theorem 3 is the same as the proof of Lemma 5 in [8]. Therefore, we omit their details for brevity.

Theorem 1. *\mathcal{BEC} satisfies SS-CKA-I security based on the pseudo-randomness property of SPHF.*

Theorem 2. *\mathcal{BEC} satisfies SS-CKA-II security based on the pseudo-randomness property of SPHF.*

Theorem 3. *\mathcal{BEC} satisfies SS-CKA-III security based on the pseudo-randomness property of SPHF.*

4.3 Extension

Our construction could be used as a building block for the predicate p like:

$$p(x, y) := (v_1 \leq (ax + by) \leq v_2), \quad (7)$$

where a, b, x, y are integers. This property could be very useful and necessary in some applications. A scenario of this predicate is the following range join query, which joins the raised salary (2 times the current salary) of the employees in department A with the salary of employees in department B. The join condition is that the difference between the new salary of employees in department A with the salary of employees in department B is greater than 100 and smaller than 500:

```

SELECT depA.salary, depB.salary
FROM depA, depB
WHERE (2depA.salary-depB.salary)>=300 AND
(2depA.salary-depB.salary)<=500

```

Executing such a query on encrypted data can be achieved efficiently by just testing whether

$$2M_1 - M_2 = t \text{ for any integer } t \in [300, 500].$$

5 Instantiated BCE Schemes

First we recall an efficient SPHF instantiation from the DDH assumption, which has the linear and homomorphic properties (See Theorem 4).

5.1 An Instantiation of the Linear and Homomorphic SPHF_{DDH}

1. Setup(1^k): $\text{param} = (\mathbb{G}, p, g_1, g_2)$;
2. HashKG(\mathcal{L}_{DDH} , param): $\text{hk} = (s_1, s_2) \xleftarrow{\$} \mathbb{Z}_p^2$;
3. ProjKG(hk , $(\mathcal{L}_{DDH}, \text{param})$): $\text{hp} = g_1^{s_1} g_2^{s_2} \in \mathbb{G}_p$;
4. Hash(hk , $(\mathcal{L}_{DDH}, \text{param})$, $W = (g_1^r, g_2^r)$): $\text{hv} = g_1^{rs_1} g_2^{rs_2} \in \mathbb{G}_p$;
5. ProjHash(hp , $(\mathcal{L}_{DDH}, \text{param})$, $W = (g_1^r, g_2^r)$, $w = r$): $\text{hv}' = \text{hp}^r \in \mathbb{G}_p$

As shown in [8], we have the following theorems for SPHF_{DDH}.

Theorem 4. SPHF_{DDH} is a smooth projective hash function with the linear and homomorphic properties.

Proof. We show that SPHF_{DDH} has the linear and homomorphic properties:

1. For a word $W = (g_1^r, g_2^r)$ with the witness $w = r \in \mathbb{Z}_p$, and $\Delta w \in \mathbb{Z}_p$, we have

$$\begin{aligned} \text{Hash}(\text{hk}, \Delta w \bullet W) &= (g_1^{\Delta wr})^{s_1} (g_2^{\Delta wr})^{s_2} = (g_1^{rs_1} g_2^{rs_2})^{\Delta w} \\ &= \Delta w \circ \text{Hash}(\text{hk}, W) \end{aligned}$$

2. For words $W = (g_1^{r_1}, g_2^{r_1})$ with the witness $w_1 = r_1 \in \mathbb{Z}_p$ and $w_2 = r_2 \in \mathbb{Z}_p$, respectively, we have

$$\begin{aligned} \text{Hash}(\text{hk}, W_1 \odot W_2) &= (g_1^{r_1+r_2})^{s_1} (g_2^{r_1+r_2})^{s_2} = (g_1^{r_1 s_1} g_2^{r_1 s_2}) (g_1^{r_2 s_1} g_2^{r_2 s_2}) \\ &= \text{Hash}(\text{hk}, W_1) * \text{Hash}(\text{hk}, W_2) \end{aligned}$$

Therefore, this theorem is proven.

Based on the SPHF_{DDH}, we choose two types of binary equation (f_1, f_2) , which has corresponding different definitions of h .

Case 1: $h(M) = g^M \in \mathbb{G}$. In this case, $M \in \mathbb{Z}_p$. Under this condition, we can check whether M_1 and M_2 satisfy

$$f_1(M_1, M_2) = g^{aM_1} g^{bM_2} g^c = 1 \tag{8}$$

In other words, we would check whether M_1 and M_2 satisfy

$$aM_1 + bM_2 + c = 0. \tag{9}$$

Case 2: $h(M) = M \in \mathbb{G}$. In this case, $M \in \mathbb{G}$. Under this condition, we would check whether M_1 and M_2 satisfy

$$f_2(M_1, M_2) = M_1^a M_2^b c = 1. \tag{10}$$

For any $c \in \mathbb{G}$, it exists an element $q_c \in \mathbb{G}$ and $d \in \mathbb{Z}_p$ to satisfy $c = (q_c)^d$. Therefore, Eq. (10) would be also written as follow.

$$M_1^a M_2^b q_c^d = 1. \tag{11}$$

For simplicity, we choose Eq. (10) for the following statement.

5.2 Our BEC Instances₁ for f_1

- **KeyGen**(k): It generates $\text{param} = (\mathbb{G}, p, g_1, g_2)$ using the security parameter k for the SPHF_{DDH} system. It randomly choose $(s_1, s_2, t_1, t_2) \xleftarrow{\$} \mathbb{Z}_p^4$ to generates a pair of the public/secret keys (pk_{FS}, sk_{FS}) of the front server and a pair of the public/secret key (pk_{BS}, sk_{BS}) of the back server:

$$\begin{aligned} (pk_{FS}, sk_{FS}) &: (u_1 = g_1^{s_1} g_2^{s_2}, (s_1, s_2)), \\ (pk_{BS}, sk_{BS}) &: (u_2 = g_1^{t_1} g_2^{t_2}, (t_1, t_2)). \end{aligned}$$

We choose $h(M) = g^M$ defined on $\mathbb{Z}_p \rightarrow \mathbb{G}_p$.

- **Enc**(pk_{FS}, pk_{BS}, M): It randomly chooses $r \in \mathbb{Z}_p$ and outputs a ciphertext C of the message $M \in \mathbb{Z}_p$:

$$c = (g_1^r, g_2^r, u_1^r u_2^r g^M).$$

- ★ We randomly choose $(r_1, r_2) \in \mathbb{Z}_p^2$ and compute two ciphertexts C_1 and C_2 , respectively.

$$\begin{aligned} C_1 &= (C_1^{(1)}, C_1^{(2)}, C_1^{(3)}) = (g_1^{r_1}, g_2^{r_1}, u_1^{r_1} u_2^{r_1} g^{M_1}), \\ C_2 &= (C_2^{(1)}, C_2^{(2)}, C_2^{(3)}) = (g_1^{r_2}, g_2^{r_2}, u_1^{r_2} u_2^{r_2} g^{M_2}). \end{aligned}$$

- **FSTest**(sk_{FS}, C_1, C_2, f_1): For testing if $aM_1 + bM_2 + c = 0$, it randomly picks $\Delta w \in \mathbb{Z}_p$ and computes:

$$\begin{aligned} W &= \left((C_1^{(1)})^a (C_2^{(1)})^b, (C_1^{(2)})^a (C_2^{(2)})^b \right), \\ &= (g_1^{ar_1} g_1^{br_2}, g_2^{ar_1} g_2^{br_2}), \\ C &= (C_1^{(3)})^a (C_2^{(3)})^b g_1^c (g_1^{ar_1} g_1^{br_2})^{-s_1} (g_2^{ar_1} g_2^{br_2})^{-s_2}, \\ &= (u_1^{r_1} u_2^{r_1} g^{M_1})^a (u_1^{r_2} u_2^{r_2} g^{M_2})^b g_1^c (g_1^{s_1} g_2^{s_2})^{-ar_1} (g_1^{s_1} g_2^{s_2})^{-br_2}, \\ &= (u_1^{r_1} u_2^{r_1} g^{M_1})^a (u_1^{r_2} u_2^{r_2} g^{M_2})^b g_1^c u_1^{-ar_1} u_2^{-br_2}, \\ &= u_2^{ar_1} u_2^{br_2} g^{aM_1 + bM_2 + c}, \\ &= (g_1^{t_1} g_2^{t_2})^{ar_1} (g_1^{t_1} g_2^{t_2})^{br_2} g^{aM_1 + bM_2 + c}, \\ &= (g_1^{ar_1} g_1^{br_2})^{t_1} (g_2^{ar_1} g_2^{br_2})^{t_2} g^{aM_1 + bM_2 + c}, \\ W' &= (g_1^{ar_1 \Delta w} g_1^{br_2 \Delta w}, g_2^{ar_1 \Delta w} g_2^{br_2 \Delta w}), \\ C' &= (g_1^{ar_1 \Delta w} g_1^{br_2 \Delta w})^{t_1} (g_2^{ar_1 \Delta w} g_2^{br_2 \Delta w})^{t_2} g^{aM_1 + bM_2 + c}. \end{aligned}$$

Finally, it sends $IV = (W', C')$ to the back sever.

- **BSTest**(sk_{BS}, IV): For $IV = (W', C') = (((W')^{(1)}, (W')^{(2)}), C')$, it checks whether

$$C' = ((W')^{(1)})^{t_1} ((W')^{(2)})^{t_2}. \quad (12)$$

If Eq.(12) holds, it outputs 1 indicating that M_1 and M_2 satisfy f_1 , otherwise 0.

5.3 Our BEC Instances₂ for f_2

- **KeyGen(k)**: This algorithm is the same as **KeyGen(k)** algorithm in the BEC Instance₁. We choose $h(M) = M$ defined on $\mathbb{G}_p \rightarrow \mathbb{G}_p$.
- **Enc(pk_{FS}, pk_{BS}, M)**: It randomly chooses $r \in \mathbb{Z}_p$ and outputs a ciphertext C of the message $M \in \mathbb{G}_p$:

$$c = (g_1^r, g_2^r, u_1^r u_2^r M).$$

- * We randomly choose $(r_1, r_2) \in \mathbb{Z}_p^2$ and compute two ciphertexts C_1 and C_2 as follows:

$$\begin{aligned} C_1 &= (C_1^{(1)}, C_1^{(2)}, C_1^{(3)}) = (g_1^{r_1}, g_2^{r_1}, u_1^{r_1} u_2^{r_1} M_1), \\ C_2 &= (C_2^{(1)}, C_2^{(2)}, C_2^{(3)}) = (g_1^{r_2}, g_2^{r_2}, u_1^{r_2} u_2^{r_2} M_2). \end{aligned}$$

- **FSTest(sk_{FS}, C_1, C_2, f)**: For testing if $M_1^a M_2^b c = 1$, it randomly picks $\Delta w \in \mathbb{Z}_p$ and computes:

$$\begin{aligned} W &= (g_1^{ar_1} g_1^{br_2}, g_2^{ar_1} g_2^{br_2}), \\ C &= (g_1^{ar_1} g_1^{br_2})^{t_1} (g_2^{ar_1} g_2^{br_2})^{t_2} M_1^a M_2^b c, \\ W' &= (g_1^{ar_1 \Delta w} g_1^{br_2 \Delta w}, g_2^{ar_1 \Delta w} g_2^{br_2 \Delta w}), \\ C' &= (g_1^{ar_1 \Delta w} g_1^{br_2 \Delta w})^{t_1} (g_2^{ar_1 \Delta w} g_2^{br_2 \Delta w})^{t_2} M_1^a M_2^b c. \end{aligned}$$

Finally, it sends $IV = (W', C')$ to the back sever.

- **BSTest(sk_{BS}, IV)**: For $IV = (W', C') = (((W')^{(1)}, (W')^{(2)}), C')$, it check whether

$$C' = ((W')^{(1)})^{t_1} ((W')^{(2)})^{t_2}. \tag{13}$$

If Eq.(13) holds, it outputs 1 indicating that M_1 and M_2 satisfy f_2 , otherwise 0.

6 Conclusion

In this paper, we extended the functionality of the cryptosystem provided in [8] to support the complex queries like testing whether $aM_1 + bM_2 + c = 0$ or $M_1^a M_2^b c = 1$, which would have many interesting applications. Adopting the two-server setting, our general construction could overcome the offline message guessing attack just like in the single-server framework of searchable encryption. An efficient instantiation of smooth projective hash function from DDH assumption led to our practical scheme without any pairing.

References

1. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_30
2. 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. *J. Cryptol.* **21**(3), 350–391 (2008)
3. Jeong, I.R., Kwon, J.O., Hong, D., Lee, D.H.: Constructing PEKS schemes secure against keyword guessing attacks is possible? *Comput. Commun.* **32**(2), 394–396 (2009)
4. Byun, J.W., Rhee, H.S., Park, H.-A., Lee, D.H.: Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In: Jonker, W., Petković, M. (eds.) SDM 2006. LNCS, vol. 4165, pp. 75–83. Springer, Heidelberg (2006). https://doi.org/10.1007/11844662_6
5. Yau, W.-C., Heng, S.-H., Goi, B.-M.: Off-line keyword guessing attacks on recent public key encryption with keyword search schemes. In: Rong, C., Jaatun, M.G., Sandnes, F.E., Yang, L.T., Ma, J. (eds.) ATC 2008. LNCS, vol. 5060, pp. 100–105. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69295-9_10
6. Fang, L., Susilo, W., Ge, C., Wang, J.: Public key encryption with keyword search secure against keyword guessing attack without random oracle. *Inf. Sci.* **238**, 221–241 (2013)
7. Xu, P., Jin, H., Wu, Q., Wang, W.: Public-key encryption with fuzzy keyword search: a provably secure scheme under keyword guessing attack. *IEEE Trans. Comput.* **62**, 2266–2277 (2013)
8. Chen, R., Mu, Y., Yang, G., Guo, F., Wang, X.: A new general framework for secure public key encryption with keyword search. In: Foo, E., Stebila, D. (eds.) ACISP 2015. LNCS, vol. 9144, pp. 59–76. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19962-7_4
9. Yang, G., Tan, C.H., Huang, Q., Wong, D.S.: Probabilistic public key encryption with equality test. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 119–131. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11925-5_9
10. Chen, R., Yi, M., Yang, G., Guo, F., Wang, X.: Dual-server public-key encryption with keyword search for secure cloud storage. *IEEE Trans. Inf. Forensics Secur.* **11**(4), 789–798 (2016)
11. Tang, Q.: Public key encryption schemes supporting equality test with authorization of different granularity. *Int. J. Appl. Crypt.* **2**(4), 304–321 (2012)
12. 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). https://doi.org/10.1007/3-540-46035-7_4



Efficient Software Implementation of Modular Multiplication in Prime Fields on TI's DSP TMS320C6678

Eito Miyamoto^(✉), Takeshi Sugawara^(✉), and Kazuo Sakiyama^(✉)

The University of Electro-Communications, 1-5-1 Chofugaoka,
Chofu, Tokyo 182-8585, Japan
{e.miyamoto,sugawara,sakiyama}@uec.ac.jp

Abstract. Fast modular multiplication on the state-of-the-art digital signal processor (DSP) is studied in this work. More specifically, Montgomery multiplication over a prime field for an arbitrary 256-bit p is implemented on TMS320C6678 DSP by Texas Instruments. Two implementations optimized for latency and throughput are designed. The implementations are based on the k -bit divided Montgomery modular multiplication algorithm by Kornerup. The algorithm is extended to run two independent Montgomery multiplication in parallel thereby running efficiently on the target DSP by exploiting its symmetric functional units. The proposed implementations are advantageous than the previous implementation proposed by Itoh et al. in terms of latency and throughput. The latency of 0.496 $[\mu\text{s}]$ of the proposed implementation is only 17% of 2.86 $[\mu\text{s}]$ for the implementation proposed by Itoh et al. Moreover, the throughput 4.03×10^6 [Montgomery multiplication(MM)/s] in the present case is more than $\times 10$ the value of 0.37×10^6 [MM/s] from the previous work.

Keywords: Modular multiplication · Montgomery multiplication
Software implementation · DSP · TMS320C6678

1 Introduction

At present, cryptography has gained extensive usage owing to the Internet. In particular, public-key cryptography enabled the creation of indispensable building blocks such as the Internet key exchange (IKE) and digital signatures that the present-day internet security relies on. The expansion of the application of cryptography continues even today. As an increasing number of embedded devices are gaining connectivity to the Internet, there is a demand for more efficient cryptographic implementations in less-expensive devices. Moreover, new cryptographic primitives, such as pairing [1], is being introduced to such embedded devices.

Modular multiplication over a prime field is an essential building block for many of the public-key cryptography involving RSA, elliptic curve cryptography

This work was supported by SECOM Science and Technology Foundation.

(ECC), and even pairing. Therefore, efficient implementation of modular multiplication is the key to efficient public-key cryptography. Consequently, considerable effort has been expended to realize faster modular multiplication on various platforms such as Intel CPU, application specific integrated circuit (ASIC), and field-programmable gate array (FPGA). In addition to these relatively computationally rich environments, embedded microcontrollers and special-purpose processors such as Atmel ATmega128 [2, 3], graphics processing unit (GPU) [4], and Texas Instruments (TI) MSP430 [5] are being considered with the view of improving the performance of public-key cryptography on these platforms.

The digital signal processor (DSP) is a special-purpose processor commonly used for embedded systems. It was originally designed for digital signal processing and is capable of fast arithmetic operations used for common applications such as digital filters and matrix operations. Itoh et al. exploited the fast arithmetic operation for implementing public-key cryptography [6]. Although this seems a promising approach, there is no report of public-key cryptography on DSP since the work by Itoh et al. in 1999.

Since 1999, the performance of DSPs have been significantly improved. Today, the state-of-the-art DSPs run at speeds of more than 1 GHz and achieves 160 GFLOPS [7]. Moreover, DSPs are now being integrated in heterogeneous multicore processors. For example, TI AM572x Sitara SoC [8] has embedded DSPs that is accessible from the main ARM Cortex-A15 processor. This enables an operating system running on the main processor to use DSP as an accelerator. Thus, the DPS now appears a very suitable candidate accelerator for public-key cryptography.

In this paper, fast modular multiplication on the state-of-the-art DSP is studied. More specifically, we design software for Montgomery multiplication over a prime field \mathbb{F}_p for an arbitrary 256-bit p on TI's DSP TMS320C6678 [7]. Two different implementations with different optimizations are examined.

1. The first implementation is optimized for latency. This implementation is based on the k -bit divided Montgomery modular multiplication (k MM) algorithm by Kornerup [9]. In order to reduce latency, the implementation runs exclusively on registers (i.e. no memory is used for data store).
2. The second implementation is optimized for throughput. For the purpose, an extended algorithm called the parallel k -bit divided Montgomery modular multiplication (Parallel k MM) is proposed. The algorithm runs two independent Montgomery multiplications in parallel. The algorithm improves the utilization ratio of functional units in DSP core by exploiting its symmetry.

Note that an optimization based on a special p is not used, and thus, the above implementations can be used for a wide range of applications such as ECC with a non-standard curve and pairing.

As a result, the first implementation achieved a latency of 0.496 [μ s]. This value is only 17% of 2.86 [μ s], the latency achieved in the previous work by Itoh et al. In the second implementation, a throughput of 4.03×10^6 [Montgomery multiplication (MM)/s] was achieved. This value is more than $\times 10$ the throughput of 0.37×10^6 [MM/s] realized by Itoh et al. Our implementations yield better

performance than those of Itoh et al. even when clock frequencies of DSPs are normalized. The latency of the first implementation is 412 [cycles], which is 77% of the 536 [cycles] in the work by Itoh et al. The throughput of the second design is 4.03×10^{-3} [MM/cycle], which is more than twice faster than the design of Itoh et al. with 1.87×10^{-3} [MM/cycle].

The rest of the paper is organized as follows. Section 2 describes the details of TI's DSP. Section 3 reviews literature on modular multiplication and related works. The proposed method is described in Sect. 4. Section 5 discusses the performance evaluation and comparison. Finally, we conclude the paper in Sect. 6.

2 TI C6678 DSP

2.1 Digital Signal Processor

DSP was originally designed for digital signal processing. Therefore, it is equipped with specific instructions such as product-sum operation, matrix processing, and dot product. TI released the C5000 and C6000 series DSPs. The low-end C5000 series are typically used for applications such as audio and wearable devices. Meanwhile the high-performance C6000 series are used for computationally heavy applications such as machine vision.

An emerging trend of DSP is its integration to heterogeneous multicore processors. For example, TI AM572x Sitara SoC [8] integrates ARM Cortex-A15 microprocessor with up to two C6000 series DSPs. Moreover, DSPs are accessible by Linux running on the Cortex-A15 microprocessor. Such an integration opens up the possibility to use DSP as an accelerator for cryptography.

In this work, we use C6678 [7] as the target platform. C6678 is a high-end device of the C6000 series. C6678 has eight cores and reaches 160 GFLOPS at 1.25 GHz.

2.2 Functional Units and Register Files

The architecture of C6678 DSP is shown in Fig. 1 in detail. The DSP core has two sides, namely, A and B sides. Each side has four heterogeneous functional units, namely, .D, .L, .M, and .S. The four functional units run different instructions.

- The .D units operate data load and store, and simple addition and subtraction.
- The .L units operate logic and arithmetic operation, and multi-precision addition.
- The .M units mainly operate multiplication up to 32×32 bits.
- The .S units operate data shift, branch and multi-precision addition.

There are eight functional units in all and they can work simultaneously. Therefore, eight instructions for all the units are issued in each cycle [7].

Both sides have register files containing 32 registers namely A0, ..., A31 and B0, ..., B31, respectively. Each register has a 32-bit data width. Generally,

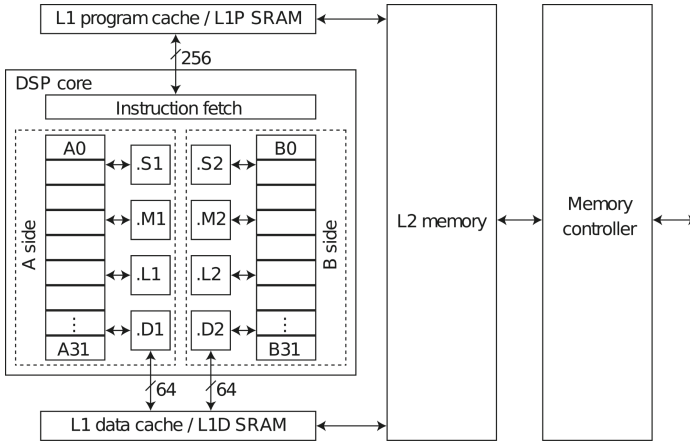


Fig. 1. Architecture of the DSP core and its peripherals

the functional units in the A side (i.e. .D1, .L1, .M1, and .S1) use A0, . . . , A31. Meanwhile .D2, .L2, .M2, and .S2 in the B side use B0, . . . , B31. Further, a special path called the cross path that bridges the two sides is provided. However, the width of the cross path is limited to 32 bits.

2.3 Memory Hierarchy

C6678 has 32KB L1 program and data caches. They are referred to as L1P and L1D caches. The L1 caches are connected to an external DRAM via an L2 cache and a memory controller [10].

The L1 caches can also be configured as random access memories called L1P SRAM and L1D SRAM. L1P SRAM can be used as a fast program store. By loading a program on the L1P SRAM, the performance of the program can be improved by reducing the cache miss penalty. Similarly, L1D SRAM can be used as a fast data store or scratch-pad memory.

3 Previous Work

Modular multiplication is one of the most important operations in public-key cryptography, including RSA and ECC. In the recent years, ECC has been considered as an alternative to RSA because it can be implemented with a shorter key length than RSA for the same security level. ECC is employed in elliptic curve digital signature algorithm (ECDSA), elliptic curve Diffie-Hellman (ECDH) key exchange, and so on. Note that such ECC-based cryptosystems are also based on modular multiplication. Thus, realizing a fast modular multiplier at a low cost is of great interest to many security researchers and engineer [9, 11–15].

Algorithm 1. Montgomery Modular Multiplication [11]

Require: multiplicand $A \geq 0$, multiplier $B \geq 0$, constant R , modulo M , $R > M$,
 $(-MM') \bmod R = 1$.

Ensure: $S = ABR^{-1} \bmod M$, $0 \leq S < M$.

1: $q := (AB \bmod R)M' \bmod R$;

2: $S := (AB + qM)/R$;

3: **if** $S \geq M$ **then**

4: $S := S - M$;

5: **end if**

6: **Return** S ;

3.1 Modular Multiplication

Modular Multiplication deals with a calculation of AB modulo M as

$$S = AB \bmod M,$$

where A and B are non-negative integers, e.g., 128-bit or 256-bit values, and M is a prime number¹. A straightforward algorithm is dividing AB by M to calculate $S = \lceil AB/M \rceil \times M$. However, it takes a long time to compute S in general owing to the slow operation of a divider.

3.2 Montgomery Modular Multiplication

Montgomery Modular Multiplication (MMM) proposed by Montgomery in 1985 [11] is one of the fast modular multiplication algorithms. The algorithm is shown in Algorithm 1. This epoch-making algorithm introduces a constant R such that $R > M$ in order to eliminate the division operations by M . R is normally chosen as a power of 2. Note that this algorithm computes not $S = AB \bmod M$, but $S = ABR^{-1} \bmod M$.

3.3 k -bit Divided Montgomery Modular Multiplication

In 1993, Kornerup proposed the k -bit Divided Montgomery Modular Multiplication (k MM) that is suitable for k -bit CPU, as shown in Algorithm 2 [9]. In the k MM algorithm, modular reduction is performed after the partial multiplication, $b_i A$, so that the intermediate value of S_i cannot exceed M . In other words, k MM divides AB and performs reduction with M' in one loop. Thanks to the flexibility, this multiplication can be made to suit various platforms by changing the value of k .

¹ Many algorithms can support the case that M is odd.

Algorithm 2. k -bit Divided Montgomery Modular Multiplication [9]

Require: multiplicand $A \geq 0$, multiplier $B \geq 0$, divider width k (bit), block count n , modulo $M > 2$, $\gcd(M, 2) = 1$, $(-MM') \bmod 2^k = 1$, $4M < 2^{kn} = R$, $RR^{-1} \bmod M = 1$, $0 \leq A, B \leq 2M$, $B = \sum_{i=0}^{n-1} (2^k)^i b_i$, $b_i \in \{0, 1, \dots, 2^k - 1\}$.

Ensure: $S_n = ABR^{-1} \bmod M$.

- 1: $S_0 := 0$;
 - 2: **for** $i = 0$ to $n - 1$ **do**
 - 3: $q_i := (((S_i + b_i A) \bmod 2^k) M') \bmod 2^k$;
 - 4: $S_{i+1} := (S_i + q_i M + b_i A) \operatorname{div} 2^k$;
 - 5: **end for**
 - 6: **Return** S_n ;
-

Algorithm 3. Bipartite Modular Multiplication [12]

Require: multiplicand A , multiplier $B = (B_{n_w-1} \dots B_0)_r = B_H r^l + B_L$, modulo $M > 2$, $\gcd(M, 2) = 1$, where $0 \leq A, B < M$, $r = 2^k$, $(-MM') \bmod 2^k = 1$, $0 < l < n_w = \lceil n/w \rceil$, $b \geq 3$, $j = \lceil \log_b M \rceil + 1$, $0 \leq AB < b^j$, $\mu = \lfloor b^{2j}/M \rfloor$.

Ensure: $S = ABr^{-\frac{1}{2}} \bmod M$.

- 1: $S_H := S_L := 0$;
 - 2: **for** $i = n_w - 1$ down to l on S_H side and $i = 0$ to $l - 1$ on S_L side **do**
 - 3: $S_H := S_H r + AB_i$; $S_L := S_L + AB_i$;
 - 4: $q := \{(S_H/b^{j-1})\mu/b^{j+1}\}$; $\hat{q} := (S_L \bmod r)M' \bmod r$
 - 5: $S_H := S_H - q\mu$; $S_L := (S_L + \hat{q}M)/r$;
 - 6: **end for**
 - 7: **if** $S_H \geq M$ on S_H side and $S_L \geq M$ on S_L side **then**
 - 8: $S_H := S_H - M$; $S_L := S_L - M$;
 - 9: **end if**
 - 10: **Return** $S = S_H + S_L$;
-

3.4 Bipartite Modular Multiplication

Bipartite Modular Multiplication (BMM) is an algorithm that calculates one modular multiplication using a parallelized computational unit proposed by Kaihara and Takagi in 2005 (see Algorithm 3). This algorithm performs two reduction algorithms in parallel—Montgomery reduction from the least significant bit and Barrett reduction from the most significant bit. The drawback of the algorithm is that the latency of Barrett reduction is normally acts as a bottleneck, limiting the speed performance of BMM. In [15], Knežević et al. proposed a fast BMM method by using special modulo.

3.5 Implementation on TI's DSP C6201

In [6], Itoh et al. proposed an improved Montgomery algorithm for TMS320C6201 in 1999. This implementation is based on the algorithm proposed by Koç et al. [16], and it divides a part of the main process in order to increase the concurrency. This work focused on implementing public-key cryptography on TI's DSP. Therefore, it can serve as a reference comparison with

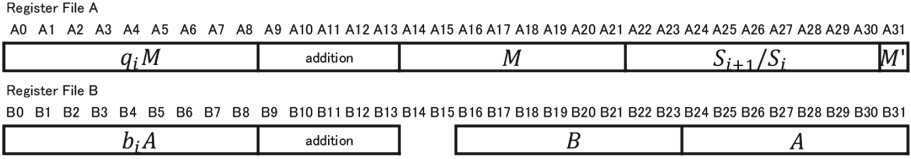


Fig. 2. The register file layout of kMM

the proposed method detailed in the next section. In particular, it is of great interest to see that the performance improvements arise from the improvement in clock frequency and/or in DSP architecture.

4 Proposed Method

In this section, fast software implementation of modular multiplication on C6678 is described. In particular, we designed two implementations optimized for latency and throughput.

The word size of the modular multiplication is determined to 256 bits considering the NIST recommended curve P-256 [17].

4.1 Optimization for Latency

Latency is the interval between the beginning of the operation and the time when the result is ready. kMM and BMM can be a candidate for low-latency implementation. However, kMM is more advantageous, as explained below. The half of BMM that executes the Barrett reduction generates more intermediate values compared to the Montgomery multiplication, especially in the computation for S_H (see Algorithm 3). Such intermediate values should be temporarily stored in an external memory in order to obtain sufficient working registers. The overhead for the memory transfer makes the Barrett reduction inefficient, causing an imbalance between the performances for the Montgomery multiplication and Barrett reduction. Therefore, the BMM is inefficient as compared to the simple kMM .

In order to reduce latency, the proposed implementation makes no memory access. All the input data and intermediate values are allocated in the registers. Figure 2 shows the layout in the register files. The figure represents the $64 \times 32 = 2048$ bits of data stored in $A0, \dots, A31$, and $B0, \dots, B31$. Symbols shown on the boxes correspond to the ones in Algorithm 2. The boxes labelled with “addition” are for temporary variables. The layout is designed to minimise the stall due to the limited bandwidth of the cross-path data transfer.

4.2 Optimization for Throughput

Throughput is the number of operations (e.g., modular multiplications) executed in a normalized time interval. Increasing the utilization ratio in the functional

units is the key to improve throughput. For the purpose, a modified algorithm called the parallel k -bit-divided Montgomery Modular Multiplication (Parallel k MM) is proposed. The algorithm is shown in Algorithm 4. In the algorithm, two independent modular multiplications, namely, $ABR^{-1} \bmod M$ and $XYR^{-1} \bmod M$ are executed in parallel.

The idea behind the Parallel k MM is to run the almost identical programs in the A and B sides by exploiting their symmetry. More specifically, $ABR^{-1} \bmod M$ and $XYR^{-1} \bmod M$ are assigned to the A and B sides, respectively. The technique efficiently reduces the need for cross-path data transfer and simplifies the scheduling of the functional units at the same time. The proposed algorithm is inspired by BMM wherein two independent calculations are executed in parallel. However, the proposed Parallel k MM is more efficient on C6678 as BMM.

A drawback of running two independent Montgomery multiplication simultaneously is that access to an external memory is necessary. This is because the number of intermediate or temporary variables are roughly doubled, and thus, they do not fit in the register files. To address the problem, the k MM algorithm in Algorithm 2 is split into two parts. In Algorithm 4, the first and second parts correspond to the lines 2–5 and 6–10, respectively. In the first part, all the partial products $C_i = Ab_i$ and $W_i = Xy_i$ for $i \in [0, \dots, n-1]$ are calculated and stored to L1D SRAM in advance. Then, the stored C_i and W_i are loaded and used in the second part. There are several benefits of splitting k MM into two parts. First, the number of temporary variables (except C_i and W_i) that should be stored in the external memory is reduced. This is because the number of intermediate values in each part is reduced thanks to the splitting. Second, the pre-computation of C_i and W_i relaxes the data dependency in the second part, enabling more efficient scheduling of the functional units. Finally, the penalty of accessing the external memory can be reduced because the pre-computed values can be loaded to a register via `.D` at an arbitrary timing. Therefore, by overwrapping the latency for memory loading with that of other operations such as multiplication, the overhead of memory access can be reduced effectively.

4.3 Common Optimization Techniques

Common optimization techniques used for both the latency- and throughput-optimized implementations are described here.

Instructions: As shown in Algorithms 2 and 4, the operations needed for Montgomery multiplication are integer multiplication and addition. First, integer multiplication is implemented with 32-bit unsigned multiplication instruction `MPY32U` that can be run on the `.M` functional units. Meanwhile the `.L` and `.S` functional units serve for integer addition. The `.D` unit is used for data transfer between registers and memory, if necessary.

L1P and L1D SRAM: As discussed in Sect. 2.3, C6678 has 32 KB L1 program and data caches that can be configured to SRAM. In our design, both of them are configured as SRAM. More specifically, the whole program is placed and fetched

Algorithm 4. Proposed Parallel k -bit Divided Montgomery Modular Multiplication

Require: multiplicand $A, X > 0$, multiplier $B, Y > 0$, divider width k (bit), block count n , modulo $M > 2$, $\gcd(M, 2) = 1$, $(-MM') \bmod 2^k = 1$, $4M < 2^{kn} = R$, $RR^{-1} \bmod M = 1$, $0 \leq A, B, X, Y \leq 2M$,
 $B = \sum_{i=0}^{n-1} (2^k)^i b_i$, $b_i \in \{0, 1, \dots, 2^k - 1\}$, $Y = \sum_{i=0}^{n-1} (2^k)^i y_i$,
 $y_i \in \{0, 1, \dots, 2^k - 1\}$.

Ensure: $S = ABR^{-1} \bmod M$ and $T = XYR^{-1} \bmod M$, $0 \leq S, T \leq M$.

```

1:  $S := T := 0$ ;
2: for  $i = 0$  to  $n - 1$  do
3:    $C_i := Ab_i$ ;
4:    $W_i := Xy_i$ ;
5: end for
6: for  $i = 0$  to  $n - 1$  do
7:    $S := S + C_i$ ;                                 $T := T + W_i$ ;
8:    $q := ((S \bmod 2^k)M') \bmod 2^k$ ;               $\hat{q} := ((T \bmod 2^k)M') \bmod 2^k$ ;
9:    $S := (S + qM) \text{div } 2^k$ ;                     $T := (T + \hat{q}M) \text{div } 2^k$ ;
10: end for
11: Return  $S$  and  $T$ ;

```

from L1P SRAM. In addition, the temporary variable and pre-computed values that should be moved from registers to an external memory are stored in L1D SRAM. By doing so, the performance penalty resulting from cache miss can be avoided.

Loop Unrolling: k MM and Parallel k MM are loop algorithm. In software implementation, loop architecture requires many clock cycles as it needs to issue the jump instruction. Loop unrolling is effective to avoid this. After unrolling, some unnecessary calculations can be removed. For instance, addition with zero caused when $i = 0$ (S and T is initialized 0 before loop) can be simply removed.

5 Evaluation

5.1 Unit Utilization Ratio and Code Size

Figures 3 and 4 show the cycle-precision utilization of each functional unit in k MM loop and Parallel k MM loop, respectively. The label i on the vertical axis represents the index of the loop. The horizontal axis represents the clock cycle. For instance, in Fig. 3, loop $i = 0$ takes 38 cycles. The utilization ratio is summarized in Table 1. Parallel k MM has a high utilization ratio which is twice that of k MM. The result is explained as follows. As described in Sect. 4.1, the register layout is designed so that the stall caused by the cross-path data transfer is reduced. However, there remain other cross-path data transfers that act as bottlenecks, resulting in a low utilization ratio. Storing data in an external memory, thereby reducing the cross pass, is a good strategy for improving the utilization ratio in C6678. The code size of k MM and Parallel k MM are 3.5 KB and 7.8 KB, respectively. Both implementations fit in the 32 KB L1P SRAM.

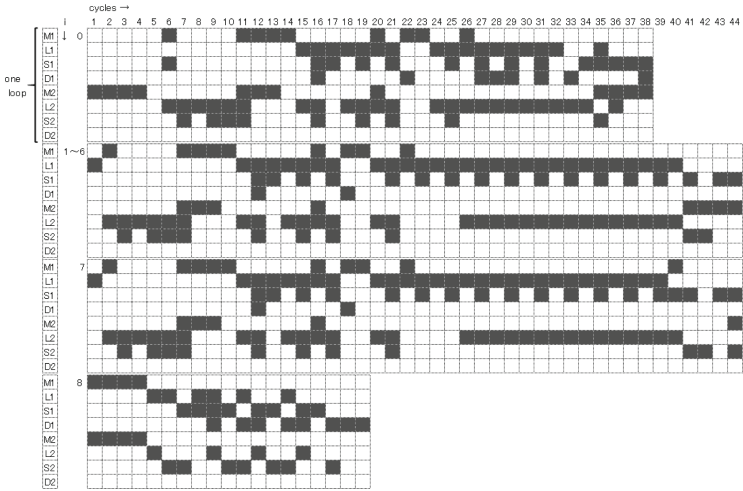


Fig. 3. Unit utilization of kMM on C6678

Table 1. Unit utilization ratio (%)

Algorithm	.M1	.L1	.S1	.D1	.M2	.L2	.S2	.D2	Average
kMM	49.0	61.6	38.9	7.4	39.2	63.3	23.8	0.0	35.4
Parallel kMM	35.6	97.8	86.7	80.4	35.6	97.8	86.7	62.2	72.8

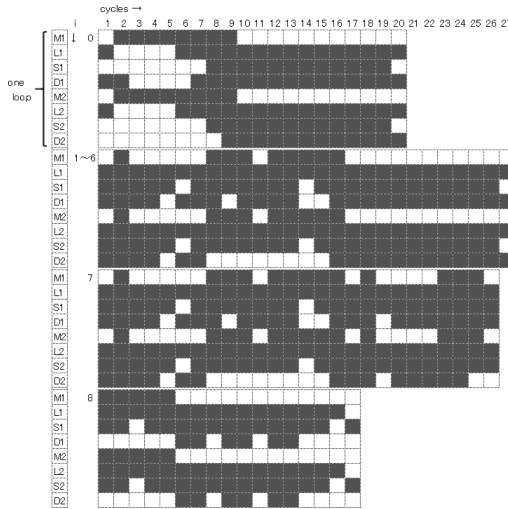


Fig. 4. Unit utilization of parallel kMM on C6678

5.2 Comparison with Previous Works

Table 2 compares the performance among software and hardware implementations of 256-bit modular multiplication. The result of Parallel k MM is the half time of whole operation because it can compute two independent 256-bit MMM.

Parallel k MM needs less clock cycles than other software implementations [6, 18, 19]. Particularly, in comparison with the implementation on TI's DSP TMS320 C6201, Parallel k MM is almost 10 times faster. The difference between hardware implementation [13] and Parallel k MM is 0.078 μ s. However, the hardware implementation employs 34 16×16 multipliers, while Parallel k MM uses 2 32×32 multipliers.

Table 2. Performance comparison of modular multiplication

Reference	Description	Platform	Freq. (MHz)	Clock cycles/one MM	Exec time (μ s/one MM)	Throughput (one MM/s) (one MM/clock cycle)	Latency (μ s)
This work (Parallel k MM)	Software implementation	TI C6678	1000	248	0.248	4.03×10^6 4.03×10^{-3}	0.496
This work (k MM)	Software implementation	TI C6678	1000	412	0.412	2.43×10^6 2.43×10^{-3}	0.412
Itoh <i>et al.</i> [6]	Software implementation	TI C6201	200	536	2.68	0.37×10^6 1.87×10^{-3}	2.86
Tenca & Koç [18]	Software implementation	ARM processor	80	344	43	0.02×10^6 2.90×10^{-3}	43
Brown <i>et al.</i> [19]	Software implementation	Pentium II	400	628	1.57	0.64×10^6 1.59×10^{-3}	1.57
Fan <i>et al.</i> [14]	4-cores 4 32×32 mults	Xilinx XC2VP30	81	-	1.5	-	1.5
Mentens <i>et al.</i> [13]	34 16×16 mults	Xilinx XC2VP30	125	-	0.17	-	0.17

6 Conclusions

In this paper, we proposed two different implementations of Montgomery modular multiplication suitable for TI's DSP TMS320C6678. The first design is optimized for latency and works exclusively on register files. The second design is

optimized for throughput. We proposed an extension of k -bit divided Montgomery modular multiplication in which two independent Montgomery multiplications run in parallel. The algorithm runs efficiently on C6678 exploiting its symmetric functional units in the DSP core.

The proposed designs perform better than the conventional design proposed by Itoh et al. in terms of latency and throughput. Although the platform (C6678) has $\times 5$ faster clock frequency than that of Itoh et al., our implementations are still advantageous even after the performances are normalized by clock frequencies. Our implementation achieved a latency of 412 [cycles], which is 77% of the 536 [cycles] in the work of Itoh et al. The throughput of our implementation is 4.03×10^{-3} [MM/cycle], which is more than twice faster than the 1.87×10^{-3} [MM/cycle] in the work of Itoh et al.

Our future work includes a more rigorous evaluation. For the purpose, we plan to compare the proposed methods with the conventional one on the same platform, and to conduct in-depth algorithm analysis for data dependency.

References

1. Costello, C.: Pairings for beginners. <http://www.craigcostello.com.au/pairings/PairingsForBeginners.pdf>
2. Gura, N., Patel, A., Wander, A., Eberle, H., Shantz, S.C.: Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 119–132. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_9
3. Kargl, A., Pyka, S., Seuschek, H.: Fast arithmetic on ATmega128 for elliptic curve cryptography. Cryptology ePrint Archive, Report 2008/442 (2008). <http://eprint.iacr.org/2008/442>
4. Szerwinski, R., Güneysu, T.: Exploiting the power of GPUs for asymmetric cryptography. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 79–99. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85053-3_6
5. Gouvêa, C.P.L., Oliveira, L.B., López, J.: Efficient software implementation of public-key cryptography on sensor networks using the MSP430x microcontroller. *J. Cryptograph. Eng.* **2**(1), 19–29 (2012)
6. Itoh, K., Takenaka, M., Torii, N., Temma, S., Kurihara, Y.: Fast implementation of public-key cryptography on a DSP TMS320C6201. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 61–72. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48059-5_7
7. Texas Instruments: C66x CPU and instruction set reference guide. <http://www.ti.com/lit/ug/sprugh7/sprugh7.pdf>
8. Texas Instruments: AM572x Sitara processors. <http://www.tij.co.jp/jp/lit/ds/symlink/am5728.pdf>
9. Kornerup, P.: High-radix modular multiplication for cryptosystems. In: Proceedings of the 11th Symposium on Computer Arithmetic 1993, pp. 277–283. IEEE (1993)
10. Texas Instruments: C66x DSP cache user's guide. <http://www.ti.com/lit/ug/sprugy8/sprugy8.pdf>
11. Montgomery, P.L.: Modular multiplication without trial division. *Math. Comput.* **44**(170), 519–521 (1985)

12. Kaihara, M.E., Takagi, N.: Bipartite modular multiplication. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 201–210. Springer, Heidelberg (2005). https://doi.org/10.1007/11545262_15
13. Mentens, N., Sakiyama, K., Preneel, B., Verbauwhede, I.: Efficient pipelining for modular multiplication architectures in prime fields. In: Proceedings of the 17th ACM Great Lakes symposium on VLSI, pp. 534–539. ACM (2007)
14. Fan, J., Sakiyama, K., Verbauwhede, I.: Montgomery modular multiplication algorithm on multi-core systems. In: 2007 IEEE Workshop on Signal Processing Systems, pp. 261–266. IEEE (2007)
15. Knežević, M., Vercauteren, F., Verbauwhede, I.: Speeding up bipartite modular multiplication. In: Hasan, M.A., Helleseht, T. (eds.) WAIFI 2010. LNCS, vol. 6087, pp. 166–179. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13797-6_12
16. Koç, Ç.K., Acar, T., Kaliski, B.S.: Analyzing and comparing Montgomery multiplication algorithms. *IEEE Micro* **16**(3), 26–33 (1996)
17. Recommended elliptic curves for federal government use. <http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf>
18. Tenca, A.F., Koç, Ç.K.: A scalable architecture for modular multiplication based on Montgomery’s algorithm. *IEEE Trans. Comput.* **52**(9), 1215–1221 (2003)
19. Brown, M., Hankerson, D., López, J., Menezes, A.: Software implementation of the NIST elliptic curves over prime fields. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 250–265. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45353-9_19



Key Managements of Underwater Acoustic Communication Environments

Hyunki Kim^(✉), Jaehoon Lee^(✉), and Okyeon Yi^(✉)

Financial Information Security, Kookmin University, 77 Jeongneung-RO,
Seongbuk-GU, Seoul 02707, Korea
{Lp1p456, guderian88, oyyi}@kookmin.ac.kr

Abstract. In the underwater environment, it is difficult to use the RF signal used on the ground due to the characteristics of the medium different from the terrestrial. Therefore, when communicating with each node, acoustic communication is performed instead. Since the acoustic communication environment is poor in various aspects such as transmission speed and bandwidth compared to RF communication, it is difficult to apply the existing security method used in RF communication as it is. In this paper, we propose a key management method between entities considering underwater environment.

Keywords: Key management · UWSN · Underwater acoustic communication

1 Introduction

In the natural environment on the ground, various data are collected by wireless communication and used in various fields such as big data analysis and environmental research. Similarly, in the oceans, a variety of marine data can be collected and used in various fields. The analysis of data collected in the water can be used to explore various marine resources such as oil and gas hydrates and other marine resources such as food. It can be used for a variety of purposes such as water quality management, disaster detection by marine environmental pollution, military defense through submarine intrusion detection. Therefore, as in the case of the ground, the information collected in the underwater environment will be treated as very important, and the importance of underwater environmental security will also be increased accordingly.

The WSN on the ground can be configured using various kinds of RF communication such as LTE, LoRa, Zigbee. In the underwater environment, however, RF communication is not used due to the nature of the medium. Since RF signals generate sharp signal attenuation and signal dispersion, they use acoustic communication in underwater environments. However, the transmission speed of the acoustic communication is lower than that of the RF communication, and the usable frequency band is very poor. Therefore, because underwater acoustic communication has lower performance than ground-based RF communication, various technologies used in underwater environments should be designed considering underwater characteristics.

Standardization of underwater acoustic communications is still in its infancy, and there are few standard documents or recommendations for security that are suitable for

underwater communications environments. Although several encryption schemes and authentication protocols that can be applied to the underwater environment are proposed, they are not as active as the WSN environment on the ground, and there is no way to manage the keys defined between entities in order to apply security.

Therefore, this paper analyzes the whole UWSN infrastructure, proposes a key management method only for the area using the acoustic communication, and compares it with the key management methods that have been proposed previously.

2 Related Research

2.1 Underwater Network Environment Analysis

An underwater communication network is a network in which underwater devices transmitting and receiving signals using a frequency of acoustic wave as a water medium. The structure of conceptual underwater communication is as shown in Fig. 1. From the base station to the gateway called the surface station, data is exchanged by the RF communication, and from the surface station to the sensor node, the underwater acoustic communication is used.

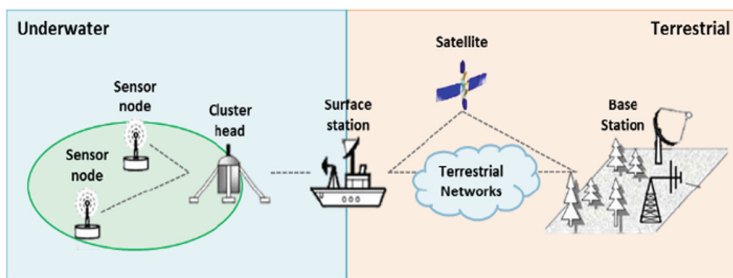


Fig. 1. Environments of underwater communication networks

When performing a acoustic communication from a sensor to a surface station, two kinds of communication can be largely as shown in Fig. 2.

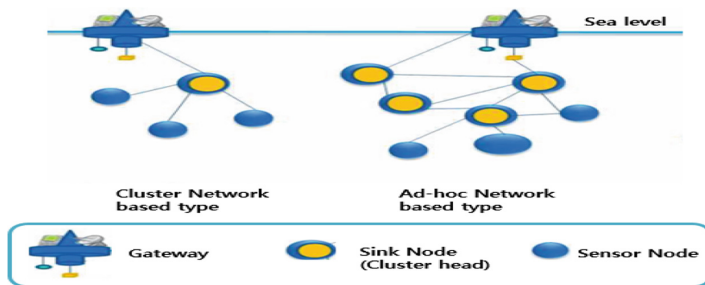


Fig. 2. Underwater acoustic communication network

Cluster Network Based Type. A cluster head and a plurality of sensors form a cluster to communicate. The cluster head merges the data collected by the sensors and transmits them to the gateway.

Ad-hoc Network Based Type. There is a sensor that can communicate at regular intervals and it is a structure to exchange data by multi-hop communication.

2.2 Various Security in UWSN

CCM-UW (Counter with CBC-MAC for Underwater). It is an underwater media access control protocol. This protocol is an encryption method in which the existing CBC-MAC (CCM *) mode is modified to suit the underwater environment. It provides six security levels depending on security strength, energy consumption, transmission time, and so on. The protocol provides confidentiality, integrity, and replay attack protection.

UW-AKE (Lightweight Authentication and Key Establishment Protocol for Underwater Acoustic Sensor Networks). It is an authentication protocol of underwater environment considering the configuration of sensor node, cluster head, surface station, and base station in a network environment formed by cluster head based topology. The main feature is that mutual authentication of all entities is possible by transmitting data combining the authentication data when communicating between nodes by one time.

UW-TAP (Ticket Based Authentication Protocol for Underwater Wireless Sensor Network). Like UW-AKE, it consists of Sensor node, Cluster head, Gateway (Surface station) and Server (Base station). This is an authentication protocol considering the mobility of the sensor node according to the water flow in the underwater environment. Assuming the sensor node is out of the existing cluster and is included in the new cluster, it re-authenticates the sensor node efficiently through the ticket issuing method.

Piecewise Key Management Design Considering Capability of Underwater Communication Nodes. This paper proposes key management methods and requirements for each entity to communicate securely in underwater environment. It classified the underwater nodes according to their capabilities. And it explains a brief summary of what keys should be shared and how they should be distributed.

This paper is written by the author who wrote the current paper. The current paper complements the lack of ‘piecewise key management design’ and shows the advantages of the proposed scheme and the features of each security scheme compared to other security schemes.

3 Key Management Among Underwater Nodes

When the entities of the underwater communication infrastructure are layered, it is divided into the RF communication part and the acoustic communication part based on the entity existing at the sea level as shown in Fig. 3. In case of RF communication

part, from the base station to the surface station is possible, so the existing communication and protocol can be applied to the situation. However, since the existing nodes in the water use the acoustic communication, the existing protocol and security method cannot be applied. Therefore it is the part that needs to be studied continuously.

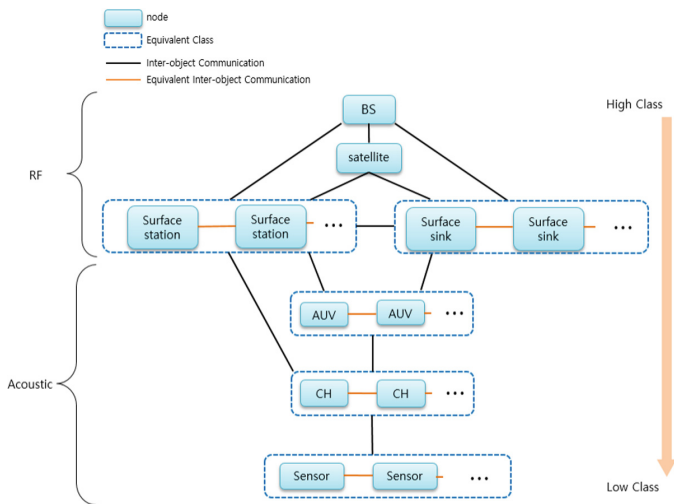


Fig. 3. Underwater network model

Figure 3 shows the hierarchical structure of each underwater communication infrastructure. Each entity has the following characteristics as it goes down from the base station to the sensor.

- The computational capacity is lowered.
- The memory size is reduced.
- The communication capacity is lowered.
- Production cost is reduced.
- The size of communication node become smaller.
- Battery replacement becomes difficult.
- The number of nodes increases.

Nodes that perform underwater acoustic communication can change the surrounding entity members due to currents, terrain, and so on. This may result in re-opening the communication session or re-authentication to establish trust between the new members. Therefore, the power consumption may increase more than the ground situation, and each node has different computation ability and communication ability, so each node should have different security requirements. Therefore, key management is required in consideration of mobility, low computation, and communication capability, which are typical characteristics of an underwater acoustic communication network.

3.1 Analysis the Functions Performed by Each Node and the Required Security Type

Depending on the functions performed by each entity in the underwater communication infrastructure, the security to be applied to the entity also changes. Figure 4 shows the end-entity with which each node communicates, the master/slave relationship in the communication, the function to perform, and the security required for the node.

Base Station. It is the most senior entity in the underwater infrastructure, which controls, manages underwater objects and collects the data. Since it mainly communicates with the surface station on the water surface, authentication and encryption communication with the surface station is required. And to do this, it manages the keys for the Surface station.

Surface Station. It acts as a gateway between the underwater and the terrestrial and manages objects in the water. Therefore, it is necessary to be able to communicate with the underwater nodes, AUV, and cluster head, etc. and authentication and encryption communication between each node is required to establish a reliable communication session. Surface stations can manage the keys of underwater entities according to the configured security policy of the infrastructure.

Autonomous Underwater Vehicle. It acts as an underwater gateway to transmit and receive data collected in the water to/from the surface station and to enable the cluster head to communicate smoothly to the surface station. Since it is necessary to authenticate and encrypt communication with the surface station or to authenticate the cluster head so that the object to be communicated can be trusted, the key corresponding to that part must be managed.

Cluster Head. This merges the data from the sensor and sends it to the surface station or the control data sent from the surface station to the sensor. Because it merges the data and manages multiple sensors within a single underwater cluster, the cluster head must manage key information for sensors added to the cluster by sensors or water currents in its own cluster.

Sensor. This transmits the collected data to the cluster head periodically, and supports the ad-hoc communication so that other nearby sensors can communicate smoothly. Although the capability of node is the least, it must be able to securely transmit data collected in the water to the trusted interval. Therefore, it must share a key with the cluster head.

Much research has been done on RF communications, such as between the base station and the surface station, that are already suitable for RF communications. Therefore, in this paper, we propose a method of managing keys between Gateway (Surface Station), Cluster head, and Sensor, which is a sound communication entity, considering underwater environment.

It is assumed that the key pool is distributed through the off-line immediately before the surface station, the cluster head, and the sensor are initially put into the field.

The initial key pool is assumed to follow the ‘q-composite random key pre-distribution scheme’ [5]. Also, it is assumed that ID, Nonce, MAC_{K_i} are 4-byte and K_i is 16-byte in each process.

Entity	End-entity of Communication	Master/Slave	Functions	Required Security type
Base Station (BS)	• SS	• Master of SS	<ul style="list-style-type: none"> Control and management of underwater and sea level entity Data Collection 	<ul style="list-style-type: none"> SS Authentication SS Key management
Surface Station, Surface sink (SS)	<ul style="list-style-type: none"> BS SS AUV CH 	<ul style="list-style-type: none"> Slave of BS SS Master of AUV Master of CH 	<ul style="list-style-type: none"> Underwater Device Management Sea level Gateway 	<ul style="list-style-type: none"> BS Authentication AUV Authentication CH Authentication Encryption communication with end entities AUV, CH, Sensor Key management
AUV	• SS	• Slave of SS	<ul style="list-style-type: none"> Underwater Gateway 	<ul style="list-style-type: none"> SS Authentication CH Authentication Data encryption
Cluster Head (CH)	<ul style="list-style-type: none"> SS Sensor 	<ul style="list-style-type: none"> Slave of SS Master of Sensor 	<ul style="list-style-type: none"> Sensor management Merging data Relaying 	<ul style="list-style-type: none"> SS Authentication AUV Authentication Sensor Authentication data encryption
Sensor	• CH	• Slave of CH	<ul style="list-style-type: none"> sensing ad-hoc communication with Sensors 	<ul style="list-style-type: none"> CH Authentication Sensor Authentication

Fig. 4. Features of each underwater entity

3.2 Key Establishment Between Equivalent Nodes

The equivalent node among the underwater acoustic communication nodes means between a cluster head and a cluster head, or between a sensor and a sensor. The proposed key establishment method is shown in Fig. 5 and the procedure is as follows.

- The node A that wants to establish the key first broadcasts its ID_A and Nonce N_A around.
- The Node B which receives the data from A checks whether the ID_A received from its key pool exists. If so, it generates MAC_B using the key K_i corresponding to ID_A and sends it to Node A together with ID_B and Nonce N_B .
- Node A verifies MAC_B received from B and authenticates that B is legitimate. It generates MAC_A , which is its authentication information, and transmits it to B. Node A generates a K_{AB} to use with B because it has certified that B is a legitimate opponent.
- Node A and B update K_i in the key pool to newly generated K_{AB} .

3.3 Key Establishment Between Different Nodes

The different node among the underwater acoustic communication nodes means between a cluster head and a Sensor, or between a Gateway and a Cluster head. The proposed key establishment method is shown in Fig. 6 and the procedure is as follows.

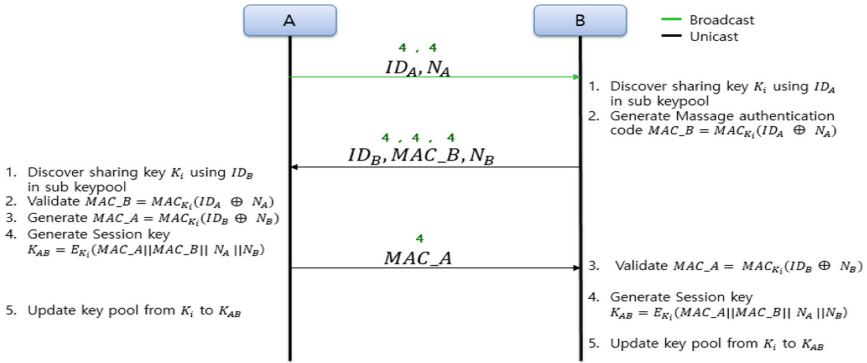


Fig. 5. Key establishment between equivalent nodes

- UA (Upper A) broadcasts its own ID_{UA} to the surrounding nodes in order to inform its existence.
- Node B that has received the corresponding data from the UA checks whether ID_{UA} received from its key pool exists. If so, B generates MAC_B with the key K_i corresponding to ID_{UA} and transmits it to the UA together with its own ID_B .
- Node UA checks if ID_B received from its key pool exists, and if so, decrypts MAC_B using key K_i corresponding to ID_B . After that, it extracts Nonce N_B and authenticates that B is a legitimate partner through $MAC_{K_i}(N_B)$, and generates RES_{UA} which is authentication data of UA itself. Since B is a legitimate node and its computational capability is relatively low, the UA generates and encrypts the session key K_{UAB} and sends it to B along with the RES_{UA} .
- Node B verifies the RES_{UA} received from UA to authenticate that UA is legitimate, and decrypts the MSG_{UAB} with the pre-shared key to obtain a newly generated key.
- Node B updates the key after transmitting a message to the UA indicating that the new key has been Successfully received.

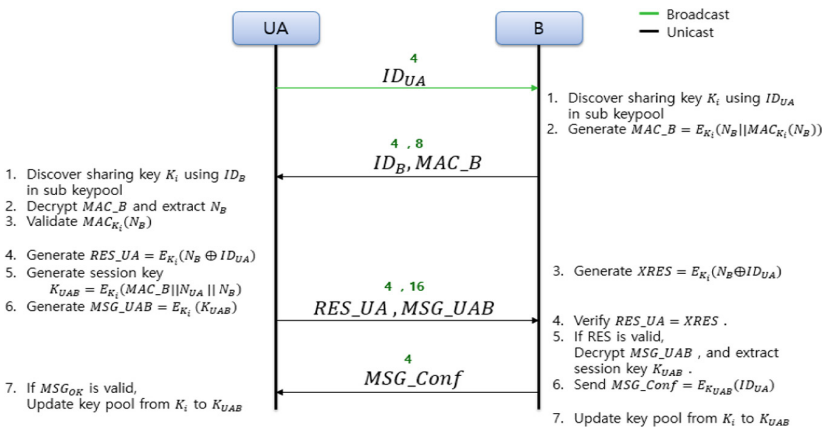


Fig. 6. Key establishment between different nodes

4 Analysis of Key Management Aspects

4.1 Kind of Functions and Number of Usage

Authentication and key management can be performed in various ways depending on the network structure and the situation. Next, we compare UW-AKE and UW-TAP, the key management method, to the proposed method. Both UW-AKE and UW-TAP and the newly proposed key management method are cluster network based method, but the authentication entity and the data to be shared in advance are different. Figure 7 shows the protocol-specific characteristics.

The proposed method reduces the scope of entity authentication and increases the number of communications per node more than the existing UW-AKE or UW-TAP method. However, the number of cryptographic functions used in the protocol process is reduced by about 30% compared to UW-AKE, and the maximum length of the data format to be transmitted is reduced by 25% so that it is possible even in a poor underwater environment. And compared to UW-TAP, the length of nonce and MAC is four times as long, so it has high success rate such as error detection or data loss detection in underwater environment, and cryptographic safety.

	UW-AKE		UW-TAP		Key establishment between different nodes
Network topology	Cluster Network based		Cluster Network based		Cluster Network based
Authentication entity	BS, Surf, CH, Sensor		BS, Surf, CH, Sensor		CH, Sensor, Surf
pre-shared data	pre-shared key		pre-shared key group index table		sub Key pool
number of Cryptographic function usage	Gen. Random number : 4 Enc/Dec : 12 KDF : 2		Gen. Random number : 4 Enc/Dec : 9 MAC/verify : 10 KDF : 2		Gen. Random number : 2 Enc/Dec : 8 MAC: 2
size of transport authentication data	underwater 17 ~ 26	Terrestrial 17 ~ 35	underwater 6 ~ 13	Terrestrial 10 ~ 18	underwater 4 ~ 20
number of communication	BS, Surf, CH, Sensor 1 time each		BS, Surf, CH, Sensor 1 time each		Surf-CH, CH-Sensor 2 time each
size of data field(Byte)	ID: 1 Nonce: 8 MAC: 8 Key: 16		ID: 1 Nonce: 1 MAC: 1 Key: 16		ID:4 Nonce:4 MAC:4, 8 Key:16

Fig. 7. Comparison of UW-AKE, UW-TAP and key establishment between different nodes

5 Conclusion

Underwater acoustic communication technology has recently been under development in advanced countries, and has attracted attention as a technique that is essential for the development of a large amount of underwater resources. Among the underwater communication infrastructure, security and key management system for RF communication between each entity have been extensively studied, but security in the

underwater acoustic communication part is still in its initial state. In this paper, we proposed a key management method that can be applied among the objects communicating through the acoustic wave among the underwater communication infrastructure. It is possible to provide partial key establishment even in a similar structure in which computation capability, communication capability, memory size, etc. are limited and layered, not a method applicable only to a specific object.

Afterwards, it is necessary to define the key management method according to various scenarios such as a situation where an object is newly added to the underwater environment, an object is discarded due to the end of the life cycle, and the like.

Acknowledgments. The work is a part of the results of the research “Development of the wide-band underwater mobile communication systems” supported by Ministry of Oceans and Fisheries, Korea.

References

1. Park, M.: UW-AKE: lightweight authentication and key establishment protocol for underwater acoustic sensor networks (2016)
2. Yun, C.: Ticket-based authentication protocol for underwater wireless sensor network (2016)
3. Ibragimov, M.: CCM-UW security modes for low-band underwater acoustic sensor networks
4. Kim, H., Lee, J., Yi, O.: Proposal of piecewise key management design considering capability of underwater communication nodes (2017)
5. Urunov, K., Namgung, J.-I., Park, S.-H.: Custody transfer of bundle layer in security mechanism for underwater internet of things (UIoT) (2015)
6. Chan, H.: Random key predistribution schemes for sensor networks (2003)
7. TTAK.KO-06.0352.: Underwater Acoustic Communication Network System Overview and Requirements (2013)
8. TTAK.OT-06.0048.: Message Structure for Underwater Acoustic Modem (2014)
9. Raazi, S.M.K.R.: Key management schemes of wireless sensor networks: a survey
10. Fu, J.B., Li, Q.L., Li, S., Ssanyu, L.: A modified Q-composite random key predistribution scheme based on kryptograph (2010)



Parallel Implementations of SIMON and SPECK, Revisited

Taehwan Park¹, Hwajeong Seo², Garam Lee¹, Md. Al-Amin Khandaker³,
Yasuyuki Nogami³, and Howon Kim¹(✉)

¹ Department of Electrical and Computer Engineering, Pusan National University,
Busandaehakro 63beongil 2, Geumjeong-Gu, Busan 46241, Republic of Korea
{pth5804,rkfk218,howonkim}@pusan.ac.kr

² IT Department, Hansung University, 116 Samseong Yoro-16gil Seongbuk-gu,
Seoul 136-792, Republic of Korea
hwajeong@hansung.ac.kr

³ Graduate School of Natural Science and Technology, Okayama University,
3-1-1, Tsushima-naka, Kita, Okayama 7008530, Japan
khandaker@s.okayama-u.ac.jp, yasuyuki.nogami@okayama-u.ac.jp

Abstract. In this paper, we revisited the parallel implementation of SIMON and SPECK block ciphers. The performances of SIMON and SPECK are significantly improved by using ARM NEON SIMD (Single Instruction Multiple Data) parallel computing and OpenMP SIMT (Single Instruction Multiple Thread). We optimized the implementation on ARM NEON architecture. For optimized NEON, we reduced the number of registers for round key and increased the number of registers for plaintexts. Furthermore, we proposed the efficient forward and backward alignment methods. Finally, we maximize the performance by using SIMT (Single Instruction Multiple Threads). In the case of performance of proposed methods and proposed methods with SIMT, SIMON 128/128 encryption within 32.4, 14.3 cycles/byte, SIMON 128/192 encryption within 30.1, 15.9 cycles/byte, SIMON 128/256 encryption within 32.4, 16.9 cycles/byte, SPECK 128/128 encryption within 9.7, 5.1 cycles/byte, SPECK 128/192 encryption within 10.4, 5.6 cycles/byte, SPECK 128/256 encryption within 11.0, and 5.6 cycles/byte respectively on ARM Cortex-A53 environment.

Keywords: SIMON · SPECK · ARM NEON · SIMD · SIMT
OpenMP

This work was supported by the Energy Efficiency & Resources Core Technology Program of the Korea Institute of Energy Technology Evaluation and Planning (KETEP), granted financial resource from the Ministry of Trade, Industry & Energy, Republic of Korea. (No. 20152000000170) and has been supported by JSPS KAKENHI Grant-in-Aid for Scientific Research (A) Number 16H01723. Hwajeong Seo was supported by the ICT R&D program of MSIP/IITP. [B0717-16-0097, Development of V2X Service Integrated Security Technology for Autonomous Driving Vehicle].

1 Introduction

The SIMON and SPECK family block ciphers were announced by the NSA in 2013 [1]. The SIMON and SPECK family block ciphers follow ARX architecture and these are the first family block ciphers which provide various block and key size. Recent research results on SIMON and SPECK block cipher are as follow. Dofe et al. [4] proposed the fault-tolerant method for SIMON block cipher, they suggested 3 fault-tolerant methods for SIMON64/96 block cipher. In Shi et al. [5] proposed the linear (hull) cryptanalysis of SIMON block cipher round-reduced version by using linear characteristic on round encryption function of SIMON. The differential fault analysis on SIMON and SPECK ciphers [6]. There is a research on the side-channel perspective of SIMON block cipher FPGA implementation [7]. There are hardware and software implementation [1]. In the case of software implementation, they implemented SIMON block cipher on 8-bit AVR microprocessor environment [1,2], they proposed memory usage optimization such as RAM-minimizing and speed optimization method such as the loop-unrolled method by using AVR 8-bit assembly language [2]. The other software implementation of SIMON block cipher focuses on 16-bit MSP430 micro-controller, x86 platform (Xeon E5640, Core i7-4770) and 32-bit ARM processor (Samsung Exynos 5 Dual) using C language [3].

Especially, Park et al. [11] proposed the ARM-NEON SIMD implementations of SIMON and SPECK block cipher by using ARM-NEON intrinsic functions. However, Park et al. [11] did not use all of NEON registers and did not consider the SIMT (Single Instruction Multiple Thread).

In this paper, we proposed more efficient parallel implementations of SIMON, SPECK family block ciphers on 64-bit ARM Cortex-A series processor NEON environment (RaspberryPi3 model B based on ARM Cortex-A53) by using NEON SIMD (Single Instruction Multiple Data) and OpenMP for SIMT (Single Instruction Multiple Threads). The remainder of this paper is organized as follows. Section 2 discusses how to describe the SIMON and SPECK family block cipher and the previous literature related to the implementations of the cipher on 32-bit processor environment by using ARM-NEON processor. We suggest efficient optimized parallel implementations of SIMON and SPECK family block cipher on 64-bit ARM Cortex-A53 processor by using ARM NEON and OpenMP in Sect. 3. Section 4 gives insights on how to present numerical results or applications that illustrate the results provided in the body of the paper. Section 5 provides conclusions.

2 Related Works

2.1 SIMON Family Block Cipher

The SIMON lightweight block cipher was introduced by NSA (National Security Agency) of the USA in 2013 [1]. The SIMON light-weight block cipher family support various block/key size such as below (Table 1).

Table 1. SIMON block cipher parameters [1]

Block size (2n)	Key size (mn)	Word size (n)	Key word (m)	Rounds (T)
32	64	16	4	32
48	72	24	3	36
	96		4	36
64	96	32	3	42
	128		4	44
96	96	48	2	52
	144		3	54
128	128	64	2	68
	192		3	69
	256		4	72

SIMON Encryption Round Function. The round functions of SIMON light-weight block cipher consist of bitwise XOR (\oplus) operation, bitwise AND ($\&$) operation, left circular shift (S^j , by j-bit) operation. The key of SIMON block cipher can be defined by $k \in GF(2^n)$. The round function of SIMON block cipher has key-dependency property and consist of the two-stage Feistel map as follow.

Algorithm 1. SIMON Block Cipher Encryption [1]

Require: Plaintext(x,y), Round number(T).

- 1: **for** i = 0 ... T-1 **do**
 - 2: *tmp* \leftarrow x
 - 3: x \leftarrow y \oplus (Sx & S⁸x) \oplus S²x \oplus k[i]
 - 4: y \leftarrow tmp
 - 5: **return** Ciphertext(x, y)
-

2.2 SPECK Family Block Cipher

The SPECK light-weight block cipher support various block/key size same as SIMON block cipher. In the case of rotation operation, there are two parameters α, β for rotation operation. If block/key size is 32/64, α is 7, and β is 2. Otherwise, α is 8, and β is 3.

SPECK Encryption Round Function. The round functions of SPECK light-weight block cipher consist of bitwise XOR (\oplus), addition modulo 2^n (+) operation, left and right circular shift (S^j, S^{-j} , by j-bit) operation. The round function of SPECK will operate round number (T) time according to SPECK block cipher parameters. The SPECK encryption round function is as below.

Algorithm 2. SPECK Block Cipher Encryption [1]

Require: Plaintext(x,y), Round number(T).

```

1: for  $i = 0 \dots T-1$  do
2:    $x \leftarrow (S^{-\alpha}x + y) \oplus k[i]$ 
3:    $y \leftarrow S^{\beta}y \oplus x$ 
4: return Ciphertext( $x, y$ )

```

2.3 FELICS Triathlon

The Fair Evaluation of Lightweight Cryptographic Systems (FELICS) the open-source software benchmarking framework was proposed by the University of Luxembourg in 2015. This is a benchmark framework targeting for embedded processors such as 32-bit ARM, 16-bit MSP430, 8-bit ATmega128. They benchmarked three different metrics such as execution time, RAM and code size. They tested in three different scenarios such as cipher operation, communication protocol, and challenge-handshake authentication protocol. There are a lot of block ciphers including SIMON, SPECK and etc. Especially, SPECK block cipher was 2nd grade on first triathlon and 3rd grade on the second triathlon.

2.4 ARM-NEON Processor

The NEON is SIMD (Single Instruction Multiple Data) architecture for ARM Cortex-A series [8]. The NEON instruction support 64-bit and 128-bit register and it considers register as a vector. The data types can use on NEON are unsigned/signed 8, 16, 32, or 64-bit, the vector can be an element of NEON data types. The NEON vectors can be categorized as two vectors. The double-word (64-bit) vector and quad-word (128-bit) vector. In the case of double-word (64-bit) vector, it can be made 8 of 8-bit elements or 4 of 16-bit elements or 2 of 32-bit elements or 1 of 64-bit elements, it can connect with D register on NEON and support 64-bit size operation. The quad-word (128-bit) vector can be made 16 of 8-bit elements or 8 of 16-bit elements or 4 of 32-bit elements or 2 of 64-bit elements, it connects with Q register and supports 128-bit size operation.

The NEON registers consist of 16 Q registers and 32 D registers, it can be operated by using 256-byte registers. The NEON has supported SIMD operation since 2005.

The NEON-based cryptographic algorithms such as Salsa20, Poly1305, Curve 25519 and Ed25519 implementations were introduced at CHES 2012 [10]. ARM-NEON SIMD implementation of Grøstl was proposed in CT-RSA'13 [12]. The ARM-NEON SIMD implementation of multiplicand reduction method for the NIST curves was introduced in HPEC 2013 [13]. The Curve41417 ARM-NEON SIMD implementation adopting 2-level Karatsuba multiplication in the redundant representation in CHES 2014 [14]. Seo et al. introduced a novel 2-way Cascade Operand Scanning (COS) ARM-NEON SIMD multiplication for RSA implementation in ICISC 2014 [15, 16]. Seo et al. introduced an efficient modular multiplication and squaring operations for NIST P-521 curve in ICISC 2015 [17].

In the case of the Post-quantum cryptography, Ring-LWE NEON SIMD implementation is proposed [18]. There are two ARM-NEON SIMD implementations of the LEA block cipher. Seo et al. proposed the first NEON SIMD implementation of LEA block cipher in ICISC 2013 [19]. In WISA 2015, Seo et al. proposed the most efficient ARM-NEON SIMD implementation of the LEA block cipher [20].

3 Proposed Method

In this section, we present new efficient parallel implementations of SIMON and SPECK. We firstly improve implementations of SIMON and SPECK for NEON architectures by designing SIMON, SPECK architecture for maximizing usage of NEON registers and using efficient NEON assembly instructions sets for forward/backward alignments, and block cipher operations such as addition, XOR, Rotations. Finally, multiple threads are exploited to use the full capabilities of platforms.

3.1 Optimized NEON Implementation

Park et al. [11] used ARM-NEON intrinsic functions, but we used ARM-NEON assembly instruction sets such as Table 2 describes the basic NEON instruction sets.

Table 2. NEON instruction sets summary

Mnemonics	Operands	Description	Cycles
VADD	Qd, Qn, Qm	Vector addition	1
VAND	Qd, Qn, Qm	Vector bitwise AND	1
VEOR	Qd, Qn, Qm	Vector exclusive-OR	1
VSHL	Qd, Qm, #imm	Vector shift left	1
VSRI	Qd, Qm, #imm	Vector shift right with insert	2

For efficient ARX (Addition, Rotation, eXclusive-OR) operations on SIMON and SPECK block cipher, we used NEON instructions as Table 3 for 16-bit, 32-bit, and 64-bit size blocks.

For efficient implementation, we used `uint16x4_t` data type (D register, 64-bit size, including 2 plain-texts) for SIMON, SPECK32/64 and `uint32x2_t` data type (D register, 64-bit size, 1 plain-text) for SIMON, SPECK64/96, 64/128. In the case of the SIMON, SPECK128/128, 192, 256, We used `uint64x1_t` data type (Q register, 64-bit size, half of plain-text).

In NEON SIMD implementation, the developer can use maximum 256 bytes data of registers. For efficient using of NEON registers, the NEON register allocation for efficient implementation of SIMON is as below. The NEON registers

Table 3. NEON instruction sets for ARX operation (where $d1$ and $q1$ represent destination registers and $d0$ and $q0$ represent source registers)

Block size	Addition	Exclusive-OR	Rotation right by 8-bit
16-bit	$vadd.i16\ d1, d1, d0$	$veor\ d1, d1, d0$	$vshl.i16\ d1, d0, \#8$ $vsri.i16\ d1, d0, \#8$
32-bit	$vadd.i32\ d1, d1, d0$	$veor\ d1, d1, d0$	$vshl.i32\ d1, d0, \#8$ $vsri.i32\ d1, d0, \#24$
64-bit	$vadd.i64\ q1, q1, q0$	$veor\ q1, q1, q0$	$vshl.i64\ q1, q0, \#8$ $vsri.i64\ q1, q0, \#56$

($D0-D11$) are used for plain-texts, registers ($D12-D17$) is used for saving results of rotation left by 1 bit, registers ($D18-D23$) is used for saving results of rotation left by 8 bit, registers ($D24-D29$) is used for saving results of rotation left by 2 bit, and register ($D30$) is used for saving round key in SIMON32/64, 64/96, and 64/128. In the case of SIMON128, the NEON registers ($Q0-Q5$) are used for plain-texts, registers ($Q6-Q8$) is used for saving results of rotation left by 1 bit, registers ($Q9-Q11$) is used for saving results of rotation left by 8 bit, registers ($Q12-Q14$) is used for saving results of rotation left by 2 bit, and register ($Q15$) is used for saving round key.

The NEON register allocation for efficient implementation of SPECK is as below. The NEON registers ($D0-D13$) are used for plain-texts, registers ($D14-D20$) is used for saving results of rotation right by α bit, registers ($D21-D27$) is used for saving results of rotation left by β bit, and register ($D28$) is used for saving round key in SPECK32/64, 64/96, and 64/128. In the case of SPECK128, the NEON registers ($Q0-Q9$) are used for plain-texts, registers ($Q10-Q14$) is used for saving results of rotation left by α bit, rotation left by β bit, and register ($Q15$) is used for saving round key.

In the case of forward and backward alignment for NEON SIMD implementation, We used *vtvn* NEON instruction for transposing of vectors on SIMON, SPECK32/64, 64/96, and 64/128. For SIMON, SPECK128/128, 192, and 256, we exchange the value of registers such as Fig. 1 describes the efficient forward and backward alignment for 64-bit size blocks. This efficient forward and backward alignment method can reduce the number of clock cycles for 64-bit blocks alignments because 16-bit or 32-bit blocks alignment requires several *vtvn* NEON instructions for among inner values of the register and between register.



Fig. 1. Forward and backward alignment for 64-bit blocks

3.2 Thread Level Optimization (OpenMP)

Recent ARM-NEON processors such as ARM Cortex-A9 or Cortex-A53 support multiple cores and multiple thread operation on each different core. It is a kind of SIMT (Single Instruction Multiple Thread) programming method. In this paper, we used OpenMP library for SIMT programming. The SIMT programming supports each core in target device can operate source code, so it is a kind of parallel computing.

4 Results

We used the RaspberryPi3 Model B board which is based on a quad-core 64-bit ARM Cortex A53 (BCM2837) running at 1.2GHz for performance evaluation. We developed ARM NEON implementation by using GCC6.5.1 and OpenMP library version0.0.0 for SIMT (Single Instruction Multiple Threads). For measuring the performance of the proposed method, we calculated the average performance of 10,000 times of SIMON and SPECK encryption including forward and backward alignment procedure. In the case of performance on SIMT, we changed the number of threads from 1 to 8 and calculated the average performance of 10,000 times encryption operation on multi-thread.

We present performance results of the proposed methods by comparing with Park et al. [11]. We firstly present the performance on ARM NEON implementation and then present the performance of using multiple threads. Figure 2 and Table 4 describes performance comparison between Park et al. [11].

In this paper, we further improve the performance of SIMON and SPECK by suggesting methods for maximizing usage of NEON registers for efficient implementation, reducing the number of registers for round key and efficient forward

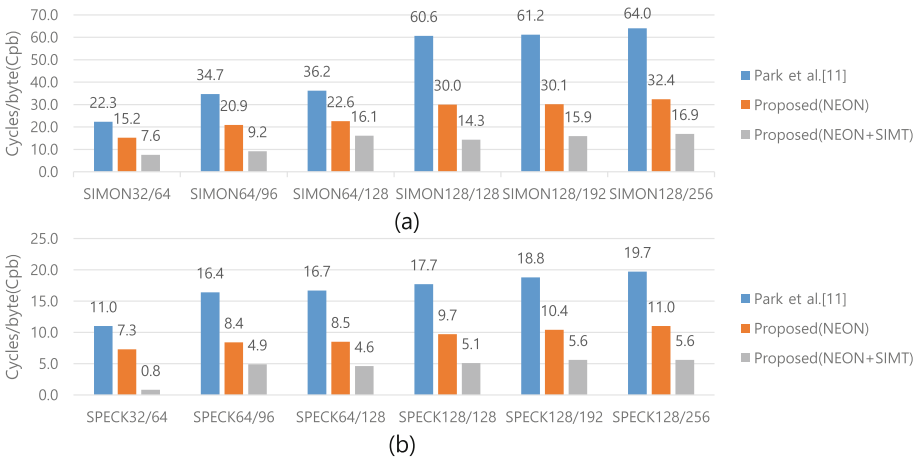


Fig. 2. Performance comparison results ((a) SIMON, (b) SPECK)

Table 4. Performance comparison results between Park et al. [11] and proposed methods

Ciphers		Cycles/byte	Ciphers		Cycles/byte
SIMON32/64	Park et al. [11]	22.3	SPECK32/64	Park et al. [11]	11.0
	Proposed (NEON)	15.2		Proposed (NEON)	7.3
	Proposed (NEON+SIMT)	7.6		Proposed (NEON+SIMT)	0.8
SIMON64/96	Park et al. [11]	34.7	SPECK64/96	Park et al. [11]	16.4
	Proposed (NEON)	20.9		Proposed (NEON)	8.4
	Proposed (NEON+SIMT)	9.2		Proposed (NEON+SIMT)	4.9
SIMON64/128	Park et al. [11]	36.2	SPECK64/128	Park et al. [11]	16.7
	Proposed (NEON)	22.6		Proposed (NEON)	8.5
	Proposed (NEON+SIMT)	16.1		Proposed (NEON+SIMT)	4.6
SIMON128/128	Park et al. [11]	60.6	SPECK128/128	Park et al. [11]	17.7
	Proposed (NEON)	32.4		Proposed (NEON)	9.7
	Proposed (NEON+SIMT)	14.3		Proposed (NEON+SIMT)	5.1
SIMON128/192	Park et al. [11]	61.2	SPECK128/192	Park et al. [11]	18.8
	Proposed (NEON)	30.1		Proposed (NEON)	10.4
	Proposed (NEON+SIMT)	15.9		Proposed (NEON+SIMT)	5.6
SIMON128/256	Park et al. [11]	64.0	SPECK128/256	Park et al. [11]	19.7
	Proposed (NEON)	32.4		Proposed (NEON)	11.0
	Proposed (NEON+SIMT)	16.9		Proposed (NEON+SIMT)	5.6

and backward alignments. By these proposed methods, SIMON block cipher performances are improved average 43.3% and SPECK block cipher performances are also improved average 44.3% than the previous works on SIMON and SPECK NEON implementation [11]. In the case of proposed methods with SIMT (Single Instruction Multiple Threads), SIMON block cipher performances are improved average 69.8% and SPECK block cipher performances are also improved average 74.7% than the previous works [11]. Figures 3 and 4 describe performances

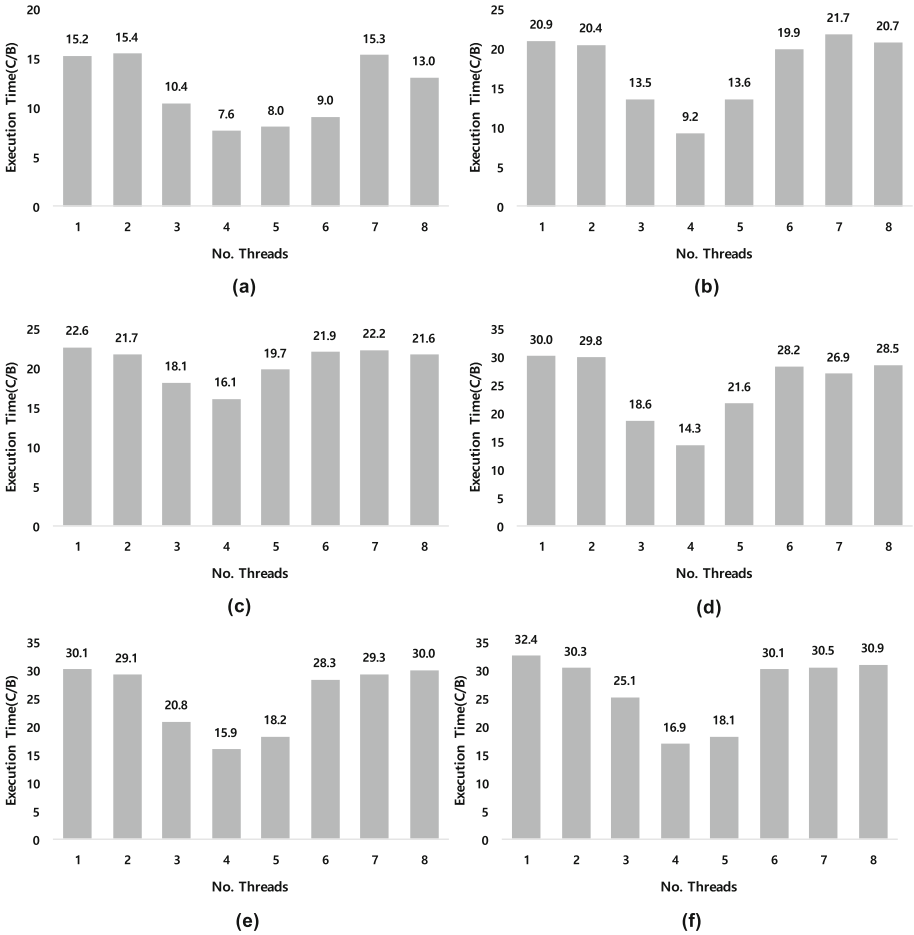


Fig. 3. SIMON block cipher SIMT performance results ((a) SIMON32/64, (b) SIMON64/96, (c) SIMON64/128, (d) SIMON128/128. (e) SIMON128/192, (f) SIMON128/256, C/B: Cycles/byte)

of SIMON block cipher implementation using proposed methods with SIMT at each number of threads Especially, the best performance of proposed methods with SIMT is at 4 threads. Details of our proposed methods and techniques can be founded in our source codes on Github¹.

¹ https://github.com/pth5804/SIMON_SPECK_ARM_NEON.

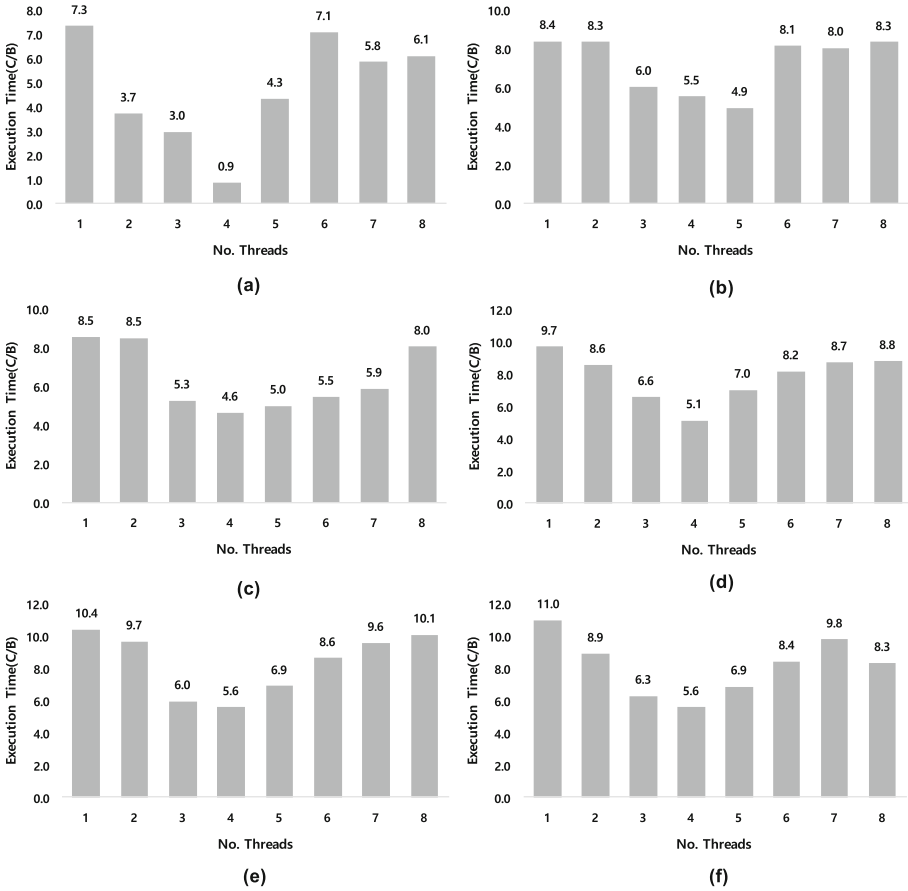


Fig. 4. SPECK block cipher SIMT performance results ((a) SPECK32/64, (b) SPECK64/96, (c) SPECK64/128, (d) SPECK128/128. (e) SPECK128/192, (f) SPECK128/256, C/B: Cycles/byte)

5 Conclusion

In this paper, we proposed new efficient parallel implementation methods for SIMON and SPECK family block ciphers on ARM-NEON SIMD environment. For efficient ARM-NEON parallel implementation, we maximized usage of NEON registers for plaintexts and ciphertexts and reduced the number of registers for round key and efficient forward and backward alignments. Especially, proposed forward and backward alignments for 64-bit blocks can reduce the number of NEON instruction and clock cycles. Then we accelerated the performance of proposed methods by using SIMT (Single Instruction Multiple Threads).

By these proposed methods and accelerating performance based on SIMT, SIMON block cipher performances are improved average 43.3% and 69.8% than the previous works [11]. SPECK block cipher performances are also improved average 44.3% and 74.7%. The proposed methods improved the performance of SIMON and SPECK on ARM-NEON environment. For this reason, we can improve another ARX block cipher such as Simeck. Then we can also apply the proposed method on the other SIMD instruction sets such as SSE and AVX on Intel and AMD processor with using multiple threads by using OpenMP library. We will optimize SIMON and SPECK block ciphers more efficiently by using ARM Assembly and ARM-NEON mixing in the future.

References

1. Beaulieu, R., et al.: The SIMON and SPECK families of lightweight block ciphers. IACR Cryptology ePrint Archive 2013/404 (2013)
2. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK block ciphers on AVR 8-bit microcontrollers. In: Eisenbarth, T., Öztürk, E. (eds.) LightSec 2014. LNCS, vol. 8898, pp. 3–20. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16363-5_1
3. Beaulieu, R., et al.: SIMON and SPECK: block ciphers for the internet of things
4. Dofe, J., et al.: Fault-tolerant methods for a new lightweight cipher SIMON. In: 2015 16th International Symposium on Quality Electronic Design (ISQED). IEEE (2015)
5. Shi, D., et al.: Improved linear (hull) cryptanalysis of round-reduced versions of SIMON. IACR Cryptology ePrint Archive, Report 2014/973 (2015). <http://eprint.iacr.org/2014/973>
6. Tupsamudre, H., Bisht, S., Mukhopadhyay, D.: Differential fault analysis on the families of SIMON and SPECK ciphers. In: 2014 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC). IEEE (2014)
7. Bhasin, S., et al.: A look into SIMON from a side-channel perspective. In: 2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST). IEEE (2014)
8. NEON Technology. <http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0205ik/BABDJECB.html>. Accessed 2015
9. Using NEON Support. <http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0472j/chr1360928368901.html>. Accessed 2015
10. Bernstein, D.J., Schwabe, P.: NEON crypto. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 320–339. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33027-8_19
11. Park, T., Seo, H., Kim, H.: Parallel implementations of SIMON and SPECK. In: 2016 International Conference on Platform Technology and Service (PlatCon), p. 16. IEEE (2016)
12. Holzer-Graf, S., Krininger, T., Pernull, M., Schläffer, M., Schwabe, P., Seywald, D., Wieser, W.: Efficient vector implementations of AES-based designs: a case study and new implemenations for Grøstl. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 145–161. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36095-4_10

13. Faz-Hernández, A., Longa, P., Sánchez, A.H.: Efficient and secure algorithms for GLV-based scalar multiplication and their implementation on GLV-GLS curves. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 1–27. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04852-9_1
14. Bernstein, D.J., Chuengsatiansup, C., Lange, T., Schwabe, P.: Kummer strikes back: new DH speed records. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 317–337. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45611-8_17
15. Seo, H., Liu, Z., Großschädl, J., Choi, J., Kim, H.: Montgomery modular multiplication on ARM-NEON revisited. In: Lee, J., Kim, J. (eds.) ICISC 2014. LNCS, vol. 8949, pp. 328–342. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15943-0_20
16. Seo, H., Liu, Z., Großschädl, J., Kim, H.: Efficient arithmetic on ARM-NEON and its application for high-speed RSA implementation. IACR Cryptology ePrint Archive, 2015:465 (2015)
17. Seo, H., Liu, Z., Nogami, Y., Park, T., Choi, J., Zhou, L., Kim, H.: Faster ECC over $\mathbb{F}_{2^{521}-1}$ (feat. NEON). In: Kwon, S., Yun, A. (eds.) ICISC 2015. LNCS, vol. 9558, pp. 169–181. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30840-1_11
18. Azarderakhsh, R., Liu, Z., Seo, H., Kim, H.: NEON PQCrypto: fast and parallel ring-LWE encryption on ARM NEON architecture
19. Seo, H., Liu, Z., Park, T., Kim, H., Lee, Y., Choi, J., Kim, H.: Parallel implementations of LEA. In: Lee, H.-S., Han, D.-G. (eds.) ICISC 2013. LNCS, vol. 8565, pp. 256–274. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12160-4_16
20. Seo, H., et al.: Parallel implementations of LEA, revisited. In: Choi, D., Guilley, S. (eds.) WISA 2016. LNCS, vol. 10144, pp. 318–330. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56549-1_27

Attact Detections and Legal Aspects



Detecting Online Game Chargeback Fraud Based on Transaction Sequence Modeling Using Recurrent Neural Network

Namsup Lee¹, Hyunsoo Yoon¹, and Daeseon Choi²(✉)

¹ KAIST, Daejeon 34141, Korea

² Kongju National University, Gongju, Choongnam 32588, Korea
sunchoi@kongju.ac.kr

Abstract. We propose an online game money chargeback fraud detection method using operation sequence, gradient of charge/purchase amount, time and country as features of a transaction. We model the sequence of transactions with a recurrent neural network which also combines charge and purchase transaction features in single feature vector. In experiments using real data (a 483,410 transaction log) from a famous online game company in Korea, the proposed method shows a 78% recall rate with a 0.057% false positive rate. This recall rate is 7% better than current methodology utilizing transaction statistics as features.

Keywords: Online game chargeback fraud · Sequence modeling
Recurrent neural network

1 Introduction

Online game users charge “game money”, a kind of virtual currency in the online game world that is spent on purchasing game items such as weapons. This conversion charge from real world to game money is done using online credit card payments.

Two kinds of anomalies are known in these game money charges when someone has lost his or her credit card or credit card information, and the stolen card is used to charge game money, and when a malicious game user charges game money to his or her credit card, spends this in the online game, and files a claim with the credit card company that his or her card or information has been stolen. In the first case, the victim requests a refund and the credit card company grants that request to refund the payments. In the second case, the malicious user can get a refund of the transaction if the game company cannot present evidence that it was intentionally fraudulent. This type of fraud, called *chargeback fraud*, has led to heavy losses for game companies, as they are not usually able to provide sufficient evidence [18, 19].

To detect chargeback fraud, online game companies run rule-based risk management systems defined by experts. Unfortunately, these rules leave out many fraud patterns and are unable to detect newly emerging patterns of fraud. Machine learning based classification methods have been proposed to overcome this limitation [5, 6]. These approaches classify users based on statistics such as the number of countries

where the user has been located and average amount of money charged according to the transaction log. Although these approaches have been able to improve accuracy or recall rate of detection, some normal and abnormal users continue to be misclassified when their transaction statistics are similar with each other. Even if those users' transaction statistics are similar, the sequence of each of their transactions could be quite different. Observing these differences can be used to improve the accuracy of such classification systems. Therefore, this study has proposed a method of user transaction sequence modeling capable of reflecting these differences. The contributions of this work are as follows:

- **First transaction sequence modeling in online game chargeback fraud detection.** We designed a sequence model based upon game money charges and item purchase transactions. To the best of our knowledge, this work is the first attempt to apply sequence modeling for fraud detection in game area. As a result, the proposed method provided a 7% improvement in fraud detection recall rate in experiments using real data.
- **Feature construction including both game money charge and game item purchase transactions for the recurrent neural network.** To model the transaction sequence, we constructed single feature vector that includes features from two different types of transactions and presented recurrent neural network structure for modeling transaction sequence using this feature vector.
- **Evaluation using real transaction data.** We conducted several experiments using real transaction log data provided by a famous online game company in Korea. This work was done to improve the company's chargeback fraud detection system, currently a rule-based detection system. We presented a set of features for this company's data. It is believed that this feature set could be applied to the other game services or similar content businesses. Further, we evaluated the classification accuracy of the proposed method using the real data.

This paper is structured as follows. In the second section, related works on fraud detection in finance and game are introduced. In the third section, the limitations of the statistical detection model are described and the proposed sequence detection model is introduced. In Sect. 4, the proposed model is evaluated based upon some experiments. Section 5 contains a discussion on some considerable points. The paper ends with concluding remarks in Sect. 6.

2 Related Work

FDS (Fraud Detection System) is a complex system designed to identify abnormal transactions by comprehensively analyzing various collected information. It formalizes the usage patterns of users and determines whether a transaction is normal or abnormal based on the established pattern. The previous standard FDS is usually a rule-based system designed by a data analysis expert who analyzes the patterns of abnormal transactions and sets up rules for detecting them [13]. Strong rules make it easier to detect abnormal transactions, but are also more likely to falsely identify normal transactions as abnormal. These rules are also unable to detect some frauds, especially

those following new patterns which will require the expert to make new rules for detecting them. As the volume of data increases, it is necessary to be able quickly and accurately discover usage patterns for users in big data; however, it is impossible for a few human experts to determine such patterns. Some works have applied machine learning to FDS as a feasible method of accomplishing such a task.

A great deal of research has been conducted on detecting fraud in payment transaction data in the financial area. Lim et al. [1] proposed a learning method by giving more heavily weights to recent transactions. Mahmoudi and Duman [2] proposed a method to maximize profits that can be obtained through detection of fraudulent transactions by modifying fisher discriminant function to be cost sensitive to credit card fraud detection problem. Coppolino et al. [3] proposed a fraud detection model for the case of taking and abusing accounts in the mobile banking system: a rule-based learning method and probability-based model are constructed around the logs generated by MMT (Mobile Money Transfer), which are then able to analyze intruders' behavioral patterns to succeed in when committing mobile payment fraud and calculate the possibility of an abnormal transaction. Schaidnageland et al. [4] proposed a model to detect abnormal transactions by analyzing the time series pattern of credit card transactions.

Moving away from purely financial business, research has also been conducted using actual game payment transaction data from MMORPG online games where payment fraud occurs frequently. Woo et al. [5] identified other types of fraud related to payment fraud, analyzed the payment data of each type, and extracted the features necessary for learning via statistical method. In this research, a period of 1 year was divided into 3 months intervals and the statistical features for each section were extracted as learning data, allowing them to present a detection model applying decision tree algorithm. Seo and Choi [6] also proposed a statistical fraud detection model. Their research chose the optimal feature selection for chargeback fraud detection using various methods for data set construction and applying various feature selection algorithms. However, statistical models are limited in their inability to reflect sequence information in data, such that even if the gradient information of some feature's definition of normal and abnormal users is different they will have the same value in a statistical model.

3 Proposed Method

3.1 Sequence Modeling

As mentioned above, statistical modeling is fundamentally limited in that no gradient information in data is reflected from previous transactions. As shown in Fig. 1, a feature for classifying a transaction can be statistically defined as a chunk of previous transactions. If the window size is 3, transaction T3's feature is $\langle 700, 1, 1, \dots \rangle$ which provides the statistical values of c_1 , c_2 and c_3 . In the sample image, T6's feature is also $\langle 700, 1, 1, \dots \rangle$. However, the value of charge amount decreases from 1000 to 550 in the u1 normal user case, whereas it increases from 550 to 1000 in the u2 abnormal

user case. Although the gradient of data is different between normal and abnormal users, a statistical model yields the same data for both. This may result in incorrect classification.

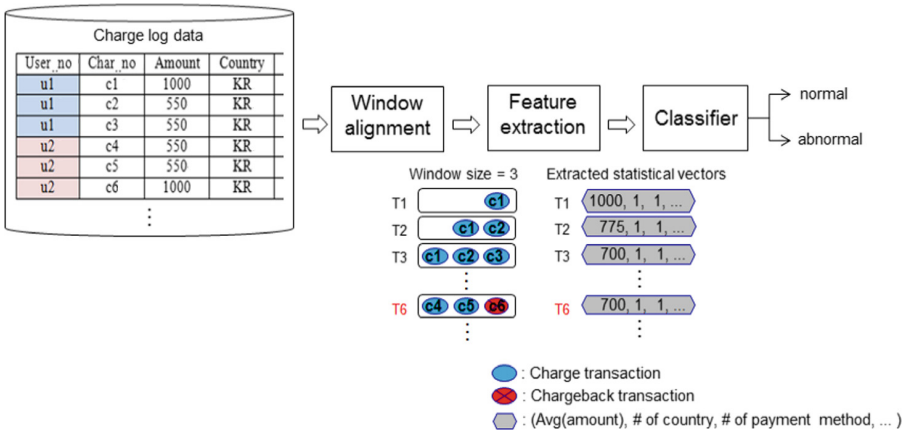


Fig. 1. Statistical modeling of charge data

Therefore, to improve the accuracy of classifications, differences in sequences should be modeled. In Fig. 2, T3's feature is <1000, ...> - <550, ...> - <550, ...> and it is tagged as normal. T6's feature is <550, ...> - <550, ...> - <1000, ...> and it is tagged as abnormal. An RNN (Recurrent Neural Network), which is well-known to show good performance for modeling sequential data, is used to model this sequence feature [7].

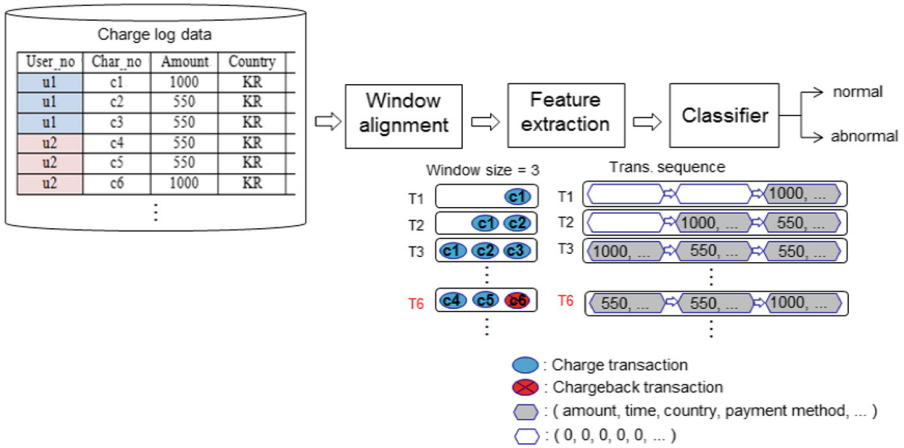


Fig. 2. Sequence modeling of charge data

Feature extraction. The original charge log data is shown in Table 1. ‘Transaction_no’ is used for sorting transactions in chronological order. ‘User_no’ is used for grouping transactions by user. These processes are a part of preprocessing for sequence modeling. ‘Status’ indicates whether or not the transaction has been charged back later, and this field is therefore a class tag.

Table 1. Charge data provided by a game company

Fields	Description
Transaction_no	Transaction identifier
User_no	User identifier
Datetime	Transaction date and time
Country	Country codes
Status	Present the steps of charging
Payment_method	Payment method
Amount	Charge amount
Ip_addr	IP address

Some fields are chosen and processed from the original log data to derive feature vectors. The selected features were ‘Country’, ‘Datetime’, ‘Payment_method’ and ‘Amount’ as shown in Table 2. Because the ‘Country’ and ‘Payment_method’ features are categorical data, the input vector for the features are constructed by one-hot encoding, a feature engineering technique. In the case of the ‘Datetime’ feature, trigonometric [8] functions such as sine and cosine were used to reflect cyclic characteristics. The ‘Amount’ feature was used as it is.

Table 2. New features extracted from the original charge data

Features	Description
Country	Country codes
Payment_method_no	Identifier of payment methods
Time_x	x-coordinate of time in a cyclic form
Time_y	y-coordinate of time in a cyclic form
Amount	Charge amount

Recurrent Neural Net. The machine learning algorithms capable of training sequential model can be either HMM (Hidden Markov Model) based on probability theory or RNN (Recurrent Neural Network) based on an artificial neural network. It has already been proven that if there are enough units in the hidden layer, an RNN has the capability to map a sequential data to another sequential data as we want. Therefore, we adopt a RNN, actually LSTM RNN (Long-Short Term Memory Recurrent Neural Network) that is a variation of RNN to be able to train long sequence of data, as a learning algorithm.

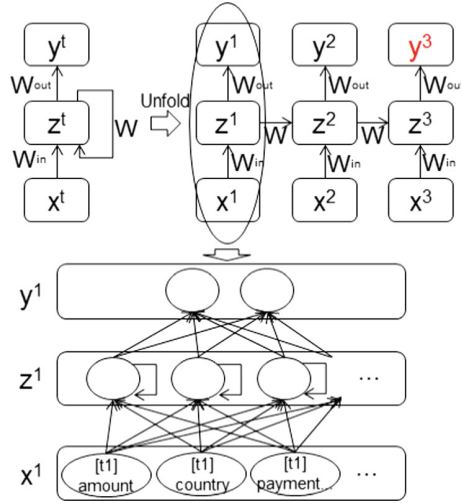


Fig. 3. Basic RNN architecture

The basic architecture of a RNN is that the result of calculating the current input vector becomes a piece of the calculation of the next steps' input vector. The problem dealt with in this work is accurately predicting whether or not the current transaction will be charged back in the future. Therefore, the window aligned dataset has to be modeled forward from previous transactions to the current transaction.

The internal procedure of calculating the output value is as follows. The status of hidden layer at $t(z^t)$ is calculated with the previous status (z^{t-1}) and current input vector (x^t), the result of feature engineering as mentioned above, as shown in Fig. 3. This step is repeated to calculate each time t steps. The formula of this procedure is as follows.

$$z^t = f\left(W^{(in)}x^t + Wz^{t-1}\right) \tag{1}$$

The output of the hidden layer at $t(y^t)$ is calculated with z^t . We use the softmax function for 2-class parameters as the activation function (f), since the problem is a binary classification of whether or not each transaction will be charged back. We take the last step of output as a prediction value.

$$y^t = f^{(out)}\left(W^{(out)}z^t\right) \tag{2}$$

3.2 Purchase-Combined Sequence Modeling

As mentioned above, there are two types of transaction data: charge data and purchase data. These game-dependent types of transaction can present a considerable difference in behavioral pattern, making it possible to differentiate between normal and abnormal

users. Usually, a normal user will make one charge and then purchases game items until the charge amount has been depleted, whereas abnormal users are more likely to make many charges at one time and then purchases game items. The difference is shown in Fig. 4.

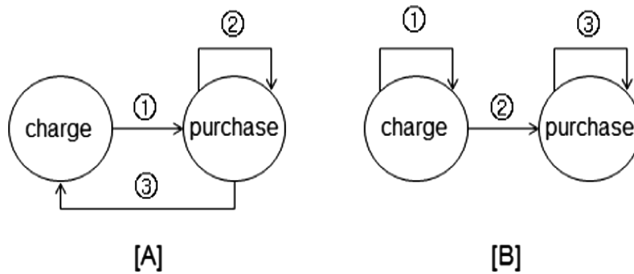


Fig. 4. State diagram of purchase behavioral pattern: [A] normal vs [B] abnormal user patterns

The result of analysis on real game transaction data is as follows: pattern ‘A’ is seen 68% for normal users and 33% for abnormal users, while the ratio for pattern ‘B’ is 32% normal and 67% abnormal users. Thus, sequential modeling that combines purchase transaction has a positive meaningful correlation to classification.

The basic concept of purchase-combined sequence modeling is as follows. The first purchase transaction per dedicated charge transaction is taken (this is sufficient to reflect the features described above), and uncommon features between charge and purchase transactions are solved by padding with 0 (an acceptable solution in RNN). A more detailed description is given in part 3.2.1, Feature Extraction. The whole description is shown in Fig. 5.

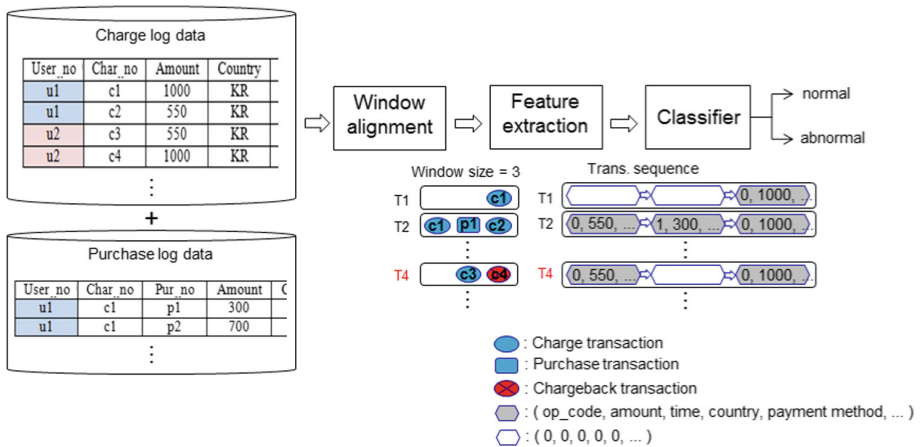


Fig. 5. Purchased-combined sequence modeling

Feature Extraction. Table 3 shows the extracted features for applying purchase-combined sequence modeling to RNN.

Table 3. New features extracted from the original charge and purchase data

Features	Description
Country	Country codes
Payment_method_no	Identifier of payment methods
Time_x	x-coordinate of time in a cyclic form
Time_y	y-coordinate of time in a cyclic form
Amount	Charge amount
Op_code	Identifier of whether type of input vector is for charge transaction or purchase transaction

These extracted features are made by combining some charge and purchase features, consisting of common, uncommon, and new features. We extract only one feature for each of the common features even though this represents two features, one for charge and another for purchase. In order to reflect the purchase behavioral pattern, we extract a new feature named ‘op_code’ for distinguishing types of input vector. When making an input vector for a charge transaction, purchase features are filled with 0 and vice versa.

4 Experiment and Evaluation

4.1 Dataset

Our original dataset, provided by a world-famous game company in South Korea, was collected over 34 months (May 2014 to February 2017). The transaction data is from European users participating in this online game. The data contained 93,520 normal users who never charged back and 621 abnormal users who maliciously charged back at least once. There were 483,410 normal transactions and 3,452 abnormal transactions, as in Table 4.

Table 4. Number of normal and abnormal users and transactions

	# of user	# of transaction
Normal	93,520	483,410
Abnormal	621	3,452

4.2 Evaluation Method

Prior work [6] has used data of purely abnormal users who chargeback all transactions and created a user-level detection system that classified users rather than transactions. However, the real data used in this work contained 379 impurely abnormal users (60% in total abnormal users) who charged back only some of their transactions. This shows

that the evaluation approach in previous work’s evaluation is not appropriate for comprehensive fraud detection. Thus, we compare previous statistical model with the proposed sequence model in transaction-level so as to show that the sequence model is better. We also compare sequence model with purchase-combined sequence model in transaction-level to ascertain that purchase-combined sequence model is better.

Table 5. Confusion matrix

		Prediction	
		True	False
Observation	True	True Positive (TP)	False Negative (FN)
	False	False Positive (FP)	True Negative (TN)

The confusion matrix defined in Table 5 was used as a performance estimation tool. The performance metric was set as the f1-score, since the goal of abnormal transaction detection is to detect the greatest number of abnormal transactions with the highest accuracy. The formula of f1-score is below:

$$f_1 = 2 * \frac{precision * recall}{precision + recall} \tag{3}$$

In the previous works, a decision tree was used as a classifier. However, a decision tree has a disadvantage of large variation in the results or performance. Additionally, it is difficult to regard a decision tree as a generalized model, as a decision tree generated can be different every time. Thus, a random forest that overcomes these shortcomings and has good generalization performance is applied to the statistical model for estimating performance. Scikit-learn [14], the python machine learning library, was used to implement this random forest. For the sequence model, RNN is applied as mentioned above, and the number of units in the hidden layer is set to 250 while the epoch is set to 10 (based on the result of finding optimal epoch, shown in Fig. 6). Keras [15], the python deep learning library, was used to implement the RNN. This is a high-level neural networks API, capable of running on top of TensorFlow, CNTK or Theano. The deep learning framework used in this work is TensorFlow.

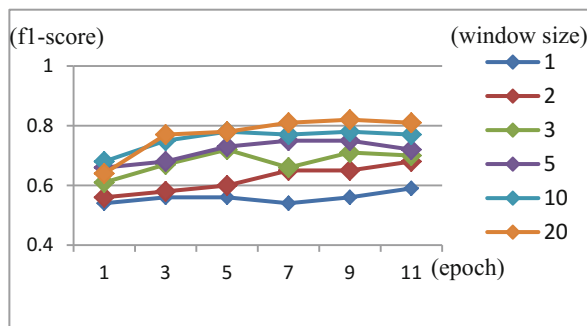


Fig. 6. Epoch of abnormal class for window size

4.3 Experimental Results

The performance for each model was measured with increasing window size. The experimental results are shown in Table 4. Since there is a class imbalance problem between normal and abnormal transactions, all models have a good performance for the normal class ‘0’ as shown in Table 4. Therefore, model evaluation was conducted by extracting the f1-scores of the abnormal class according to window size increase, as shown in Fig. 7(a), and the ROC (Receiver Operating Characteristic) curve of the abnormal class, shown in Fig. 7(b).

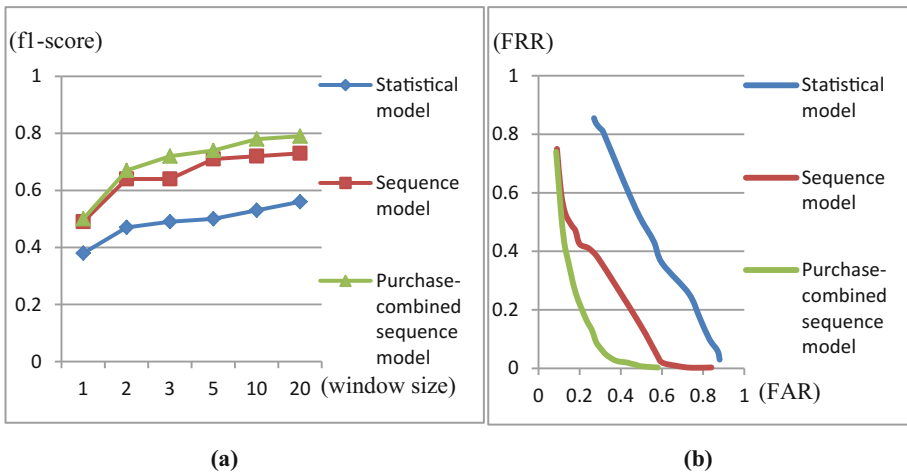


Fig. 7. (a) F1-score for each model (b) ROC curve for each model at window size 20

In terms of f1-score, the performance of the sequence model is seen to be better than that of statistical model while the performance of purchase-combined sequence model is still better than that of sequence model. As the window size increases, the performances of all models increase as additional historical data is available for reflection. The performance of the two sequence models is shown to be bounded at some window size, whereas the performance of the statistical model is shown to be unbounded in Fig. 6. In other words, it is possible to misread the data and assume that the performance of statistical model may be above that of the sequence models if window size increases. Therefore, the f1-score of the statistical model was measured at window size of 50, and it was found that it does not increase above 0.6.

The purchase-combined sequence model is also better than the others in terms of the ROC curve. The ROC curve of the purchase-combined sequence model is closer to a good ROC curve than those of the statistical model and basic sequence models. It is therefore shown that the purchase-combined sequence model would be optimal for a service provider to give an option that which one he will focus on between rejection and acceptance service.

Table 6. Test results of confusion matrix for each model at window size 20.

Model		True	False
Statistical model	True	239797	182
	False	992	734
Sequence model	True	239822	157
	False	432	1294
Purchase-combined sequence model	True	239842	137
	False	381	1345

Table 6 shows the test results of the confusion matrix for each model at a window size of 20. It can be seen that the false positive value decreases dramatically between statistical model and sequence model. It is shown that the purchase-combined sequence model has the smallest false positive value (0.057%) and the highest detection ratio of abnormal transactions at 78%.

5 Discussion

There remain several issues that should be considered for applying the proposed methodology in a real service environment.

The first issue is the time required for classification. In our experiment, the time consumed for 241,705 transactions was 322 s in the server using a Xeon E5-2609 1.7 GHz CPU. For each transaction, 0.0013 s is required. The average daily transaction count given in the experimental data was 703. Therefore, the proposed method will not critically increase the time overhead.

The model construction period is another potential issue. The time required during the experiment for training 241,705 transactions in a sequence of 20 steps each in a RNN model was approximately 1,770 s, when the training epoch is 10. It would therefore be possible to update the model daily in a real world environment.

A third issue worth discussing is the potential of false classifications. In the results of this experiment, the false positive rate of fraud detection for the proposed method is 0.057%. This is lower than the previous method’s 0.076%. In game money charge operations, the game service provider does not directly abandon the transaction. They require additional procedures such as secondary authentication for the suspicious transaction. Thus, a 0.057% false positive rate is so trivial that could be neglected.

The final issue is about resistance to attacks for deceiving machine learning based classifiers. Attacks such as the poisoning attack [9, 10] and evasion attack [11, 12] were proposed. The poisoning attack is performed by injecting poisoned data (e.g. a fraud transaction with normal tag) as training data to a classifier, causing the classifier to produce a false result. In the performed experiment, a charge that is refunded within 6 months (after that the credit card company does not refund the transaction) is defined as abnormal. The attacker should pay for such charge transactions and not get refunded in order to make deceiving transactions that have shape of typical their transaction patterns and are not refunded. As such attackers are in the minority, each has to make

many transactions by themselves to affect the classifier's decision. A classifier is trained with more than 200,000 transactions. In [9], it was shown that about 10% of data must be poisoned in order to degrade the classifier's accuracy by 10%. Therefore the poisoning attack is not practical in the respect of cost effectiveness. In evasion attack designed by making data such that a classifier produces false results, the confidence value of any decision should be required with the classification result of any test data. Of course, the proposed system does not provide any confidence value to users, so that the evasion attack could be impossible. However, in the case of normal users that turn to charging back transactions, they could easily make an evasion attack with only the classification result because they have a sequence of normal transactions that were classified to normal. Also, a classifier is not able to detect an attack that doesn't exist within a given dataset. Therefore, additional research is required for making a classifier capable of defending against those attacks: and making malicious data artificially via GAN (Generative Adversarial Network) [16] and training the data may be one of the solutions that improve the capability of a classifier so that it can defend such attacks.

6 Conclusion

A sequence model for online game fraud detection at the transaction-level was proposed which overcomes the limitations of the currently used statistical model, including misclassification of normal and abnormal users who have different transaction sequences. In addition, a difference in purchasing behavioral patterns was found between normal and abnormal users. Based on this, a purchased-combined sequence model was suggested to reflect the difference. The proposed models were tested with real game transaction data provided by a world-famous game company in South Korea, and results showed that the performance of both developed sequence models was better than that of the existing statistical model, with the purchase-combined sequence model showing the best performance overall.

This work focused on explaining the evaluation of superiority in our proposed model. However, there is a class imbalance problem in the data, which will also require research work for performance improvement. Therefore, future work will try to solve this class imbalance problem by applying a GAN as a novel approach and comparing the performance with existing techniques such as SMOTE [17] or under-sampling. Further research will also be required to defend against such attacks as mentioned in the discussion; and applying GAN as a method may also be capable of making a classifier defendable against such an attack.

Acknowledgments. This work was partly supported by Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (B0717-16-0139, Security Technologies for Financial Fraud Prevention on Fintech) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2016R1A4A1011761).

References

1. Lim, W.-Y., Sachan, A., Thing, V.: Conditional weighted transaction aggregation for credit card fraud detection. In: Peterson, G., Sheno, S. (eds.) *DigitalForensics 2014*. IAICT, vol. 433, pp. 3–16. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44952-3_1
2. Mahmoudi, N., Duman, E.: Detecting credit card fraud by modified Fisher discriminant analysis. *Expert Syst. Appl.* **42**(5), 2510–2516 (2015)
3. Coppolino, L., D’Antonio, S., Formicola, V., Massei, C., Romano, L.: Use of the Dempster-Shafer theory for fraud detection: the mobile money transfer case study. In: Camacho, D., Braubach, L., Venticinque, S., Badica, C. (eds.) *Intelligent Distributed Computing VIII*. SCI, vol. 570, pp. 465–474. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-10422-5_48
4. Schaidnagel, M., Connolly, T., Laux, F.: Automated feature construction for classification of time ordered data sequences. *Int. J. Adv. Softw.* **7**(3), 632–664 (2014)
5. Woo, J.Y., Kim, H.N., Kwak, B.I., Kim, H.K.: Abnormal transaction detection model based on online game payment data analysis. *Korea Inst. Inf. Secur. Cryptol.* **26**(3), 38–44 (2016)
6. Seo, J.H., Choi, D.: Feature selection for chargeback fraud detection based on machine learning algorithms. *Int. J. Appl. Eng. Res.* **11**, 10960–10966 (2016)
7. Hammer, B.: On the approximation capability of recurrent neural networks. *Neurocomputing* **31**(1–4), 107–123 (2000)
8. Trigonometric Function. https://en.wikipedia.org/wiki/Trigonometric_functions. Accessed June 2016
9. Mozaffari-Kermani, M., et al.: Systematic poisoning attacks on and defenses for machine learning in healthcare. *IEEE J. Biomed. Health Inform.* **19**(6), 1893–1905 (2015)
10. Poison attacks against machine learning, Security and spam-detection programs could be affected (2012). <http://www.kurzweilai.net/poison-attacks-against-machine-learning>
11. Vaidya, T., Zhang, Y., Sherr, M., Shields, C.: Cocaine noodles: exploiting the gap between human and machine speech recognition. In: *9th USENIX Workshop on Offensive Technologies (WOOT 2015)* (2015)
12. Szegedy, C., et al.: Intriguing properties of neural networks, preprint arXiv, [arXiv:1312.6199](https://arxiv.org/abs/1312.6199) (2013)
13. Patcha, A., Park, J.M.: An overview of anomaly detection techniques: existing solutions and latest technological trends. *Comput. Netw.* **41**(12) (2007)
14. Scikit-learn Homepage. <http://scikit-learn.org/>. Accessed 15 June 2017
15. Keras Homepage. <https://keras.io/>. Accessed 15 June 2017
16. Goodfellow, I.J., et al.: Generative Adversarial Networks, preprint arXiv, [arXiv:1406.2661](https://arxiv.org/abs/1406.2661) (2014)
17. Chawla, N.V., et al.: SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **16**, 321–357 (2002)
18. <https://chargeback.com/events/money2020-2016/>. Accessed 15 June 2017
19. <http://gametoc.hankyung.com/news/articleView.html?idxno=42921>. Accessed 15 June 2017



The Digits Hidden in the Virtual World: Approximate Estimation Applying Capture and Recapture

Da Mi Hwang and In Seok Kim^(✉)

Korea University, Center for Information Security Technologies (CIST),
145, Anam-ro, Seongbuk-gu, Seoul, Republic of Korea
{dami_hwang, iskim11}@korea.ac.kr

Abstract. In general, game players want their own characters that project themselves to have stronger power and honor in a virtual world. Their aspiration is achieved by pulling up character's level or accumulating wealth in game. Some players, thus, cheat at games in a variety of ways to skip the repetitive and tedious process of gaining experience and wealth that consumes a considerable amount of time and effort. Cheatings are regarded as harmful behaviors toward both game producers and players in good faith, and if the rate of the cheating players exceeds a certain threshold, the game producers will not be able to provide fair and successful services anymore. Therefore, the game producers make various efforts to detect and impose a sanction on the cheaters. In this paper, we propose a method to estimate population size of the cheaters more quickly and accurately, in a relatively shorter time and lower cost than traditional methods, by using the method which is used frequently by ecologists. Among the various ecological estimation methods, Jolly-Seber estimator, based on capture and recapture method, was selected to estimate the characteristics of players in virtual world. Moreover, the video footage recorded for estimation is expected to become a legal evidence for game producers to impose sanctions such as permanent account suspension. Based on the estimated population size, the ratio of the cheaters among ordinary players can be estimated, and this ratio is expected to help the game producers to make swift decisions on the timing of sanctions. In this paper, we estimate the population size of cheating players in *Blade & Soul*, a popular MMORPG game. The total number of cheating players was estimated to be 274,639 players in four selected areas. In 2012, according to official press release of the game producer (NCSOFT Corporation), *Blade & Soul* had 230,000 concurrent users at every second. The number of active users is approximately estimated to be 2,300,000. Using the method proposed in this paper, the rate of cheating players in the game is approximately 11.94%.

Keywords: Online game security · Estimation of cheaters population
Capture and recapture method · Jolly-Seber estimator · *Blade & Soul*

1 Introduction

The online game service is one of the most successful services on the Internet along with the World Wide Web. SUPERCELL, a mobile game company that developed all-time-best games such as *Hay Day* (2012), *Clash of Clans* (2012) and *Boom Beach*

(2014), raised a revenue of \$ 2.3 billion with just 3 games mentioned above in 2016 [1]. Riot Games, a PC-based game producer that published League of Legends (2009), was recorded \$ 1.7 billion in revenues in 2016 [2]. Riot Games also announced that up to 7.5 million players were simultaneously connected to the League of Legends by 2014 [3].

The growing numbers of players who enjoy role-playing games want their own game characters to have more power and honor. Their aspiration is achieved by pulling up character's level or accumulating wealth in game. Most of the game producers then design games in which the cash and items should be essentials for players to gain power and honor. These are given to players as a reward for their considerable time and effort they put in the game. Some players, however, cheat at games in a variety of ways, such as purchasing cash or items through illegal channels or using game bots in order to skip this repetitive and tedious process and achieve greatness in shorter period of time. Cheating has considerable harmful effect on the game producers and players in good faith for the following reasons: [4, 5].

- The economy in game is carefully designed in such a way that game cash has steady value as items are distributed. Cheating, however, leads to devaluation of currency, breaks down the economy in the direction of hyper-inflation, unintended arbitrage transactions, and distorted incentives.
- In-game contents are designed to be consumed for a given period of time on par with level of difficulty. Cheating, however, accelerates the consumption of in-game contents, causing difficulties in operating the game.
- Cheating is related to (1) account theft, (2) identity theft, and (3) personal information theft. It remains a terrible experience for innocent players and ultimately leads to exodus of players.

For this reasons, if the ratio of cheating players exceeds a certain threshold, the game producers will not be able to provide game services eventually. Therefore, the game producers make various efforts to detect cheating players and impose sanctions against them, but these efforts require considerable time, cost, and manpower. In this paper, we propose a methodology to estimate the population size and rate of cheating players in game more quickly and efficiently than traditional methods by using ecological estimation method. This will also help the game producers to determine when to impose sanctions against cheating players.

2 Related Works

Researches related to detection and prevention of the cheating game players can largely classify into first generation, second generation and third generation according to their detection methods [6].

The first generation is divided into a client side and network side. The client side detection method is implemented by introducing security programs into the game client program or CAPTCHA (Completely Automated Public Turing test to tell Computers

and Humans Apart) tests in the authentication phase. The network side detection method is implemented by encrypting the packets or changing the structure of communication protocols. However, these methods excessively increase the minimum system requirements to run the game client programs. Most security programs can be bypassed by reverse engineering, and CAPTCHA tests can also be bypassed by OCR or crowd sourcing. In addition, even if the game producers change the structure of communication protocols, the cheat program makers analyze the changes and optimize their cheat programs in just a few hours. This is an endless battle between the game producers and the cheat program makers. These kind of detection methods rather hurts the convenience of players, simultaneously burdens a client systems and networks, and finally makes players feel frustrated. The second generation is implemented through data mining, HIPs (Human Interaction Proofs) and HOPs (Human Observational Proofs) based on identification of player behavior patterns. It is also performed at the server side to detect the cheating players and impose sanctions against them without compromising the convenience of the players. However, when the game producer detects the cheating players through data analysis mentioned above and imposes sanctions such as an account suspension, they immediately notice the detection patterns and respond to the actions by changing their behavior patterns or methods. They eventually remain in game with extra accounts. In addition, these kind of data analyses cannot detect undisciplined organized groups that supply players with cash, items, cheat programs, etc., in game. The third generation is implemented through analysis of cash or items transaction networks generated by interacting among players, social network analysis using diffusion theory and dynamics model. It is aimed to selectively restrict some of the key players in game and detect not only the cheating players but also the organized groups. It also performed at the server side like the second generation. These kind of restrictions effectively cause economic damages to the organized suppliers and alleviates the economic losses due to large-scale sanctions against the cheating players. However, even if the game producers select some key players as targets through graph analysis mentioned above, for possible legal disputes, it is necessary to monitor them before impose sanctions against them. The professional monitoring agents are called “game masters (GMs)”.

Table 1 summarizes the features of the detection methods by generation, and it can be confirmed that the detection methods require considerable time, cost, and highly trained experts as the generation continues. This is a significant burden on most game producers. Therefore, it is necessary to study the methods of estimating the size of cheating players by effectively using the GMs that can accurately judge cheating and prepare for the legal disputes. Roy et al. in [19] proposed new methods for estimating and identifying cheating players such as gold farmers. They estimated population size of cheating players hidden in MMOGs (Massively Multiplayer Online Games) using capture and recapture technique which is one of the popular techniques in ichthyology. They, then, identified them using graph partitioning algorithm in the coextensive networks.

Table 1. Features of detection methods by generation

Generation	Features	
1 st	Description	<ul style="list-style-type: none"> - Signature based - Client-side detection - Network-side detection
	Client-side methods	<ul style="list-style-type: none"> - Security programs (e.g., anti-virus, anti-debugger, memory protector, process protector and so on) - CAPTCHA tests
	Network-side methods	<ul style="list-style-type: none"> - Packet encryption - Encryption key changes - Communication protocol structure changes
	Bypass methods	<ul style="list-style-type: none"> - Reverse engineering - Protocol analysis - OCR reading - Crowdsourcing
	Issues	<ul style="list-style-type: none"> - Making players annoying - Giving a burden to client systems and network - Endless battle between game producers and cheating program makers
2 nd	Description	<ul style="list-style-type: none"> - Player behavior patterns based - Data mining - Server-side detection
	Server-side methods	<ul style="list-style-type: none"> - Game play pattern analysis [7] - Communication pattern analysis [8] - Chatting pattern analysis [9] - Party play analysis [10] - Action sequence analysis [11] - Self-similarity analysis [12]
	Bypass methods	<ul style="list-style-type: none"> - Changing behavior patterns - Making a variant of existing cheating tools
	Issues	<ul style="list-style-type: none"> - Detecting only cheating players, not entire related groups
3 rd	Description	<ul style="list-style-type: none"> - Surgical strike - Server-side detection
	Server-side methods	<ul style="list-style-type: none"> - Surgical analysis [13] - Trade network analysis [14] [15] - Social network analysis [16] - Contagion analysis [17] - Malicious behavior spread analysis [18] <p>* Detecting the entire groups related to cheating</p>

3 Methodology

In the ecology, to understand how geographic, physical, and biological factors affect species distribution and density, changes of population size are measured. The simplest method for estimating the changes is to investigate the entire population size. However, in general, even when targeting plants as well as animals, the methods of estimating the

entire population size are rarely applied due to the problems regarding with habitat destruction and artificial interference along with the time, cost, and manpower required for the investigation [20]. For this reasons, a number of statistical techniques have been studied in ecology to estimate the population size of individuals, which is quite similar to the problem of estimating the population size of cheating players in game. In this paper, we estimate the population of them using ecological estimation method.

The ecology that studies the relationship between living things and the surrounding environment defines the population as a group of allogeneic individuals living in a certain area, and the population size as the number of individuals included in the group at a specific time. There are various methods for estimating the population size depending on characteristics and distribution forms of individuals, typically, quadrat method, distance method, removal method and capture and recapture method [21, 22]. In general, it is difficult to estimate the entire population size of cheating players in game because they always conceal themselves and move as a herd. Because of their properties, sampling methods generally used in statistics such as bootstrapping cannot be used. Especially, they have high mobility such as animals. This high mobility causes a lot of bias in the method (the ratio of cheating players in a sample * the entire normal players) of estimating the size of all them by randomly extracting n size samples. Each time a sample is extracted, their ratio would be very different. Thus, ecological method is required. Among the representative methods used in ecology, a quadrat method and a distance method is not suitable for estimation because cheating players have high mobility such as animals. A removal method is also not suitable because they cannot be completely eliminated. Therefore, a capture and recapture method aimed at individuals which has high mobility and moves as a herd is quite suitable. This method is divided into two steps; the capture step and the recapture step [23, 24]. In the first step shown in Fig. 1(a), a set of individuals is captured and marked distinctively. After enough time has elapsed to ensure that marked individuals in the first capture diffuse sufficiently, the second step is embarked upon. In the second step shown in Fig. 1(c), a set of individuals is recaptured and individuals marked in first capture are examined [25]. After this process is repeated N times as scheduled, the population size is estimated using an estimator such as Peterson, Schnabel and Jolly-Seber estimator. The game world has inflow and outflow of cheating players over time, and they have an intensive density. So, we use Jolly-Seber estimator to estimate the population size.

The process of estimating is as follows. First, select the time and investigation areas. The selected areas should represent the entire. Second, observe the cheating players in the selected areas, and simultaneously record their nicknames for marking. Repeat this process on a schedule and check the number of marked cheating players based on the nicknames recorded in the previous experiments. Third, estimate the population size using Jolly-Seber estimator. This estimates the population size through the proportion of the marked among the whole observed individuals and the size of the population. The conceptual equation is as follows.

$$\text{Population size} = \frac{\text{Size of marked population}}{\text{Proportion of marked individuals}}$$

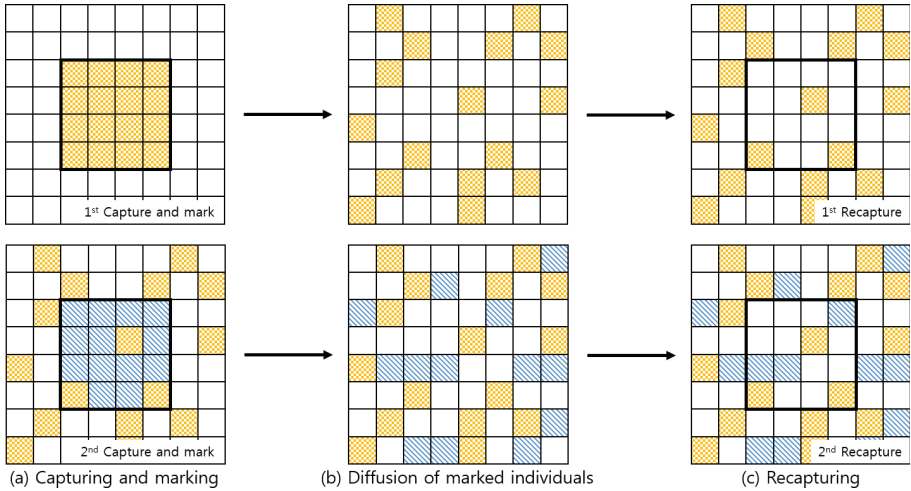


Fig. 1. Illustration of a capture and recapture method

In other words, it is estimated through the size and proportion of cheating players which have the same nickname as observed in the previous experiments. Table 2 is describing the variables used in the Jolly-Seber estimator. s_t is tentatively equal to n_t since making of cheating players does not affect the reduction of their number.

Table 2. Notation of Jolly-Seber estimator

Notation	Description
m_t	Number of marked individuals observed in time t
u_t	Number of unmarked individuals observed in time t
n_t	Number of individuals observed in time $t = m_t + u_t$
s_t	Number of individuals survived at time $t = n_t - \text{accidental removals}$
R_t	Number of individuals survived at time t and observed again in some later time
Z_t	Number of individuals marked from before time t and observed after time t

The proportion of the marked cheating players ($\hat{\alpha}_t$) is estimated as the ratio of the marked (m_t) among all observed individuals (n_t) at time t (see Eq. 1). Where ‘+1’ is used for correction (Seber 1982, p. 204).

Proportion of marked individuals

$$\hat{\alpha}_t = \frac{m_t + 1}{n_t + 1} \tag{1}$$

The size of the marked cheating players (\hat{M}_t) is difficulty estimated relative to estimation of the proportion. It is estimated by using the following variables (see Eq. 2): the number of repeatedly observed individuals since observed at time t (R_t),

the number of individuals survived at time t ($s_t = n_t$), the number of repeatedly observed individuals from the beginning to time $(t-1)$ (Z_t), and the number of marked individuals at time t (m_t).

Size of marked population

$$\widehat{M}_t = \frac{(s_t + 1)Z_t}{R_t + 1} + m_t \tag{2}$$

The population size (\widehat{N}) is estimated using the proportion estimator and size estimator (see Eq. 3). It is also be transformed into $T_1(\widehat{N})$.

Size of population

$$\widehat{N} = \frac{\widehat{M}_t}{\alpha_t} \text{ Transform of the estimate } \widehat{N} \text{ by } T_1(\widehat{N})$$

$$T_1(\widehat{N}) = \log_e(\widehat{N}) + \log_e \left[\frac{1 - (\frac{p}{2}) + \sqrt{1-p}}{2} \right] \text{ where, } p = \frac{m}{\widehat{N}} \tag{3}$$

The variance for the population size ($\text{Var}[\widehat{N}] = \text{Var}[\widehat{T}_1 N]$) is required to estimate the confidence interval. It is estimated using the modified population size estimator (see Eq. 4).

Variance of population

$$\widehat{\text{Var}}[T_1(\widehat{N}_t)] =$$

$$\left(\frac{\widehat{M}_t - m_t + s_t + 1}{\widehat{M}_t + 1} \right) \left(\frac{1}{R_t + 1} - \frac{1}{s_t + 1} \right) + \frac{1}{m_t + 1} - \frac{1}{n_t + 1} \tag{4}$$

The 95% confidence interval is estimated using the proportion estimator and its variance. The lower bound of confidence interval is estimated as T_{1L} , the upper bound is estimated as T_{1U} (see Eq. 5).

Confidence limits for population size

$$\frac{(4e^L + n_t)^2}{16e^L} < N < \frac{(4e^U + n_t)^2}{16e^U}$$

$$L = T_{1L} = T_1(\widehat{N}) - 1.6 \sqrt{\widehat{\text{Var}}[T_1(\widehat{N})]} \tag{5}$$

$$U = T_{1U} = T_1(\widehat{N}) + 2.4 \sqrt{\widehat{\text{Var}}[T_1(\widehat{N})]}$$

Assumptions of Jolly-Seber estimator is following:

- All individuals have the same observation probability.
- The survival probability of all the marked individuals is constant from time t to time $(t+1)$.

- All the marked individuals with do not lose their marking, and the investigators do not omit that.
- The experiments do not affect the number of individuals.

In summary, Jolly-Seber estimator repeatedly observes and records the cheating players on a schedule, then estimates the population size based on the size and proportion of the marked individuals.

4 Experiment

4.1 Target and Schedule

The target of experiments is the MMORPGs (Massively Multiplayer Online Role Playing Games) Blade & Soul (2012). The game players can freely choose their own character's tribe, occupation, gender, appearance and so on. They are also given a variety of quests and contents in the game. So, the game has attracted a lot of popularity among players since its launch, at the same time many cheat programs that automate hunting and questing have been introduced. The cheat programs are classified according to (1) purpose, (2) implementation type such as software or hardware, and (3) operation type such as OOG(out of game client) bots or IG(in game client) bots. Especially, the OOG bots operate separately from game client programs and are implemented with sufficient analysis of game clients, servers, and communication protocols between them. On the other hand, the IG bots work with game client programs and are implemented in a way that hooks game processes. In general, the IG bots are more difficult to detect than the OOG bots, and the IG bots exist in games where OOG bots exist. Most of cheat programs also can be implemented in hardware type such as keyboard, mouse, and USB. The experiments were conducted on the ISIM-JEONSIM server among 33 servers that players access to enjoy Blade & Soul. The world of the game is geographically classified into MUIL MOUNTAINTOP, JERYONG FOREST, GREAT DESERT, MOONWATER PLAINS, HALL OF MUSHIN, SILVERFROST MOUNTAINS, PACHEON CITY, GUNWON CITY, SKY FARM, and SEORAK. Among the above 10 areas, especially, JERYONG FOREST, GREAT DESERT, MOONWATER PLAINS, and SILVERFROST MOUNTAINS in which the huge hunting fields and dungeon are concentrated were selected as the target areas (See Fig. 2).

The monsters equipped with high attack damage and defense strength are called 'named monster'. The players will be given a lot of gold and precious items as a reward if they win the battle against named monsters. Thus, huge hunting field and dungeons where these named monsters appear are easier for players to acquire a lot of gold, items, and experience points than other areas, so many players mostly remain there to hunt. The experiments were conducted five times at intervals of three days from May 19, 2017. The game producer (NCSOFT Corporation) conducts system checks and patches once a week, every Wednesday. When the system is patched, most of the existing cheat programs are blocked from accessing to the system. Therefore, the cheat



Fig. 2. Blade & Soul world map

program makers optimize their programs according to the patch schedule. If the optimization is not done in a timely fashion, the cheat programs are no longer worthy of use. And cheat programs that stay in the same place and repeat anomalous behavior are easily detected and deterred by normal players or detection systems empirically. Thus, they are programmed to accomplish their objectives by going around.

So, we set a week as an activity cycle of cheat programs and expect the programs to be fully spread in three days. All five experiments were conducted for three hours after 6:30 pm, because a preliminary investigation found that most cheating players were observed after that time. The interval of three days is expected to be enough time which all marked cheating players spread randomly on the whole population.

4.2 Population Estimation

We observed the players for a certain amount of time in the selected area, and confirmed the number of cheating players (the general criteria for cheating players are described in the following subsection 4.3 in detail). Because players can move freely within game, all the observations had to be recorded for making of them. Then, we recorded the nicknames separately through the video. We repeated this on a schedule and eventually checked the number of marked cheating players. The total number of individual observed in 5 experiments were 1,014 individuals ($\sum n_t$), and an average of 203 individuals were observed and marked for each turn. Out of these numbers of individuals, the total number of marked individuals repeatedly observed for each turn were 56 individuals ($\sum m_t$), and an average of 11 marked individuals were repeatedly observed. Table 3 shows the observed capture and recapture data in 5 experiments. Based on this, the population is estimated to be 8,322 individuals. The lower and upper bound is estimated to be 3,560~30,234 individuals at the 95% confidence. Thus, assuming that the distribution of cheating players is uniform in 33 servers provided by

Blade & Soul, the total number of cheating players was estimated to be 274,639 individuals. The lower and upper bound is estimated to be 117,480~997,722 individuals at the 95% confidence.

Table 3. Capture and recapture data of the cheaters in Blade & Soul

Time of last capture	Time of capture				
	1	2	3	4	5
1		3	0	2	0
2			4	15	5
3			m_3	8	13
4					6
Total marked (m_t)	0	3	4	25	24
Total unmarked (u_t)	188	191	197	195	187
Total observed ($n_t = s_t$)	188	194	201	220	211

Z_3

 R_3

4.3 Features of Individuals

The cheating players using game bot programs have a habit of acting in groups. The composition of the group varies according to the type of monsters, the tribes specializing in close combat and ranged combat are mainly observed. All of the bots in the herd have similar nicknames, and they are the same tribe, occupation, gender, and they show a considerable resemblance in terms of appearance compared to the ordinary players who customize their characters. They are also significantly different from the ordinary players in terms of behavior. They behave unnaturally, repetitively and response with an incredible speed. They are usually found in the hunting field or at the entrance of dungeon where named monsters appear. Figure 3 shows bots found in the huge hunting field. The bots were also supported social activities such as chatting, party play and so on. Of course, most chats are meaningless, and after 5 to 10 min they leave the party, but it is clear that they mimic the social activities of the ordinary players.



Fig. 3. Cheating players in the hunting field

5 Conclusion

In this paper, we estimate the population size of cheating players based on capture and recapture method which is one of the frequently used estimation methods in ecology in Blade & Soul, a MMORPG. The total numbers of cheating player were estimated to be 274,639 players in four areas. The lower and upper bound is estimated to be 117,480–997,722 individuals at the 95% confidence. The game industry generally estimates concurrent users * 10 as active users. According to the game producer, in 2012, Blade & Soul recorded 230,000 concurrent users at every second. Therefore, the estimated number of active users is approximately 2,300,000. The number of active users is approximately estimated to be 2,300,000. Using this, the rate of cheating players in the game is approximately estimated to be 11.94%.

Previous studies related to detecting the cheating players have focused on analyzing the logs to detect cheating players. However, these methods require considerable time, cost, and trained data analysts. Also, additional monitoring and capturing is needed to prepare for legal disputes. This paper proposed a method of estimating population size of cheating players quickly and accurately with relatively shorter time and lower cost by using the method used frequently by ecologists. Among the various ecological estimation methods, we selected Jolly-Seber estimator considering the characteristics of players in virtual world. In addition, it is expected that the videos recorded in experiments will become an evidence for legal disputes due to sanctions such as permanent account suspension. Based on the estimated population size, it is possible to estimate the ratio of cheating players, and this ratio is also expected to help game producers to make decisions on the timing of sanctions. In order to apply the capture and recapture method in practice, it is necessary to evaluate the Jolly-Seber estimator and to reduce the difference between the upper and lower bounds. So, we will carry out our study on this in the future.

Acknowledgement. This research was supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under the “Employment Contract based Master’s Degree Program for Information Security” supervised by the KISA (Korea Internet & Security Agency) (H2101-17-1001).

References

1. Takahashi, D.: With just 3 games, Supercell made \$924M in profits on \$2.3B in revenue in 2015 [Internet]. VentureBeat (VB) (2017). <https://venturebeat.com/2016/03/09/with-just-3-games-supercell-made-924m-in-profits-on-2-3b-in-revenue-in-2015>. Accessed 1 May 2017
2. Why Riot Games Is Inc.’s 2016 Company of the Year [Internet]. Inc.com (2017). <https://www.inc.com/magazine/201612/burt-helm-lindsay-blakely/company-of-the-year-riot-games.html>. Accessed 1 May 2017
3. League Players Reach New Heights in 2014 [Internet]. Riot Games (2017). <http://www.riotgames.com/articles/20140711/1322/league-players-reach-new-heights-2014>. Accessed 1 May 2017
4. Keegan, B., et al.: Dark gold: statistical properties of clandestine networks in massively multiplayer online games. In: 2010 IEEE Second International Conference on Social Computing (SocialCom). IEEE (2010)
5. ID Theft, RMT & Lineage [Internet]. Terra Nova (2017). http://terranova.blogs.com/terranova/2006/07/id_theft_rmt_nc.html. Accessed 1 May 2017
6. Kim, H.K., Woo, J.: Detecting and Preventing Online Game Bots in MMORPGs, pp. 1–8 (2015)
7. Chen, K.-T., Hong, L.-W.: User identification based on game-play activity patterns. In: Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games. ACM (2007)
8. Kang, A.R., Woo, J., Kim, H.K.: Data and text mining of communication patterns for game BOT detection. In: Proceedings of the 3th International Conference on Internet (2011)
9. Kang, A.R., Kim, H.K., Woo, J.: Chatting pattern based game BOT detection: do they talk like us? KSII Trans. Internet Inf. Syst. **6**(11), 2866–2879 (2012)
10. Kang, A.R., et al.: Online game BOT detection based on party-play log analysis. Comput. Math Appl. **65**(9), 1384–1395 (2013)
11. Lee, J., et al.: I know what the BOTs did yesterday: full action sequence analysis using Naïve Bayesian algorithm. In: Proceedings of Annual Workshop on Network and Systems Support for Games. IEEE Press (2013)
12. Lee, E., et al.: You are a game BOT!: uncovering game bots in MMORPGs via self-similarity in the wild. In: Proceedings of Network and Distributed System Security Symposium (NDSS) (2016)
13. Kwon, H., et al.: Surgical strike: a novel approach to minimize collateral damage to game BOT detection. In: Proceedings of Annual Workshop on Network and Systems Support for Games. IEEE Press (2013)
14. Woo, K., et al.: What can free money tell us on the virtual black market? ACM SIGCOMM Comput. Commun. Rev. **41**(4), 392–393 (2011)
15. Kwon, H., et al.: Crime scene reconstruction: online gold farming network analysis. IEEE Trans. Inf. Forensics Secur. **12**(3), 544–556 (2017)
16. Woo, J., Kang, A.R., Kim, H.K.: Modeling of BOT usage diffusion across social networks in MMORPGs. In: Proceedings of the Workshop at SIGGRAPH Asia. ACM (2012)

17. Woo, J., Kang, A.R. Kim, H.K.: The contagion of malicious behaviors in online games. In: ACM SIGCOMM Computer Communication Review. ACM (2013)
18. Ki, Y., Woo, J., Kim, H.K.: Identifying spreaders of malicious behaviors in online games. In: Proceedings of the 23rd International Conference on World Wide Web. ACM (2014)
19. Roy, A., et al.: The ones that got away: false negative estimation based approaches for gold farmer detection. In: 2012 International Conference on and 2012 International Conference on Social Computing (SocialCom), Privacy, Security, Risk and Trust (PASSAT). IEEE (2012)
20. Krebs, J.R., Davies, N.B.: Behavioural ecology: an evolutionary approach. John Wiley & Sons, Hoboken (2009)
21. McClintock, B.T., et al.: Seeking a second opinion: uncertainty in disease ecology. *Ecol. Lett.* **13**(6), 659–674 (2010)
22. Smith, T., Smith, R.: Elements of Ecology, 1st edn. Pearson, Boston (2015)
23. Choi, S.-Y., Rho, P.: Population size estimation of the *Kaloula borealis* in the Daemyung retarding basin. *Korean J. Environ. Ecol.* **30**(4), 684–693 (2016)
24. Bae, D.-Y., et al.: Applying the Jolly-Seber model to estimate population size of Miho Spine roach (*Cobitis choii*) in the Backgok stream, Korea. *Korean J. Ecol. Environ.* **45**(3), 322–328 (2012)
25. Krebs, C.: Ecological Methodology, 2nd edn. Addison Wesley Longman, Menlo Park (1999)



Legal Consideration on the Use of Artificial Intelligence Technology and Self-regulation in Financial Sector: Focused on Robo-Advisors

Keun Young Lee, Hun Yeong Kwon, and Jong In Lim^(✉)

Graduate School of Information Security, Korea University, 145 Anam-ro,
Seongbuk-gu, Seoul, Republic of Korea
{starnyou, khy0, jilim}@korea.ac.kr

Abstract. Artificial Intelligence (AI) technology is being used throughout the industry due to the introduction of the era of the Fourth Industrial Revolution. In the financial industry, AI technology is used in sales and marketing, fraud and illegality prevention, credit evaluation and screening, chat-bot and etc. The robo-advisor can apply the AI technology in case of investment advisory to provide a large and cost-effective portfolio of investment information. It also has positive function to the field in the fact that it has ability to generate popular investors and create new customers and services. However, robo-advisor that uses AI is still at its initial stage in introducing the technology and there are currently legal, institutional and policy limitations in providing comprehensive and customized advisory services. Thus, at first, this paper will consider the area of legal argument on the issues related to AI on the legal status and liability, financial IT, security and privacy. And focused on robo-advisor, the main issues concerning the current legal system and security self-regulatory method are elucidated and analyzed to provide the basic direction of regulation for development of utilization of AI technology in financial sector. In an environment that is shifting from ex-ante regulation to ex-post regulation, which is a current paradigm of financial IT security regulation, in order to modernize the regulations for the digital age, we propose specific measures to strengthen the use of regulatory sandbox as an autonomous regulatory scheme for the use of new technologies such as AI.

Keywords: Deep learning · Artificial Intelligence · Robo-advisors
Self-regulation · Information security · Personal information protection
Privacy

1 Introduction

According to the World Economic Forum (WEF), AI will lead the fourth industrial revolution and through technological convergence with robot, AI, IoT, etc. it is expected to be the source of new growth engine. Also, due to the influence of the Lee se-dol 9 dan rank and AlphaGo (5 matches done during 9th ~ 15th of March in 2016), with the re-recognition of the introduction to the fourth industrial revolution phase, attracts attention throughout industries on AI technology. According to MyPrivate

Banking, the size of assets managed by robo-advisor around the world is expected to grow 23 times from \$ 20 billion in 2015 to \$ 450 billion in 2020. In the United States, where robo-advisor is most developed, the proportion of assets managed by robo-advisor services among total investment assets is expected to rise from 0.5% in 2015 to 5.6% in 2020 [1]. In Korea, the Financial Services Commission identified and operates a robo-advisor financial regulatory test bed plan through the Financial Market Advisory Initiative (2016.3).

Therefore, in this paper, we raise basic problems on legal argument related to AI and discuss the major issues on regulation of IT finance and security, and the main issues of self-regulation on utilization of AI technology in the financial sector mainly focusing on robo-advisor. Therefore, the issues that are required to be discussed on the utilization of AI technology in the field of law which is expected to not exist in the country. In addition, to maximize the use of robo-advisor and to develop a rational way to improve regulation, preemptive self-regulation measures using regulatory sandbox will be provided in the paper.

2 Theoretical Background

2.1 Concept of Artificial Intelligence (AI) and Robo-Advisor (RA)

Artificial Intelligence (AI) can be defined as a technology or science that researches the methodology or feasibility that can be created or made using the conceptual framework and tools of computer science that is created based on human intelligence, such as cognition, reasoning, and learning.

Table 1. Views of AI fall into four different perspectives

	Human-like intelligence <Strong AI> <i>Systems that think like humans</i>	"Ideal" intelligent/pure rationality <Weak AI> <i>Systems that think rationally</i>
Thought/ reasoning	<ul style="list-style-type: none"> • "The exciting new effort to make computers think, machines with minds, in the full and literal sense" (Haugeland, 1985) • "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning..." (Bellman, 1978) • Approches on the cognitive modeling 	<ul style="list-style-type: none"> • "The study of mental faculties through the use of computational models" (Charniak and McDermott, 1985) • "The study of the computations that make it possible to receive, reason, and act" (Winston, 1992) • Approches on the formalizing "laws of thought"
	<i>Systems that act like humans</i>	<i>Systems that act rationally</i>
Behavior/ actions	<ul style="list-style-type: none"> • "The art of creating machines that perform functions that require intelligence when performed by people" (Kurzweil, 1990) • "The study of how to make computers do things at which, at the moment, people are better" (Rich and Knight, 1991) • Approches on the turing test 	<ul style="list-style-type: none"> • "A field of study that seeks to explain and emulate intelligent behavior in terms of computational processes" (Schalkoff, 1990) • "The branch of computer science that is concerned with the automation of intelligent behavior" (Luger and Stubblefield, 1993) • Approches on the rational agent

The first use of AI was Turing's "Computing Machinery and Intelligence" published in the 1950 Philosophical Journal MIND, which states that machines can be considered thinking "If the reactions of the computers cannot distinguish from those of human reactions" [2]. The beginning of academic research was by McCarthy et al. Shannon at the 1956 Dartmouth Conference insisting that "All aspects of learning or all other characteristics of intelligence can be described precisely by machine and it can be simulated" [3]. Searle distinguishes between Weak AI and Strong AI on the feasibility of realizing human intelligence or consciousness in the machine, not the body, in the base of philosophical grounds [4]. Russell and Norvig classify AI into four perspectives as shown in Table 1 by two dimensions (thinking versus acting, human versus rational) [5]. Weak AI corresponds to 'a system that reasonably thinks or behaves', and Strong AI corresponds to 'a system that thinks or behaves like a human'.

Robo-Advisor is a compound word of Robot and Advisor. It is an online asset management service that automatically 'constructs', 'rebalances', and manages 'portfolio' based on individual investment trends with technology such as algorithms, big data analysis and AI. The robo-advisor minimizes human intervention to provide an investment portfolio and more advanced AI technology is expected to be applied.

2.2 Concept of Self-regulation and Regulatory Sandbox

Self-regulation is can be defined as regulatory approach that pursues rationalization and efficiency of through participation of private sector in the regulatory sector that was once considered to be the governmental sector and the government sector actively supporting the activities and roles of the private sector.

Financial IT autonomous security system refers to a change in the security system under autonomous regulation, in which market participants self-regulate themselves in the framework of principle regulation and self-check for implementation due to absence of regulation as financial security regulatory paradigm shifts from non-regulatory to principle-based self-regulation.

Regulatory Sandbox is a financial regulation test bed that is similar to a sandbox that built on a playground that allows children to play safely, but combined with financial environment so that innovative financial services can be tested for a period of time before actual market launch under the regulatory environment.

3 Regulatory Trends and Key Issues on Robo-Advisor

3.1 Overseas Regulatory and Regulatory Sandbox Utilization Trend on RA

In other countries, under self-regulatory system, the supervisory authority or the self-regulatory organization presents the Consultation (or Discussion) paper to establish the regulatory policy for RA and establish policies to protect investor.

In the United States, the supervisory authorities, the Securities and Exchange Commission (SEC) and the Financial Industry Regulatory Authority (FINRA) issued "Investor Alert: Automated Investment Tools" related to RA (2015.5.8). And through

the announcement of the Regulatory Compliance Guidance for Online Robo-Advisers (2017.2.), under the US Investment Advisors Act, SEC has emphasized to assign fiduciary obligation to the RA and provide clear information disclosure, provide appropriate advice, and establish effective compliance policies. In the EU, the European Securities and Markets Authority's (ESMA), the European Banking Authority (EBA), the European Insurance and Occupational Pensions Authority (EIOPA) presented the characteristics of RA and potential investor risk through "Joint Discussion Paper on Automation Financial Advice (2015.12.4)". England's Her Majesty's Treasury and the Financial Conduct Authority (FCA) presented "the Financial Advice Market Review (2016.3)". It considered a number of issues, including the regulatory and legal framework, the economics of providing advice, consumer engagement and the role of technology. FCA published "Advice Unit (2016.6)". It provides regulatory feedback to firms developing automated models that seek to deliver lower cost advice to consumers. Australia's Australian Securities & Investments Commission (ASIC) has proposed "the CP (Consultation Paper) 254" and "Regulating Digital Financial Product Advice (2016.3)". This consultation paper sets out the regulation of digital financial advice in Australia.

Starting from the UK, other countries overseas regulatory sandbox that uses real verification methods have been introduced with in restricted ecosystem services for new technologies like robo-advisor and Singapore MAS (Monetary Authority of Singapore), also announced plans to introduce the system from "Fintech Regulatory Sandbox Guidelines (2016.6)". The main features of the overseas regulatory sandbox are as follows.

First, during regulatory sandbox operation, it is necessary to identify and revise the elements of amendment of laws and actual consumer participation in test and the provision of consumer protection devices is also to be carried out in parallel. The UK's FCA announced "the Regulatory Sandbox (2016.5)" and in case of unauthorized businesses, limited only to sandbox activity, approval requirements are relaxed compared to re-existing business licenses. For Authorized businesses, Individual guidelines are provided for new ideas that are difficult to apply to existing regulatory frameworks. If pre-existing regulations are violated, the applicable regulatory could be exempted and amended to the extent within the scope not violating the objectives of the sandbox program.

Second, the implementation plan and test parameters of the regulatory sandbox is implemented relatively and individually depending on circumstances of the applicant company. In the case of the UK, the applicant and the financial authority formulate a test plan together and the specific conditions for testing are agreed on a case-by-case basis, and the company have to report to the regulatory authorities once a week on progress, key outcomes, and risk management status. The Australian ASIC proposed a Regulatory Sandbox in "CP 260, Further measures to improve innovation in financial services (2016.6)". In case of 6 months testing period, 100 clients for retail clients, up to \$ 10,000 for each client, wholesale clients, more specific and restrictive testing parameters than the UK is presented such as limited number of clients/maximum \$ 5,000,000 investment per person.

Third, not only when selecting regulatory sandbox entrepreneurs, at the time of examination, Sandbox default standards also make consumer protection a key

consideration. In the UK and Singapore, when selecting the target business operators for regulatory sandbox, whether the service model of the operator is beneficial to the financial consumer and whether company has the ability to manage consumer protection or consumer risk is the major factor. Australia plans to inspect sandbox sponsors who have been approved by the Australian Securities and Investments Commission to ensure that their sponsors are not at risk of causing serious harm to consumers.

3.2 Trends on Utilization of RA Under Current Domestic Legislation

In Korea, the financial authorities are under examination regulation by the government, while granting financial companies autonomy, it is gradually spurring the transition to the level of strengthening accountability. In addition, self-regulatory institutions such as the “Korea Exchange” and “the Korea Financial Investment Association” under the 「Capital Market and Financial Investment Business Act」, unfair trade practices are regulated and conduct business to protect investors.

Domestic robo-advisor is divided into four types according to the participation of customers and consultants in the asset management process as shown in Table 2.

Table 2. Classification of services using robo-advisor.

Application information	Investor subject		
	Investors (advisory type)	Financial company (fully automated type)	Regulatory status
Use RA in back office	<Step 1> Advisors consult customers using RA’s asset allocation results	<Step 2> Managers directly manage customer assets using the program’s asset allocation results	Now permitted
RA is serviced at back office	<Step 3> RA advises customers on asset allocation results without human intervention	<Step 4> RA manages customer assets without human intervention	Allow incremental steps in the future

In accordance to Article 98 of the 「Capital Market and Financial Investment Business Act」, it limits the advisory and discretion work of those ‘Non-investment advisory manpower (or investment manpower)’ and advisors are allowed only to use the results of asset allocation done by robo-advisor for advisory and management tasks (Steps 1 and 2).

However, in accordance to 「Notice on the amendment of the Enforcement Decree of the Capital Market and Financial Investment Business Act」 (FSC Announcement No. 2016-199, ‘16.6.16.) and 「Amendment to the Financial Investment Business Regulations」 Financial Services Commission Announcement No. 2016-200, ‘16.6.27.), When using an automated computerized information processing device

(legal term: electronic investment advice device) with a certain investor protection device, investment advisory allows investment advisory business and investment account business to be conducted without advisory and investment manpower (RA steps 3 and 4). The specific requirements are: a. Electronic investment advice device conduct direct investor orientation analysis, b. The contents of investment advice should not be focused on one item or asset, c. Rebalancing investors' assets more than once every quarter, d. Prevent hacking and disasters, prevent recurrence, provide a recovery system, e. To have one professional manpower to be responsible for the operation and maintenance of the electronic investment advice device, f. RA means to go through the test bed.

The Financial Services Commission and the Financial Supervisory Service through “The basic operation plan of RA test bed (2016.8.29)”, arrange TF based on financial authorities, academia and industry. Also, to ensure that minimum discipline is working properly to conduct investment consultation and appointment, run a test bed. The RA testbed application requirements are: company with pure RA technology (unregistered investment advisory business), advisors and solicitors with RA skills, and a consortium between companies. The RA test bed presents the overall process of examination (pre/main/final examination) and the main examination items (requirements, the rationality of the algorithm, personalization, portfolio output, maintenance specialist, reasonable rebalancing, simultaneous management of multiple accounts, compliance, system security, system stability) at ‘www.ratestbed.kr’.

3.3 Major Issues Regarding Domestic RA Under the Current Legal Framework

3.3.1 Problems of Legal Status, Responsibility and Validation of AI

Due to the pace of development of new technologies and the divergence between existing legal contents, there are major issues on the use of AI technology.

First, In order to discuss the issue of responsibility and validation of AI, the legal status of AI should be considered. The current criminal law and justice system is a dual legal system of people and goods. There for in case of Weak AI, which is the current level of technology, it is similar to a person or recognized as a third status which is expected to be positively examined for legal status it possesses strong AI similar to those of human beings in the future. Darling talks about the legal right of social robots, which is similar to animal rights, that can be extended to “second-order right” [6] which make them to feel personified and indirect emotional ties. It is difficult to recognize the status of AI at the current level as a responsible entity and bear legal status and responsibility. In terms of legal liability of AI, since Weak AI is currently considered as a tool, the creator or owner will be responsible and Strong AI, as the subject of responsibility, responsibilities of AI and creator are separated. However, in consideration of the civil compensation ability or the criminal's national law, the owner or creator may be held liable. The increasing use of AI is changing the concept of traditional tort and legal liability [7]. The robo-advisor service can be broadly divided

into financial companies, robot advisor algorithm developers applying AI technology, and users. When an illegal act or damage occurs due to an error or mistake in the operation of the robo-advisor, issues on legal liability and how to prove the problem can be raised as a priority. In addition, if the robo-advisor is in the form of a consortium, the issue of distribution of responsibility among the consortium members (robo-advisor technology provider, service provider) is also unclear.

Second, it is necessary to review existing regulations on investor protection for new technology-enabled financial services. consumer protection measures are needed if robo-advisor service yields lower or algorithm error occurs that can lead to investor's damage more than formal people face-to-face style.

Third, there is uncertainty as to whether AI is included in the current copyright law use and rights. If robo-advisor directly advises and dismisses (Steps 3 and 4) are allowed in the future, in the process of big data processing when perform learning using algorithms such as deep learning, whether the use of the various works by others is included in the use of copyright law or the rights of the works may arise.

Fourth, it is necessary to discuss the issue of ethical norms on the use of AI technology. Recently, there has been ethical problems on AL algorithm such as racial bias controversy case on beauty pageant processed by AI (2016.7), Algorithm manipulation suspicion case on trending Topic, a news editing service on Facebook (Nunez, 2016), Google's Autonomous Driving Alienation Algorithm Accident (2016.5), etc. Including market manipulation problems that can arise from the market oligopoly of robo-advisor and trolley dilemma, it is also necessary to consider ethics and norms as well as laws and policies on newly emerging issues such as the obligation to prevent conflicts of interest in the robo-advisor algorithm (Investor interests first).

3.3.2 Major Issues in Financial IT Regulation

There are major issues in financial IT regulation. The current regulatory system is a positive legal form and has been regulated in detail by items such as 「Electronic Financial Transactions Act」 and 「Electronic Financial Supervisory Regulations」 which oblige domestic financial companies to use specific security technologies. However, due to recent changes in the policy, the financial security regulatory paradigm from ex-ante regulation to principle-based self-regulation. Government-led deregulation policies are being implemented to promote Fintech, but in principle they are prohibited. But in the form of a current legal system, there is a limit to activating the AI-utilizing industry such as robo-advisor.

3.3.3 Major Issues in Financial IT Security and Personal Information Protection Regulations

As a major issue in the financial IT security and privacy protection laws, first, the current AI-related laws such as 「Intelligent Robot Development and Promotion Act」 do not include information protection related contents since it focuses on industrial promotion value. “Intelligent Robot” in the 「Development and Promotion of Intelligent Robot Act」 is defined as a mechanism that autonomously operates by recognizing the external environment by itself and judging the situation. Especially, intelligent service robots have mobility and activity leading to significantly higher risks compared to other products. Therefore, it is necessary to legislate the management and

supervision system in order to prevent such dangerous idea [8]. In addition, under the Financial Regulation Act, including the 「Capital Market and Financial Investment Business Act」, to ensure the safety of services using new technologies such as robo-advisor, regulations on information security and privacy aspects need to be reviewed.

Second, in case of AI algorithm and profiling of robo-advisor, it is necessary to review whether new regulatory issues on privacy protection are reflected. Since the AI algorithm is designed and developed by human beings, differentiation or error is reflected and regulation from the design to the use of the algorithm is a major issue.

3.4 Major Issues Regarding Domestic Self-regulation Method

3.4.1 Limited Regulatory Sandbox Operation

In the regulatory sandbox of major foreign countries, the test parameters are presented as default standards, but in the case of the UK, specific conditions for testing are established on a case-by-case basis with the financial authorities. In the case of Australia, it has features that set more specific parameters than the UK, such as limiting the investment number of customers. However, in Korea, test parameters are examined only by default standards. And before the actual service, the license is given to focus on passing the examination for pre-qualification (examination). It is hard to see that the test bed is running by excluding the duties that are currently in force or under the law.

3.4.2 Strengthening the Protection of Financial Consumer Is Required

If the domestic robo-advisor passes through the testbed, through providing information on core investment strategy and operability (the yields by algorithm), it helps investors to select services and support investment decisions. However, in the test bed, the company uses its own funds or executive funds to finance it and the fact that it is run by financial consumer's funds when it enters the real market, financial consumer protection issues need to be strengthened as a key element in the review of the robo-advisor testbed.

3.4.3 Improvement of Transparency of Robo-Advisor's Algorithm Is Required

Currently, for the examination standards for “algorithmic safety” of the robo-advisor testbed, the operational goals of the algorithm and its main assumptions must be classified and the results of the algorithm test must be at a reliable level. The portfolio proposed to the investor in the robo-advisor is whether or not it operates only for the service company's product sales performance, whether the profiling work involves adjusting past quotes, and the effect of reflection on the stock price of stocks requiring the transparency of the algorithm not the interest of the investors. For the ‘rationality of algorithms’ in the main verification items of the domestic robot advisor test bed, it is necessary to further enhance transparency and error verification of international-level algorithms.

4 Challenges for the Development of Utilization of AI Technology in Financial Sector: Focused on Robo-Advisor

4.1 Basic Direction of Regulation on Utilization the AI Technology

4.1.1 Rational Restriction on Legal Status and Attribution of AI at Current Technology Level

To reduce the gap with the existing legal system for the use of new technologies, to reduce the gap with the existing legal system for the use of new technologies, it is necessary to consider the major issues and to set the direction of regulation centering on the “legal status, responsibility, and proof of AI in current AI technology utilization level”.

First, it is difficult to consider that the legal status of AI is recognized as the responsible subject in relation to the legal status of AI and ambiguity of legal liability. Robo-advisors that are still using AI that is at the automation level should distribute responsibility and proof to the service providers rather than grant legal status. If the premature responsibility is defined not taking into account the current level of technology, considering the risk of difficulties that could be evoked by cost of technology development and the deterioration of market competitiveness of services, the responsibility to the entities such as Fintech and financial companies is needed to be distributed. And the distribution of responsibility among the constituents of the consortium should be allocated desirably based on contracts and etc.

Second, since the present step is the step where consulting personnel utilize the results of asset allocation of robo-advisor for advisory and management tasks (steps 1 and 2), it is difficult for robo-advisor to be recognized as the subject of responsibility. Therefore, as a provider of investment services, companies should take security measures and take measures to protect investors taking account of the responsibility and proof of consumer protection. In addition, it will be necessary to verify whether the investment advisory services are provided for the protection of investors.

Third, since the legal status of AI is not recognized as the subject of responsibility, the basic responsibility for copyright law should be regarded as those who uses the AI. However, even if the AI is used as a tool, it is fact that the current law does not have a right if it is done without human involvement [9].

Fourth, the ethical normative issues of AI are the criteria for judging whether legal regulations such as legal status and accountability of AI are oriented toward ethical normative value. AI's technical judgments may not meet human ethical standards if there is a need for ethical judgment, such as the issue of setting responsibilities in the event of an accident.

In ‘Ethically Aligned Design’ [10], four issues and ethics guidelines are proposed and proposed and those are human rights (human rights violations before AI production), responsibility (criteria for liability when a problem occurs), transparency (maintaining transparency from AI's production process), education & awareness (Establishing a mechanism for ethical education and social awareness sharing on AI's potential misuse threats). We will look at the basic direction of regulation in the ethical normative framework, based on major issues, rather than the premature regulation of AI. And issues on ethical normative could be utilized and presented through practical recommendations and guidelines.

4.1.2 Regulation Regarding Utilization of AI Technology in Financial IT Regulations

Regulation on the use of AI technology in financial IT related regulations should be first, it is important to take full advantage of self-regulatory measures such as the use of Regulatory Sandbox. When new technologies are applied to services in combination with finance such as robo-advisor, it is necessary to find a way to make use of self-regulatory measures as a way to supplement the positive regulatory method and minimize the gaps due to deregulation.

Domestic robo-advisor has a short introduction period to provide pure robo-advisor service without human intervention and there are few functional problems and possibility of risk because of accumulated data. Therefore, the robo-advisor should be operated in coexistence of both Human Advisor (HA) and the Robo-advisor (RA). Through safety assessment, and risk management, in related laws such as the 「Capital Market and Financial Investment Business Act」, it is necessary to gradually allow the regulation of the robot advisor.

4.1.3 Regulation on Utilization of AI Technology for Financial IT Security and Privacy

Regulatory direction for the use of AI technology for financial IT security through policy approach by utilizing the guidelines that help to establish regulations that can be secured while supporting industrial development in industrial development law related to AI. In Europe, “the Robo Law Project (2012.3–2014.3)” promoted the ‘Robot Regulation Guidelines’ through policy research to cope with robot-related regulations. One of the main points is that rather than raising the general theories of regulation, the regulation direction was established through representative cases expressing the technical, ethical, social and legal implications of robots such as specific application like self-driving car, computer integrated surgical system, and care robot. In regulatory review of the use of AI technology in financial IT security regulations, through concrete use cases of financial intelligence such as robo-advisor, the process of developing guidelines is expected to be very helpful in determining the direction of regulation.

Also, in the recent EU “GDPR (General Data Protection Regulation, adopted on 2018.5.), in order to protect personal information, there has been attempts on institutionalizing the algorithm regulation, comprehensive regulations on the collection, storage and use of personal information [11]. In GDPR, the most relevant part regarding regulatory issues related to AI algorithms and privacy is, Article 21 With respect to profiling ‘Right to object’ and Article 22 ‘Automated individual decision-making, including profiling’. Algorithmic regulation with AI technology not only positively includes ‘right to explanation’ for automated personal decision making such as profiling in relation to algorithmic transparency, it is a factor that greatly enhances corporate accountability for privacy protection. Although personal information protection and algorithmic regulation are being emphasized internationally, the EU’s legislation should be closely examined whether it is applicable in domestic environment and domestic robo-advisor.

4.2 Self-regulation Through Regulatory Sandbox

4.2.1 Strengthening the Self-regulation by Expanding the Scope of Use of Regulatory Sandbox

Currently, the robo-advisor testbed focuses on pre-validation before actual service. However, it is necessary to review the way of operation to expand the application range of sandbox and to decrease the regulation burden of the target company.

Applicants and financial authorities or self-regulatory agencies after sufficient consulting, should make arrangements for establishing test plans optionally without restriction should be considered case by case. In this way, it is possible to establish an effective supervisory system by identifying and improving inadequate regulations and supervisory problems revealed in the process of testbed of innovative financial services. Which can lead to flexible structure that reflects actual laws and regulations.

4.2.2 Preparing Financial Consumer Protection Plan Over the Entire Period of the Regulatory Sandbox

Major overseas countries judge the convenience of financial consumers when selecting a project for the regulatory sandbox. And at the time of examination, consumer protection devices or risk management capabilities are considered as major factors. “Financial consumer protection” is also a very important factor in new financial services such as robo-advisor. Therefore, in order to strengthen “financial consumer protection” on the domestic robot advisor test bed as shown in Table 3, the following items should be additionally examined.

Table 3. Additional default standards for financial consumer protection in the robo-advisor test bed

Categories	Default standards
Customer benefit	<ul style="list-style-type: none"> • Possibility to provide better products to consumers as efficiency improves • Providing indirect benefits to consumers (e.g. effective competition)
Number of customers	<ul style="list-style-type: none"> • A small number of limited scale and statistical data alone should be sufficient to obtain meaningful data and decision is made given the risk management and potential customer attractiveness
Disclosure	<ul style="list-style-type: none"> • Notify the information about the content of the test and the rewards that apply to customers participating in the test under informed consent • Notice of scale, scope and accompanying risks of testing
Customer safeguards	<ul style="list-style-type: none"> • Identify potential risks to consumers and suggest ways to reduce them • In case of damage suggest an appeal and financial service compensation scheme

Additionally, after entering the robo-advisor testbed, given the fact that it operates as a retail financial consumer’s funds, it is also a factor to consider how to operate retail banking consumers by making them to participation in the robo-advisor testbed to build a portfolio for a diverse customer base and to accumulate sufficient data through actual financial consumer engagement.

4.2.3 Review of Accountability (Transparency) of Algorithm Using AI Technology

Current AI algorithms of robo-advisor are problematic in terms of classification, prioritization, and prediction which leads to participation human intervention to some extent that could bring about human errors, prejudices, manipulations, etc. that could reflect the process.

Although socio-economic utilization of algorithms is increasing rapidly, the problem considering whether algorithms are fair, neutral or objective is being constantly raised [12]. However, in the case of deep learning with a black box type hidden layer among AI algorithms under the algorithmic competition which is difficult to explain and is at the heart of robo-advisor, the algorithm has a structure that is inevitable to be closed structure as a trade secret of the enterprise. In addition, even if the demand for transparency of AI algorithms has the effect of regulation, innovative financial services will be hard to come by if new technologies that are still in developmental stages such as AI is pre-regulated. Therefore, it is necessary to examine whether the basic assumptions and criteria of the algorithm are valid from the policy design through the policy approach in the ethical normative framework and there should be a way to manage such matters in periodic verification and recording of results.

5 Conclusion

Along with the start of the fourth industrial revolution era, AI technology has been actively utilized in Korea's financial sector, and the innovative and popular asset management service called robo-advisor is now in its early stages of development. Also, thanks to the policy tone of fintech activation, there are various positive policies such as aggressive introduction of robo-advisor.

Step-by-step approval of online investment advisory and asset management services and ease of entry barriers related to the registration of consulting businesses.

However, for the robo-advisor using new technologies such as AI, it is necessary to consider on the legal status, accountability and validation of AI at the current level. Also, legal review on major issues generated due to the gap between existing laws and systems such as financial IT, security and privacy is necessary. As Lessig, Lawrence's "code is law" proposition [12], Cyberspace can be built, structured, or coded to protect the values we believe are fundamental (whether real or structural). Cyber-space code that defines freedom and regulation in cyberspace is our choice to decide whom and by which value it will be made. However, untimely regulation of technologies that did not enter the full-scale stage, such as AI, could hinder technological innovation and unclear regulations may reduce the flexibility of the phenomenon or cause market turmoil. Thus, it is necessary to examine carefully whether the current law can be regulated, whether there is a gap in regulation, and whether a new legal system is necessary.

Therefore, as a task for the development of AI technology utilization in the financial sector, this paper present a rational regulatory direction on legal status and responsibility attribution of weak AI, which is the current technology level. Considering that the robo-advisor is currently used for advisory and operational tasks, the

entities such as responsibility and validation should provide investment services and must establish a framework for security measures and investor protection measures. Also, the standards of legal regulation such as legal status and accountability of AI should aim for ethical normative value. As a way to pursue stabilization such as rational establishment of current regulation, regulatory sandbox as an autonomous regulatory measure should be considered positively. Therefore, the regulatory sandbox usage trends in major foreign countries will be expanded to suit the Korean regulatory sandbox and the necessity of establishing management plan for transparency of financial consumer protection measures and algorithms has been presented.

In this paper, it is significant that when more advanced AI technology is applied to robo-advisor in the future, in order to acquire an edge in qualitative market competition and lower the entry barriers between the countries, regulatory sandbox enhancements content has been presented for reasonable regulatory direction and self-regulation measures through a legal study on the utilization of AI technology.

References

1. A.T. Kerney: Insights from the A.T. Kearney 2015 robo-advisory services study, June 2015
2. Turing, A.M.: Computing machinery and intelligence. *MIND* **49**, 433–460
3. McCarthy, J., Minsky, M.L., Rochester, N., Shannon, C.E.: A proposal for the Dartmouth summer research project on artificial intelligence. <http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html>
4. Searle, J.R.: Minds, brains, and programs. *Behav. Brain Sci.* **3**(3), 417–457 (1980)
5. Russell, S., Norvig, P.: *Artificial Intelligence: A Modern Approach*. Prentice-Hall Inc., Upper Saddle River (1995)
6. Darling, K.: Extending legal rights to social robots, Presented at We Robot Conference, April 2012
7. Karnow, C.E.A.: The application of traditional tort theory to embodied machine intelligence, Paper Presented at The Robotics and the Law Conference. Center for Internet and Society (Stanford Law School), April 2013
8. Lee, Y.-C.: Intelligent service robot - a study on the improvement of legal and institutional system. *TTA J.* (101). Special Report
9. Kim, Y.M.: Artificial intelligence and legal issues - focusing on the legal issues of AI's output, *SPRi* issue Report, 6 June 2016
10. IEEE: Ethically aligned design version 1, 13 December 2016
11. Lee, W.T.: EU algorithm regulatory issues and policy implications. *KISDI*, December 2016
12. Lawrence, L.: *Code: And Other Laws of Cyberspace Version 2.0*. Basic Books, New York City (2006)

Author Index

- Ahmed, Irfan 137
Aminanto, Muhamad Erza 212
- Bhatt, Manish 137
Bi, Jingguo 51
Boehm, Vanesco A. J. 184
- Choi, Daeseon 297
- Firoozjaei, Mahdi Daghmehchi 61
- Grimm, Jonathan 137
- Hong, James Won-Ki 184
Hong, ManPyo 137
Hou, Jong-Uk 16, 25
Hu, Zhi 72
Hwang, Da Mi 310
- Jang, Han-Ul 16
Ji, Janghyun 3, 72
Joh, Jungwoo 150
Jung, Souhwan 162
- Kasuya, Momoka 173
Khandaker, Md. Al-Amin 283
Kim, Hoon-Kyu 150
Kim, Howon 3, 72, 283
Kim, Hyoungshick 61
Kim, Hyunki 274
Kim, In Seok 310
Kim, Jong 184
Kim, Kwangjo 212
Kwon, Hun Yeong 323
Kwon, Taekyung 150
- Lee, Dong Hoon 87
Lee, Garam 283
Lee, Heung-Kyu 16, 25
Lee, Hyungu 101
Lee, Jaehoon 274
- Lee, Keun Young 323
Lee, Kyungho 127, 224
Lee, Kyungroul 236
Lee, Minsu 101
Lee, Namsup 297
Lee, Saetbyeol 115
Lee, Sang Min 61
Li, Haoyu 39
Lim, Jong In 323
Ling, Yunhao 249
- Ma, Sha 249
Mimura, Mamoru 199
Miyamoto, Eito 261
- Nguyen, Son Duc 199
Nogami, Yasuyuki 283
- Oh, Insu 236
- Pan, Yanbin 39
Park, Jin-Seok 16
Park, Joon Young 87
Park, Mookyu 224
Park, Moosung 127, 224
Park, Taehwan 3, 72, 283
- Roussev, Vassil 137
- Sakiyama, Kazuo 173, 261
Seo, Hwajeong 3, 72, 283
Seo, Yezee 150
Shin, Ji Sun 101
Song, Hyun-Ji 25
Sugawara, Takeshi 261
- Tanaka, Hidema 199
- Wang, Mingqiang 51
Wei, Wei 51
Woo, Jae Hyeon 127

Yi, Okyeon 274

Yim, Kangbin 236

Yoon, Hyunsoo 297

Yoon, Jaehyeon 162

Yoon, Jiwon 115

Yoon, Kwon-Jin 162

Yu, In-Jae 25

Yun, Jong Pil 87