A SYSTEMS APPROACH TO CYBER SECURITY

Cryptology and Information Security Series

Volume 15

ISSN 1871-6431 (print)
ISSN 1879-8101 (online)

A Systems Approach to Cyber Security

Proceedings of the 2nd Singapore Cyber-Security R&D Conference (SG-CRC 2017)

Edited by

Abhik Roychoudhury

School of Computing, National University of Singapore

and

Yang Liu

School of Computer Science and Engineering, Nanyang Technological University

IOS Press

Amsterdam • Berlin • Washington, DC

ISBN 978-1-61499-743-6 (print)
ISBN 978-1-61499-744-3 (online)
Library of Congress Control Number: 2017930551

Publisher
IOS Press BV
Nieuwe Hemweg 6B
1013 BG Amsterdam
Netherlands
fax: +31 20 687 0019
e-mail: order@iospress.nl

For book sales in the USA and Canada:
IOS Press, Inc.
6751 Tepper Drive
Clifton, VA 20124
USA
Tel.: +1 703 830 6300
Fax: +1 703 830 2300
sales@iospress.com

The proceedings of the

2nd Singapore Cyber Security R&D Conference

February 21–22, 2017, Singapore

A Systems Approach to Cyber Security

Supported by:

National Cybersecurity R&D Programme

Participating Agencies:

NATIONAL RESEARCH FOUNDATION
PRIME MINISTER'S OFFICE
SINGAPORE

NSCS
NATIONAL SECURITY COORDINATION SECRETARIAT

MINISTRY OF HOME AFFAIRS

MINDEF SINGAPORE

CSA SINGAPORE

EDB singapore

GOVTECH SINGAPORE

This page intentionally left blank

Preface

Welcome to SG-CRC 2017, the second Singapore Cyber Security R&D Conference and welcome to Singapore! This year's theme focuses on presenting how to approach Cyber-Security in complex computing systems. Thus the focus of this year's theme is on software and systems security. In order to preserve information security of systems it is necessary to consider a wide variety of techniques to avoid cyber-attacks, or to minimize potential damage caused by a successful attacker. SG-CRC 2017 focuses on techniques and methodologies oriented to construct resilient systems against cyber-attacks that helps to construct safe execution environments, improving security of both hardware and software by means of using mathematical tools and engineering approaches for designing, verifying, and monitoring cyber-physical systems.

This year's conference contains a strong technical program. In addition to three keynotes addresses by Ruby Lee, Luke Ong and Sjouke Mauw, it also includes a selection of rigorously refereed papers presented in the regular paper sessions and short paper sessions. The Program Committee has received 21 submissions internationally and each paper was reviewed by at least 3 referees. We choose 10 papers as the result of intensive discussions held among the PC members.

This year, a two-day Cybersecurity Camp is co-located with the conference, which is organized by *Singapore Cybersecurity Consortium*. We would like to thank Dawn Song for sharing her expertise in cyber-security with students and attendees in the cyber-camp via lectures and hackathon. This year conference has also seen the formal launch of the *National Cyber-security Lab*, a national infra-structure housed at the National University of Singapore.

We would like to thank National University of Singapore and National Research Foundation as co-organizers of the conference. Special thanks go to many individuals who have contributed to the success of this conference. We thank the authors for sharing their ideas with us, the reviewers for providing valuable feedback, and all the PC members for taking time from their busy schedules to support this conference.

Thank you so much for attending SG-CRC 2017. We hope that you enjoy the program and have a great stay in Singapore!

Abhik Roychoudhury and Yang Liu (Program Co-Chairs)
Hwee Kwang Lim (General Chair)

Contents

A Systems Approach to Cyber Security
A. Roychoudhury and Y. Liu (Eds.)

doi:10.3233/978-1-61499-744-3-1

Keynote Summaries

Designing Security-Aware Architectures and Systems
Professor Ruby B. Lee, Princeton University

The frequency and severity of cyber attacks escalate while our dependence on cyber space increases. However, the computers that power cyber space have not been designed with security threats in mind. Can security awareness be built into the basic design of future computing devices and servers? How can it be added to existing computer systems? Can new hardware security architectures strengthen software security solutions? In this age of cloud computing, how can a cloud customer be assured of the security health of the Virtual Machines that he leases in the cloud? How can smartphones or IoT systems be designed to help thwart attacks? How can we build security in, without sacrificing performance or usability? How can we verify the security of the security architectures we design? We examine some of these issues and provide some concrete examples. Indeed, designing computer architectures and systems that are security-aware is not only critically important, but also one of the most exciting challenges today.

Verification of Security Protocols and the Secrecy Problem
Professor Luke Ong, University of Oxford

Security protocols are distributed programs that are designed to achieve secure communications using cryptography. They are extensively deployed today to enhance the security of applications, ranging from electronic payments and internet banking to e-voting. In view of the long history of design flaws in such protocols, and the immense financial and societal costs in case of failure, formal verification is a necessity. In contrast to other safety critical systems, a key feature of the security properties of protocols is that they must hold in the presence of an arbitrary adversary or intruder, and this makes them challenging to verify. An important example of such a security property is Secrecy: to verify that a protocol satisfies Secrecy amounts to checking whether it can leak a secret to the environment as a result of interference by the intruder. The aim of this talk is to provide a survey of modern approaches for the formal modelling of protocols and automatic methods of verification. Another aim is to present recent advances in the automatic verification of the Secrecy Problem. Thanks to Durgin et al., it is widely known that the secrecy of security protocol remains undecidable even when restricted. We show, perhaps surprisingly, that Secrecy is decidable for a non-trivial class of protocols that are depth-bounded (a restriction first introduced by Meyer for the Pi-Calculus); we discuss examples of depth-bounded protocols, and type-theoretic methods for proving satisfaction of depth-boundedness.

Let's Stick Together: Digitally Ensuring Physical Proximity
Professor Sjouke Mauw, University of Luxembourg

What do traditional keys, train tickets and coins have in common? They are increasingly being replaced by digital solutions, such as electronic car keys, smart tickets and contactless payment systems. Unfortunately, various physical properties that are easily verified in the traditional setting are significantly harder to achieve in the digital world. An example is the proximity of a lock and its key. In the physical world, in order to open a lock, the key needs to be physically inserted into it. However, proximity is much harder to ensure using digital means only. And indeed, relay attacks on electronic car keys have been found. Over the past few decades, researchers have been studying security protocols that digitally ensure such physical properties. In this presentation I will discuss one particular class of protocols, namely distance-bounding protocols. I will show that a large number of proposed protocols can be formalized in a uniform model. The advantages of such a model include the ability to perform a generic security analysis, to study the trade-off between memory and security and to search for optimal distance-bounding protocols.

A Systems Approach to Cyber Security
A. Roychoudhury and Y. Liu (Eds.)
© 2017 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/978-1-61499-744-3-3

Chat-App Decryption Key Extraction Through Information Flow Analysis

Zhongmin DAI, SUFATRIO, Tong-Wei CHUA, Dinesh Kumar BALAKRISHNAN, Vrizlynn L. L. THING
Institute for Infocomm Research, Singapore
Email: {daiz, sufatrio, twchua, dineshb, vriz}@i2r.a-star.edu.sg

Abstract. Recent years have seen a pervasive usage of mobile-based instant messaging apps, which are popularly known as chat apps. On users' mobile devices, chat logs are usually stored encrypted. This paper is concerned with discovering the decryption key of chat-log database files as they are used by popular chat apps like WhatsApp and WeChat. We propose a systematic and generalized information-flow based approach to recovering the decryption key by taking advantage of both static and dynamic analyses. We show that, despite the employed code obfuscation techniques, we can perform the key discovery process on relevant code portions. Furthermore, to the best of our knowledge, we are the first to detail the employed string de-obfuscation, encrypted database file structure, and decryption-key formulation of the latest WhatsApp with crypt12 database. We also demonstrate how our key-extraction techniques can decrypt encrypted WhatsApp and WeChat database files that originate from a target device. Additionally, we show how we can construct a version of WhatsApp or WeChat that simulates the key generation processes of a remote target device, and recover the keys. Lastly, we analyze why our technique can work on widely-popular chat apps, and mention measures that can be adopted by chat-app developers to better protect the privacy of billions of their users.

Keywords. Mobile security, privacy protection, Android, mobile apps, chat apps

1. Introduction

The unprecedented proliferation of mobile devices in recent years has led to a pervasive usage of mobile-based instant messaging applications, which are also popularly known as *mobile chat apps*. Such apps allow mobile users to instantly exchange text messages and media files to each other on either an 1-to-1 or user-group basis. On user devices, chat logs are stored in database files, and are usually encrypted. Decryption of the chat-log files thus represents a very serious threat to the privacy of mobile users.

This paper is concerned with discovering the decryption key (password) of chat-log database files as they are used by two widely-popular chat apps, namely WhatsApp [1] and WeChat [2]. It proposes a systematic and generalized approach to discovering the decryption key of both apps by inspecting the flow of sensitive key-related information, particularly towards the employed cryptographic libraries. Using the two popular apps as real-world examples, we show that, despite the employed code obfuscation techniques, we can still discover the actual decryption keys in use and gain understanding of the key construction process. Furthermore, to the best of our knowledge, we are the first to detail the employed string de-obfuscation, encrypted database file structure, and decryption-key formulation of WhatsApp app that uses the latest (as of October 2016) `crypt12` database file.

To realize our information-flow analysis, we make use of a combination of both dynamic and static analyses. In our work, which is implemented on Android, we employ both the Dynamic Dalvik Instrumentation (DDI) [3,4] and Android Dynamic Binary Instrumentation (ADBI) [5,6] dynamic analysis toolkits in order to observe database-file access operations and to recover the decryption keys in use. We also utilize static analysis tools, including `Apktool` [7], jadx Dalvik-to-Java decompiler [8], and taint trackers [9,10], to assist us in understanding the code structure, component interactions, and decryption-key information flow within the target chat apps.

While there exist prior articles that explain how decryption keys of some specific older versions of WhatsApp and WeChat are derived [11,12], our work takes a generalized approach to observing key-related information flow of the chat apps. Hence, while existing scripts target only very specific older or current versions of chat apps, our proposed technique can apply to different chat-app versions as long as they retain their same design and practice of employing cryptographic libraries and passing the key information into the libraries.

Our experiments done on a rooted device show how our analysis of WhatsApp and WeChat allows us to discover the decryption keys and all cipher-operation parameters. Subsequently, we can extend the key-extraction techniques by showing how we can construct a WhatsApp and WeChat app version that can simulate other devices' key generation process. As a result, we will be able to decrypt encrypted WhatsApp and WeChat log database files that originate from a remote target device, thus discovering the target mobile user's chat activity information.

Given the huge user base of both analyzed chat apps, our decryption-key discovery results are therefore very important. Hence, we also provide an in-depth analysis of why our proposed technique can still discover the keys of highly-popular chat apps. Based on our root-cause analysis results, we propose concrete counter measures that can be adopted by chat-app developers to improve their future app versions and better protect the privacy of billions of their users.

In summary, our work in this paper makes the following contributions:

- We propose a systematic and generalized approach to discovering the decryption key of chat apps by inspecting the flow of key-related information particularly towards their employed third-party cryptographic libraries.
- Using a set of experiments on a rooted device, we demonstrate how we obtain the decryption key and cipher-operation parameters of both WhatsApp and WeChat.
- To the best of our knowledge, we are the first to detail the employed string de-obfuscation, encrypted database file structure, and decryption-key formulation of the latest crypt12 WhatsApp database file.
- We elaborate how we can derive a chat-app version that can simulate other devices' key generation process in order to decrypt encrypted log files that originate from a remote target device.
- Lastly, we provide an in-depth analysis on why our technique can work on highly-popular apps, and suggests counter measures that can be adopted by chat-app developers to prevent potential attacks.

The remainder of this paper is organized as follows. Section 2 gives some background and mentions related work. Section 3 elaborates our proposed chat-app analysis approach. Section 4 reports our experiments on WhatsApp and WeChat. Section 5 explains how we can derive a chat-app version that simulates other devices' key generation process. Section 6 suggests how chat-app developers can improve their future apps, and also discusses ethical considerations of our work. Finally, Section 7 concludes this paper.

2. Background and Related Work

2.1. Background

We give some background on WeChat and WhatsApp apps that we analyzed. We also briefly describe the dynamic and static analysis tools employed in analyzing the apps.

2.1.1. Popular Chat Apps and Their Database Files

WeChat [2] from Tencent is among the most popular chat apps. It is originally known as Weixin, and has a huge user base particularly in Asia [13]. The latest company report mentions that the combined monthly active users (MAU) of WeChat and Weixin in 2015 reached 697 million, which is an increase of 39% from the previous year.

In Android, WeChat private data is stored under the `/data/data/com.tencent.mm` folder. Two sub-folders of interest are the `MicroMsg` and the `shared_prefs`. Other than several configuration files, the `MicroMsg` folder contains a folder that has a name resembling a random alphanumeric string, e.g. `b98a831a089cef3031385d56a0f51927`. This folder contains two encrypted database files, namely `enFavourite.db` and `EnMircroMsg.db`. They are both encrypted using SQLCipher [14], which is an open-source encryption extension to SQLite. SQLCipher provides a transparent database encryption interface to SQLite by wrapping SQLite method invocations so that encryption and decryption can be carried out before the data is written into the database or after it is read from the database, respectively. The `shared_prefs` folder, meanwhile, contains `.xml` files that store key-value pairs of WeChat settings.

WhatsApp [1], which is now owned by Facebook Inc., is currently the most popular chat app [15]. In February 2016, WhatsApp reported that it had 1 billion users [16]. For Android devices, WhatsApp stores its log within two unencrypted SQLite database files `msgstore.db` and `wa.db` under the protected `/data/data/com.whatsapp/databases` folder. Additionally, WhatsApp keeps an encrypted backup of the `msgstore.db`, which is wrapped by a header and trailer containing several pieces of account information, on the phone's SD card. This file is stored with the `.crypt`⟨*version_no*⟩ filename extension, with `crypt12` being the most recent one as of October 2016. The file is utilized when the mobile user transfers his/her chat log over to a new phone while retaining the existing WhatsApp phone number [17].

Similar to WeChat, there exist articles in the literature that explain how to obtain the key of the encrypted database file of specific WhatsApp version. For instance, [11] derives the key for the now-defunct WhatsApp `crypt7` database file. Our work reported in this paper, instead, takes a generalized approach to discovering the decryption key based on an information-flow analysis of the targeted chat apps.

2.1.2. Dynamic Analysis using DDI/ADBI Toolkits

We perform our dynamic analysis of chat apps on Android by using the Dynamic Dalvik Instrumentation (DDI) and Android Dynamic Binary Instrumentation (ADBI) toolkits. The DDI toolkit [3,4] is employed to monitor the Dalvik code component of a target Android app. It performs an in-line hooking technique of Dalvik-method entry points, and diverts the invocation of a target Dalvik method into a correspondingly added JNI-based native method. The added native method will ultimately invoke the original Dalvik method, thus enabling itself as a code-instrumentation point for the original Dalvik method. The DDI toolkit also supports the loading of additional Dalvik classes into a process, thus allowing the instrumentation code to be partially written in Java.

The DDI toolkit operates on top of the ADBI toolkit [6,5], which implements the hijacking utility of the ARM binary code. Due to its binary-level hooking feature, the ADBI can hook the native code of an Android app. In our chat-app dynamic analysis, we simply print out the parameters passed into the monitored methods, which can then be viewed using the `adb logcat` command.

Using the DDI/ADBI toolkits, performing a dynamic monitoring and instrumentation of Android apps does not require any modification of the target apps, thus keeping their signature intact. As such, the conducted analysis avoids any possible issues with apps that perform self-integrity checks. This approach is thus more robust than app-rewriting based monitoring approaches that perform Dalvik code injection. Furthermore, the ability of monitoring the native code component of target apps represents a strong feature, which is lacking in many other Android dynamic analysis tools. The DDI/ADBI toolkits are previously utilized by Mulliner et al. [18] to modify the behavior of target apps related to their in-app billing transactions with Google Play. One downside of the DDI/ADBI toolkits is that they require a rooted device to run the analysis.

2.1.3. Static Analysis of Chat Apps

To inspect our target chat apps and their information flows, we utilize a few static analysis tools, including `Apktool` [7], jadx decompiler [8], and IDA Pro [19]. The `Apktool`, which incorporates a Dalvik disassembler called `baksmali`, is employed to extract a chat app's APK file and generate the smali code representation of the app's Dalvik code. The jadx Dalvik-to-Java decompiler is alternatively used to recover the Java source code of an app. One benefit of obtaining smali code, which corresponds to the assembly of a Dalvik bytecode, is that the code can be modified and reassembled to produce a modified app. In contrast, a decompiler usually cannot fully recover an app's Java source code, which could be due to missing high-level (e.g. type) information or any applied preventive obfuscation transformations [20]. The recovered Java source code is, nonetheless, easier to inspect than smali code due to the shown high-level programming constructs. We utilize the IDA Pro to analyze the native code component of the target apps.

To specifically inspect dataflow connection between two operations within our target apps, we use FlowDroid [9] and DroidSafe [10]. The two tools can help identify data flow from a specified source and sink method of a target app. We can thus utilize the tools to inspect potential data flow between two operations of interest. Analyzing large complex apps like WhatsApp and WeChat, however, take the two tools a long time to complete and require a machine with an ample amount of RAM. In our experiments, we managed to get some analysis results from FlowDroid after running it with several optimization flags [21]. It is known, however, that FlowDroid may induce false negatives as well as false positives [9,22]. We employed FlowDroid in our our experiments to help point out any potential connections between operations of interest, which then provided inputs to our manual analysis of the recovered code.

2.2. Related Work

There exist articles in the literature that describe how we can derive the decryption key of specific WhatsApp and WeChat versions. The work [11] lists the steps to obtain the key for WhatsApp's now-defunct `crypt7` database file. Meanwhile, [12] describes how WeChat currently forms its key by taking the first seven characters of the MD5 hash value of the WeChat user ID and the phone IMEI number. Instead of targeting specific chat-app versions, our work in this paper takes a systematic and generalized information-flow based approach to recovering the key by applying both static and dynamic analyses. Hence, unlike [12,11], our proposed approach can apply to future chat-app versions provided that they still retain their existing basic design and practice of employing third-party cryptographic libraries and passing the key information into the libraries.

There are a number of existing works that dynamically analyze Android apps to inspect the behavior of the apps. AppTrace [23] uses dynamic analysis of Android apps to identify any malware execution. The work, however, does not aim to find the decryption key of target app's database files, which is the goal of our work. ConDroid [24] considers the problem of locating critical, interesting or dangerous code, and enforcing its execution for observability. It combines a static analysis with concolic execution in order to observe an execution path that leads to a code section target. While the work enables the potential observability of a target app's code sections, it does not specifically aim to discover the target app's operations that hold or deal with decryption key of chat apps. Moreover, the technique applies only to Android app bytecode, and cannot deal with any included native code.

3. Decryption-Key Discovery using Information Flow Analysis and Execution Monitoring

We now explain in this section the attack models that we assume on both target chat apps, and the approach that we take in our decryption-key discovery process.

3.1. Attack Models

While both WeChat and WhatsApp use encryption to protect their log database files, the encryption is used for different purposes and is executed differently. Hence, we consider different attack models for the two chat apps as follows.

3.1.1. WeChat Attack Model

WeChat encrypts its database files using SQLCipher, and stores them under the protected `/data/data/com.tencent.mm` folder. This encryption measure thus represents an extra layer of protection since, under normal circumstances on unrooted devices, all files under the folder are accessible only to WeChat app. This measure is therefore very useful given the fact that the many Android users intentionally root their devices.

For WeChat, we thus consider the following two attack scenarios:

- (WC_1) *A locally-acquired, unlocked rootable device scenario*: Here, we assume that we have physically acquired an unlockable device with a WeChat app installed. Also, we assume that the device is *rootable*, either because it is already rooted by its owner, or is vulnerable to a root exploit. As a result, we can access the acquired device, including all WeChat related files, and perform a dynamic instrumentation and monitoring to be elaborated later in Section 4.1.
- (WC_2) *A remote, trojanized and rootable device scenario*: This scenario assumes a remote target device with an Internet connection that, along with a WeChat installed, has our trojan app running. Similar to the scenario WC_1, we also assume that the device is rootable. This allows the trojan app to transfer via the Internet the following information after its execution privilege elevation: WeChat database files, shared preference file, and device information (e.g. IMEI number). For this scenario, we need to first find out what methods to monitor as explained in Section 4.1, and subsequently run a password-generating app as shown in Section 5.

As can be observed, the two considered scenarios above do assume a rootable target device. One may thus argue that, given the prerequisite root privilege, other attack techniques can alternatively be employed to reveal the targeted decryption key. In our attack technique, the requirement for a rootable target device is put to solely extract WeChat protected files on the target device. Using this information, our attack can then run independently on our own device, and does not require any further interactions with the target device, which may be intrusive and could be observable by the mobile user.[1]

3.1.2. WhatsApp Attack Model

Under its protected `/data/data/com.whatsapp/databases` folder, WhatsApp stores its database files as well as a key file `/data/data/com.whatsapp/files/key` in clear. As mentioned earlier, WhatsApp also applies encryption to generate a backup database file, which is normally stored on the device's SD card, for a chat-log transfer purpose. Our goal is thus to decrypt the encrypted backup database file without having access to any of WhatsApp protected files.

We consider a WhatsApp attack model that enables us to obtain the following pieces of required information:

- Encrypted WhatsApp backup database file stored on the SD card.
- Google account name that is used by mobile user, which can be obtained by invoking the Android API call `android.accounts.AccountManager.getAccountsByType("com.google")` [25].

Three following attack scenarios are therefore possible:

- (WA_1) *A locally-acquired, unlockable device scenario*: where we acquire an unlockable device, with a WhatsApp app installed and its SD card accessible. Since the device is unlockable, we can thus install a simple (helper) app that reveals the required Google account name.

[1]Notice that, for WhatsApp, our attack does not require a rootable target device as explained in Section 3.1.2.

- (WA_2) *An acquired SD card with a guessable Google account-name scenario*: where we acquire the SD card containing an encrypted WhatsApp database file, and we can guess the used Google account name.
- (WA_3) *A remote, trojanized device scenario*: where we target an Internet-connected remote device installed with both WhatsApp and our trojan app. This scenario allows the trojan app to send over both the required database file and Google account name.

Unlike the two WeChat scenarios WC_1 and WC_2, all the three WhatsApp scenarios WA_1–WA_3 above do not assume a rootable target device. To achieve our objective of decrypting the database file under these three scenarios, we need to first perform the key-discovery technique on our rooted device as elaborated in Section 4.2, and subsequently run a password-generating app as expounded in Section 5.

3.2. Chat-App Execution Monitoring and Key Inspection

By using the DDI/ADBI toolkits, we are able to monitor the passed arguments to methods that perform database decryption operations, both at the the Dalvik and native code levels. The DDI/ADBI toolkits additionally allow us to inspect the operations at the `libc` level. Hence, we are also able to see all `libc`'s database-file related operations, together with the name of the files. For our experiments, we customized the ADBI toolkit to provide better memory-map support. We increased the size of the `char raw[]` array, as suggested in [26], from the default 80,000 to 800,000.

The main challenge faced in dynamically analyzing the target chat apps is determining the pertinent app methods that receive the decryption key strings as their arguments. There are a high number of methods in large Android apps like WhatsApp and WeChat. Hence, observing all methods is simply impractical. We thus need to shortlist the target methods to monitor by applying static analysis on the chat apps.

Another problem in dynamically analyzing the target apps is ensuring that decryption-key related operations are indeed executed by the apps. Fortunately, WeChat opens the decrypted files right after the user logs on. WhatsApp uses the key when it reads the encrypted duplicate database file as the user moves his/her chat log over to a new phone. Additionally, the key is used when the user initiates a WhatsApp menu of performing a chat-log backup, which we monitored in our experimentation.

3.3. Static Analysis of Key-Related Information Flow

We have discussed the challenge of determining the app methods that receive the decryption key strings as their arguments earlier. To shortlist the target methods to monitor, we perform the following static analyses of the chat apps at both the Dalvik and native code levels, together with several applied heuristics.

First, we generate a list of all native methods that are invoked by the Dalvik code of a target chat app. For this, we use the `Apktool` to recover the smali code representation of the app's Dalvik code. In smali code, a native method is declared with `.method` followed by method access identifiers, such as `public`/`private` and `static`, and the identifier `native`. An example is the declaration of `.method private static native nativeResetCancel(IZ)V`. We can thus list all invoked native methods by searching a regular expression of "`method .* native`". To further shortlist our target methods, we consider methods that accept strings or array of bytes in their parameters.

Second, we analyze the information flow within the app's native code by tracking the method interaction using the IDA Pro. Starting from the native methods shortlisted in the previous step, we employ the IDA Pro to observe the interaction among the native methods. Again, we can opt to consider only methods that accept strings or array of bytes in their parameters.

Third, we can also check if the native code methods that perform cryptographic operations are actually adopted or customized from a third-party library. Information of the library will thus be very helpful in understanding the cryptographic operations in use, especially if the library is an open-source software. While a customization is performed and code obfuscation may be applied, the core data structures and cryptographic methods typically do not differ much.

Lastly, we apply static analysis of the Dalvik code by using the static taint tracking analysis tools, such as FlowDroid. In this way, we can observe dataflow relationship between two operations of interest. The recovered Java code derived by a Dalvik-to-Java decompiler also provides useful information on how the Dalvik code component of a target app works.

4. Experimental Results

We applied the dynamic and static analysis techniques elaborated in Section 3 to WeChat and WhatsApp, and report the experimental results on a rooted device in this section.

4.1. WeChat Key Recovery

We analyzed WeChat version 6.3.22, whose APK was built on July 11, 2016. To trigger the WeChat's decryption process, we need to first log out any existing user session, and then log in again.

WeChat customizes the SQLCipher code base [27] instead of simply using it. When inspecting the smali code of WeChat's Dalvik code, we could not find any reference to the standard `net.sqlcipher.database` SQLCipher package [28]. Instead, we found out a number of SQLite-related classes under the `com.tencent.kingkong` package. One such class is `SQLiteConnection`, which is related to database file opening. We thus hooked and monitored the `SQLiteConnection.<init>(...)` method, and extract its passed string parameters. In this way, we were able to retrieve the decryption key.

Furthermore, the path of the accessed database file can also be extracted from the `SQLiteDatabaseConfiguration` object that is passed as an argument to the hooked method above. The accessed WeChat database files, which are identified as parameter `path` below, and their respective decryption key, which are identified as parameter `mPassword`, are recovered as follows:

```
SQLiteConnection.<init>:: path=/data/data/com.tencent.mm/MicroMsg/
   b98a831a089cef3031385d56a0f51927/EnMicroMsg.db
SQLiteConnection.<init>:: mPassword=fe34ed7
SQLiteConnection.<init>:: path=/data/data/com.tencent.mm/MicroMsg/
   b98a831a089cef3031385d56a0f51927/enFavorite.db
SQLiteConnection.<init>:: mPassword=fe34ed7
SQLiteConnection.<init>:: path=/data/data/com.tencent.mm/MicroMsg/
   ee1da3ae2100e09165c2e52382cfe79f/EnResDown.db
SQLiteConnection.<init>:: mPassword=7a6cc74
SQLiteConnection.<init>:: path=/data/data/com.tencent.mm/MicroMsg/
   b98a831a089cef3031385d56a0f51927/SnsMicroMsg.db
SQLiteConnection.<init>:: mPassword=<NULL>
SQLiteConnection.<init>:: path=/data/data/com.tencent.mm/MicroMsg/
   b98a831a089cef3031385d56a0f51927/SnsMicroMsg.db
SQLiteConnection.<init>:: mPassword=<NULL>
SQLiteConnection.<init>:: path=/data/data/com.tencent.mm/MicroMsg/
   b98a831a089cef3031385d56a0f51927/IndexMicroMsg.db
SQLiteConnection.<init>:: mPassword=<NULL>
SQLiteConnection.<init>:: path=/data/data/com.tencent.mm/MicroMsg/
   b98a831a089cef3031385d56a0f51927/CommonOneMicroMsg.db
SQLiteConnection.<init>:: mPassword=<NULL>
```

We additionally hooked a few native methods of WeChat to obtain several cipher parameters used by WeChat [29]. While it is possible to hook the relevant Java methods in order to obtain the cipher parameters, we found that we can obtain the parameters more easily by hooking a native SQLCipher method that is adopted by WeChat. For this purpose, we took advantage of the SQLCipher code base [27]. Although WeChat customizes SQLCipher, it seems to apply little modification on the core SQLCipher data structure and operations. We thus hooked the `libkkdb.sqlcipher_codec_ctx_set_pass(...)` method, and inspected its accessed database filename, decryption key, and `codex_ctx` object. The data structure of the `codex_ctx` object can be discovered by analyzing the relevant SQLCipher source file of `crypto_impl.c`.

We were able to retrieve all the parameters of SQLCipher database file decryption by inspecting the following recovered parameter values on our monitoring device:

```
sqlcipher_codec_ctx_set_cipher :: cipher_name=aes-256-cbc, return code=0
sqlcipher_codec_ctx_set_kdf_iter :: kdf_iter=4000, return code=0
sqlcipher_codec_ctx_set_pass :: key=fe34ed7, return code=0
sqlcipher_codec_ctx_set_pass :: codec_ctx->page_sz=1024
sqlcipher_codec_ctx_set_pass :: filename=/data/data/com.tencent.mm/
   MicroMsg/b98a831a089cef3031385d56a0f51927/EnMicroMsg.db
sqlcipher_codec_ctx_set_use_hmac :: use=0, return code=0

sqlcipher_codec_ctx_set_cipher :: cipher_name=aes-256-cbc, return code=0
sqlcipher_codec_ctx_set_kdf_iter :: kdf_iter=4000, return code=0
sqlcipher_codec_ctx_set_pass :: key=fe34ed7, return code=0
sqlcipher_codec_ctx_set_pass :: codec_ctx->page_sz=1024
sqlcipher_codec_ctx_set_pass :: filename=/data/data/com.tencent.mm/
   MicroMsg/b98a831a089cef3031385d56a0f51927/enFavorite.db
sqlcipher_codec_ctx_set_use_hmac :: use=0, return code=0

sqlcipher_codec_ctx_set_cipher :: cipher_name=aes-256-cbc, return code=0
sqlcipher_codec_ctx_set_kdf_iter :: kdf_iter=4000, return code=0
sqlcipher_codec_ctx_set_pass :: key=7a6cc74, return code=0
sqlcipher_codec_ctx_set_pass :: codec_ctx->page_sz=1024
sqlcipher_codec_ctx_set_pass :: filename=/data/data/com.tencent.mm/
   MicroMsg/ee1da3ae2100e09165c2e52382cfe79f/EnResDown.db
sqlcipher_codec_ctx_set_use_hmac :: use=0, return code=0
```

From the shown results above, we can conclude that WeChat performs its cipher operations using "aes-256-cbc" with a page size of 1,024 and the KDF iteration value [30] of 4,000. No HMAC construction is used for the operations. We can also see that the recovered keys confirm those that were obtained using the Dalvik-based method monitoring. We have validated the correctness of the recovered keys and parameter values by running Linux's `sqlcipher` command to open the encrypted database files and produce the clear database files.

4.2. WhatsApp Key Recovery and Database File Analysis

We analyzed WhatsApp version 2.16.133, whose APK was built on June 18, 2016. Its encrypted database uses `.crypt12` extension, which is still being used as of October 2016. While discussions and decryption scripts for older `crypt7` and `crypt8` files are publicly available, there are still no available details of how the more recent crypt files can be decrypted. Similar to the decryption of the older WhatsApp database versions [11], our goal is to discover both the key and the initialization vector (IV) of the decryption operation.

In our analysis, we found out that string literals of WhatsApp code are obfuscated. In smali code representation of WhatsApp Dalvik bytecode, a string literal declaration operation looks like in the following (see also [31] for Dalvik bytecode's instruction set):

```
const-string/jumbo v0, "i|6C\r|f0C[i|6\u001e\u001dc2".
```

We thus first performed string de-obfuscation on WhatsApp bytecode classes in order to help us locate target methods related to cryptographic operations more effectively. Based on our analysis of the Java source code recovered, we discovered that each string within a class is obfuscated simply by XOR-ing it with a repeated 5-byte obfuscation string that is generated for the class.

Using our string de-obfuscation script, we could pinpoint the `com.whatsapp.util.b` class, which seems to perform encryption/decryption operations. The class contains a number of private instance fields of `javax.crypto.Cipher` class type [32]. Using dynamic analysis, we discovered all decryption parameters by monitoring "`static Cipher getInstance(String)`" and "`void init(int, Key, AlgorithmParameterSpec)`" methods [32]. The former reveals the employed "AES/GCM/NoPadding" cipher transformation, while the latter shows the key and IV used. The values recovered on our device are as follows, with the last 30 characters of the key shown as '*':

```
file open :: path=/data/data/com.whatsapp/databases/msgstore.db, flags=0,
   mode=0, fd=57
file open :: path=/data/data/com.whatsapp/files/key, flags=0, mode=0, fd=92
Ljavax/crypto/Cipher :: mode=1
Ljavax/crypto/Cipher :: transformation=AES/GCM/NoPadding
size of byte array is 32
Ljavax/crypto/Cipher :: key=A0D0EB3BE2A06495CD9FEDBC93EFDCEBF0************
  ******************
size of byte array is 16
Ljavax/crypto/Cipher :: IV=E6BAC22AC9712FE5A7292E852A5C3092
```

Unlike WeChat, WhatsApp manages its encrypted database file without relying on any third-party libraries, such as SQLCipher. Since its encrypted database file is meant for user chat-log transfer, WhatsApp thus also embeds additional sensitive information into the file in order to facilitate its decryption and subsequent transfer. Understanding the structure and content of the database file is therefore important.

Our analysis on the `.crypt12` database file found that WhatsApp puts a 67-byte header before and 20-byte trailer after the database ciphertext (more information on this header and trailer below). We have successfully validated the obtained key and IV by running our decryption code on the header- and trailer-stripped database ciphertext of `.crypt12` file. The following code snippet shows how our decryption code employs the recovered cipher parameters:

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding")
SecretKeySpec keySpec = new SecretKeySpec(key, "AES")
GCMParameterSpec ivSpec = new GCMParameterSpec(128, IV)
cipher.init(Cipher.DECRYPT_MODE, keySpec, ivSpec)
compressedPlainBytes = cipher.doFinal(cipherBytes)

// Decompress the obtained compressedPlainBytes
decompresser = new Inflater(false)
decompresser.setInput(compressedPlainBytes)
clearDB = new FileOutputStream(clearDBFilename)
buffer = new byte[1024]
while(!decompresser.finished()){
    count = decompresser.inflate(buffer)
    clearDB.write(buffer, 0, count)
}
```

Lastly, we would like to understand the content of the encrypted database header and trailer, and how the decryption key is formed based on the contained information during a log transfer. WhatsApp does not store the decryption key within the header of its `crypt12` file. Instead, it needs to contact a WhatsApp server and retrieve the decryption key by supplying several pieces of information extracted from the header and trailer.

Our conducted analysis found out the following pieces of information embedded within the `crypt12`'s 67-byte header:

- *cipher-header preamble* (2 bytes): `0x0`, `0x1`;
- *key version* (1 byte): `0x2`;
- *server salt* (32 bytes): the stored pseudo-random salt from WhatsApp server;
- *Google account-name salt* (16 bytes): the salt used to produce *Google account-name hash*, which is to be sent to WhatsApp server (more on this below);
- *IV* (16 bytes): the initialization vector value.

Likewise, we found the following pieces of information within the 20-byte trailer, which are used by WhatsApp app solely to ensure the integrity of the database file:

- *MD5 hash value* (16 bytes): the MD5 hash value of the encrypted database file;
- *Phone no's suffix* (4 bytes): which is derived from the last few digits of the device's phone number.

When a mobile user transfers his/her encrypted WhatsApp chat log into a new device, the decryption key needs to be retrieved from WhatsApp server. WhatsApp client app on the new device sends both the *server salt* and the *Google account-name hash* to WhatsApp server. The former is extracted from the database file header. The latter is derived by applying the SHA-256 hash function on the mobile user's *Google account name*, which is obtainable by invoking the Android API `android.accounts.AccountManager.getAccountsByType("com.google")` [25], and the *Google account-name salt* from the file header.

Once WhatsApp app receives the decryption key at the log transfer process, it will then store the key, and other necessary information, within a file named `/data/data/com.whatsapp/files/key`. In the WhatsApp version that we analyzed, this `key` file is 158-byte long, and contiguously stores *Java serialization data header* (27 bytes), *cipher-header preamble* (2 bytes), *key version* (1 byte), *server salt* (32 bytes), *Google account-name salt* (16 bytes), *Google account-name hash* (32 bytes), *IV* (16 bytes), and the retrieved decryption key (32 bytes). Subsequent cryptographic operations on the encrypted database file simply utilize the locally stored key.

5. Key Generation for Remote Target Devices

WeChat and WhatsApp decryption keys are formed based on a few pieces of device-specific information, such as the phone IMEI number, and/or values that are locally stored on the device [12,11]. When we consider attack scenarios involving a non locally-accessible target device, such as scenarios WC_2, WA_2 and WA_3, we thus need to simulate the target device's key generation process. Knowing what methods to recover the key as expounded in Section 4, unfortunately, is still insufficient. We additionally need to modify the chat-app behavior on our monitoring device by utilizing the information sent by the planted trojan app or guessed information. This can be done by using the following two techniques.

The first technique rewrites the target chat app. It replaces all Dalvik-level Android API invocations that obtain the device-specific information with direct value assignment operations. `Apktool` [7] can be used to generate the text-based smali code representation of the chat apps, and to subsequently derive the APK file of the altered chat app.

The second technique utilizes the dynamic instrumentation feature of the DDI/ADBI toolkits. It has a benefit over the first one in that it can intercept and instrument both Java and native methods. This feature was previously employed by [18]. The work shows how the behavior of target apps can be modified so that, instead of performing an in-app billing transaction with Google Play, the instrumented code intercepts the transaction and sends back a forged in-app payment confirmation.

In addition to app behavior modification, we need to copy all relevant local files of the target device, which are assumed to be transferred by our trojan app, into our monitoring device. Our monitoring techniques as explained in Section 4 will then be able to reveal the decryption keys of WeChat and WhatsApp running on the target device.

5.1. WeChat Behavior Modification

A successful WeChat password generation of a remote device can be achieved by performing the following two steps, which are previously also pointed out by [12]:

- Modify WeChat behavior by replacing the value returned by the Android API call `android.telephony.TelephonyManager.getDeviceId()` with the IMEI number of the target device.

- Copy the shared preference files within the folder `/data/data/com.whatsapp/shared_prefs/` with those from the target device.

Using this technique, we can then reveal the targeted decryption key. The technique, in fact, will continue to work despite possible changes of the employed hash function (i.e. MD5) in future WeChat versions, as long as the key-derivation process depends upon only the IMEI number and locally stored files. This represents an advantage of our employed dynamic analysis over the static analysis used by prior work [12].

5.2. WhatsApp Behavior Modification

For WhatsApp password generation of a non locally-acquired device, we need to perform the following two steps:

- Copy the target device's database file to the monitoring device's SD card.
- Modify WhatsApp to return the *Google account name* of mobile user in the target device. In this way, the altered app will retrieve the key from WhatsApp server using the *Google account-name hash* of the target device as explained earlier.

Notice that the key request issued by the monitoring device to WhatsApp server will succeed if the server does not store and verify the association between the used Google account name and phone number of a mobile device. In our experiments, we decided not to probe WhatsApp server and ascertain this provisioning. Instead, we opt to forewarn the security community of this potential security weakness pertinent to a chat-app design that allows for a log transfer using an encrypted database. We further discuss the applicability and ethical considerations of WhatsApp password generation later in Section 6.3.

6. Discussions

6.1. Weakness Root Causes

Given the potential impact of our decryption-key recovery on the analyzed chat apps, we conducted an analysis of why our technique is possible. The following are our key observations on why our analysis technique can still attack widely-popular chat apps.

- The decryption-key construction is sometimes performed within the Dalvik code of the chat apps. As Dalvik bytecode can be easily reverse engineered and analyzed, the key construction steps can thus be inspected easier.
- The formed key is passed as a parameter in clear among the app methods. As such, dynamic monitoring can easily reveal the passed key and other accompanying pieces of sensitive information in their clear final form.
- The inclusion or customization of third-party libraries, particularly those related to database encryption/decryption, still largely maintains the key data structures and main operational methods of the libraries. Inspection of the libraries, especially the open-source ones, will therefore make code analysis and dynamic monitoring of the chat apps become significantly easier.
- The decryption-key formulation of the analyzed chat apps depends on deterministic device-dependent information, such as IMEI number as in WeChat or Google account name as in WhatsApp. An attack technique like ours can therefore simulate a key generation process of a remote target device as explained in Section 5.

Additionally, for chat apps that allow for a log transfer to a different device like WhatsApp, the following represent insecure or imprudent practices that can contribute to a successful decryption of the encrypted database by an attacker:

- Lack of a record keeping and verification by the chat-app server on the association between a mobile device's phone number and user account information.
- Lack of privacy protection that prevents the reading of information stored within the encrypted file's header.

6.2. Recommended Counter Measures

The following are counter measures that can be adopted by chat-app developers to strengthen their future app versions:

- The key-construction step can be done within the last native method performing/invoking the decryption operation. This way, dynamic analysis can thus inspect only the flow of individual pieces of information that finally constitute the key into the native method.
- Inclusion or customization of a third-party library needs to substantially alter the key data structures and main operational methods related to the decryption key. Obfuscation techniques, particularly the layout and data obfuscation techniques, can be very useful for this.
- Control-flow obfuscation can also be applied to the Dalvik code so that static taint-tracking analysis would find it more difficult to inspect the flow of device-specific information that is used for key construction.
- The Dalvik code may also spuriously flow the relevant sensitive device-specific information in order to complicate the information-flow analysis of the chat apps.
- Chat-app developers may also explore the possibility of incorporating a random value in formulating a decryption key. This measure will therefore hinder attacks that simulate the key-generation process of a remote device, such as ours. There exists, however, an issue of securely storing such a random value. If the value is stored locally on a mobile device, its confidentiality thus needs to be protected. It could be encrypted, for example, using the mobile user's account password.

For a chat-app model that provisions a log transfer to a different device, and whose remote server issues the decryption key upon request by a device, the following counter measures can additionally be exercised:

- The chat-app server needs to keep track of the association between a mobile device's phone number and user account information. The server returns the requested key only if the two pieces of information supplied in a query match.
- The database file header can be encrypted using a user-chosen password. A chat-log transfer will proceed correctly only if the user enters the correct password.

6.3. Ethical Considerations and Applicability of Our Attacks

We discuss below ethical considerations of our techniques and obtained results. We additionally review the applicability conditions of our techniques, which also relates to the security implications of the two analyzed chat apps.

We believe that our results sufficiently highlight potential security issues with the design of current popular chat apps. We have not contacted the developers of WeChat and WhatsApp to inform them of our findings for the following reasons. Techniques to decrypt the current version of WeChat and older versions of WhatsApp are already known [12,11]. Additionally, while our technique expounds a systematic and generalized approach to recovering a target chat app's key and decrypting its log, some attack prerequisites (as summarized below) do exist. Instead, we choose to share our results in this publication so as to reach a wider audience within the security community. Furthermore, we also suggest several counter measures that can be applied to prevent the attacks. This is in line with a practice accepted by the security community to responsibly disseminate potential security weaknesses, and suggest measures to address them.

When good security precautions are fully exercised by mobile users, our techniques do not directly apply. In our experiments on WeChat reported in Section 4.1, we were able to obtain its encrypted database files since the target device was rooted. Likewise, WhatsApp users must physically safeguard the SD card of their devices. They also should keep their Google account names secret, and make them difficult to guess. Lastly, the key retrieval from WhatsApp server works only if it insecurely issues an inquired key belonging to other number. Nevertheless, our attacks clearly highlight potential threats to mobile user privacy that could work on rooted or exploitable devices in conjunction with planted trojan apps, which are not uncommon.

Given the importance of our findings to chat-app security, we hope that our techniques and results presented in this paper can help the security community become aware of the privacy threats. We also hope that our paper can provide useful insights on how the security community can move forward together in securing future chat apps.

7. Conclusion

We have presented our information-flow based approach to discovering the decryption key of log database files from two highly popular chat apps, namely WhatsApp and WeChat. We have shown that, despite the employed code-obfuscation technique, we can discover the keys and obtain useful information on the key construction process. We have additionally detailed the employed string de-obfuscation, encrypted database file structure, and decryption-key formulation of the latest WhatsApp version. Moreover, we have additionally elaborated how we can construct a chat-app version that can simulate other devices' key generation process. Given our generalized approach to discovering and generating the decryption key, the proposed technique can still apply to different chat-app versions as long as they still retain the same design and practice of employing third-party cryptographic libraries and passing the key information into the libraries. Lastly, we have provided an in-depth analysis of why our technique can work on widely-popular chat apps, and listed measures that can be adopted by chat-app developers to improve their future app versions and better protect the privacy of billions of their users.

Acknowledgment

This material is based on research work supported by the Singapore National Research Foundation under NCR Award No. NRF2014NCR-NCR001-034.

References

[1] WhatsApp Messenger, https://play.google.com/store/apps/details?id=com.whatsapp.

[2] WeChat, https://play.google.com/store/apps/details?id=com.tencent.mm.

[3] C. Mulliner, Android DDI: Dynamic Dalvik Instrumentation, 30th Chaos Communication Congress, http://www.mulliner.org/android/feed/mulliner_ddi_30c3.pdf, 2013.

[4] C. Mulliner, ddi – Dynamic Dalvik Instrumentation Toolkit, https://github.com/crmulliner/ddi.

[5] C. Mulliner, Binary Instrumentation on Android, SummerCon, 2012, http://www.mulliner.org/android/feed/binaryinstrumentationandroid_mulliner_summercon12.pdf.

[6] C. Mulliner, adbi – The Android Dynamic Binary Instrumentation Toolkit, https://github.com/crmulliner/adbi.

[7] Apktool, http://ibotpeaches.github.io/Apktool.

[8] jadx – Dex to Java decompiler, https://github.com/skylot/jadx.

[9] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel, FlowDroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps, 35th Conference on Programming Language Design and Implementation (PLDI), 2014.

[10] M. I. Gordon, D. Kim, J. Perkins, L. Gilham, N. Nguyen, and M. Rinard, Information Flow Analysis of Android Applications in DroidSafe, Network and Distributed System Security (NDSS), 2015.

[11] M. Ibrahim, How to Decrypt WhatsApp crypt7 Database, Digital Internals, http://www.digitalinternals.com/security/decrypt-whatsapp-crypt7-database-messages/307, May, 2014.

[12] F. M. Darus, How to decrypt WeChat EnMicroMsg.db database?, Forensic Focus, http://articles.forensicfocus.com/2014/10/01/decrypt-wechat-enmicromsgdb-database, 2014.

[13] Wikipedia, WeChat, https://en.wikipedia.org/wiki/WeChat.

[14] SQLCipher, https://www.zetetic.net/sqlcipher.

[15] Wikipedia, WhatsApp, https://en.wikipedia.org/wiki/WhatsApp.

[16] WhatsApp Blog, One billion, https://blog.whatsapp.com/616/One-billion, February, 2016.

[17] WhatsApp, Frequently Asked Questions – How do I move my chat history over to my new Android phone?, https://www.whatsapp.com/faq/en/android/20902622.

[18] C. Mulliner, W. Robertson, and E. Kirda, VirtualSwindle: An automated attack against in-app billing on Android, 9th ACM Symposium on Information, Computer and Communications Security (ASIACCS '14), 2014.

[19] IDA Pro, https://www.hex-rays.com/products/ida.

[20] C. Collberg, C. Thomborson, and D. Low, A taxonomy of obfuscating transformations, Technical Report, 148, University of Auckland, Auckland, New Zealand, 1997.

[21] soot-infoflow-android wiki, https://github.com/secure-software-engineering/soot-infoflow-android/wiki.

[22] E. Khalaj, R. Vanciu, and M. Abi-Antoun, Comparative evaluation of static analyses that find security vulnerabilities, Technical Report, Wayne State University, Detroit, MI, 2014.

[23] L. Qiu, Z. Zhang, Z. Shen, and G. Sun, AppTrace: Dynamic trace on Android devices, International Conference on Communications (ICC), 2015.

[24] J. Schutte, R. Fedler, and D. Titze, ConDroid: Targeted dynamic analysis of Android applications, 29th International Conference on Advanced Information Networking and Applications (AINA), 2015.

[25] Android Open Source Project, AccountManager, http://developer.android.com/reference/android/accounts/AccountManager.html.

[26] adbi – Issues, Too many memory mapping, https://github.com/crmulliner/adbi/issues/5.

[27] SQLCipher, sqlcipher/sqlcipher, https://github.com/sqlcipher/sqlcipher/tree/910fd70a3fdeeaa937e0d26940f924dbefe7ba77/src.

[28] SQLCipher, SQLCipher for Android application integration, https://www.zetetic.net/sqlcipher/sqlcipher-for-android.

[29] SQLCipher, SQLCipher API, https://www.zetetic.net/sqlcipher/sqlcipher-api.

[30] Wikipedia, PBKDF2, https://en.wikipedia.org/wiki/PBKDF2.

[31] Android Open Source Project, Dalvik bytecode, https://source.android.com/devices/tech/dalvik/dalvik-bytecode.html.

[32] Oracle, javax.crypto – Class Cipher, https://docs.oracle.com/javase/7/docs/api/javax/crypto/Cipher.html.

A Systems Approach to Cyber Security
A. Roychoudhury and Y. Liu (Eds.)

doi:10.3233/978-1-61499-744-3-19

VeriFormal: An Executable Formal Model of a Hardware Description Language

Wilayat KHAN [a], Alwen TIU [a], David SANAN [a]
[a] *Nanyang Technological University (NTU), Singapore*

Abstract. The high security requirements of cyber-physical systems and the critical tasks they carry out make it necessary to guarantee the absence of any vulnerability to security attacks and that they have no unexpected behaviour. The size and complexity of the underlying hardware in cyber-physical systems are increasing and so is the risk of failures and vulnerability to security threats. Checking for errors and security holes in the early phases of hardware production is generally considered an effective approach. To ease the process of designing and testing in the early stage, hardware description languages such as Verilog are used to design hardware. Hardware designs in such description languages, however, do not correspond to mathematical models. Hence we cannot reason about hardware designs and formally verify their correctness and security related properties. In this paper, we develop a formal model of Verilog in the theorem prover Isabelle/HOL. Our model covers most constructs used in hardware designs. With our model, one can analyse Verilog designs and execute them for simulation. More importantly, our model enables formal reasoning about interesting properties for Verilog designs. To complete our tool chain, we build a translator which automatically translates an existing Verilog design to equivalent design in our formal model.

Keywords. Formal hardware verification. Verilog. Trusted low-level synthesis. Isabelle/HOL.

1. Introduction

Critical cyber-physical systems requires total absence of failures in their functionality, and also to ensure a selected set of safety and security properties. During last decades, formal methods have been playing a key role in the verification of safety-critical systems, and have been successfully applied to a large number of cases [23]. Ensuring the correctness of hardware designs is an important part of computer system verification. One of the most successful cases of hardware verification using formal methods is [13], where traditional testing techniques are replaced with formal methods for the verification of the functional behavior of the microinstructions in the Intel® Core™ i7 processor. Formal methods have also been applied on the detection of trojans on the design of System of Chip (SoC) [12,17]. Trojans in SoCs is one of the majors hardware security threats [21]. Introducing subtle modifications on the SoC's design, like modifying the behavior of low observability signals or statements that are not executed very often, it is possible to add vulnerabilities in the final hardware that are extremely difficult to detect if formal methods are not used. To verify hardware circuits and systems, one proves the correctness of the implementation with respect to some formal specification. However, due to the size and the complexity of computer systems, hardware design is usually done using hardware description languages, which are not necessarily defined in formal semantics, thus we cannot reason about hardware designs directly. That is, there is a gap between hardware description languages and the logic which we use in the verification. To remedy this, we need a formal model of hardware description language, ideally defined in a proof assistant, to enable semi-automated verification of hardware designs. We present such a formal framework, called VeriFormal, which is formalized in the proof assistant Isabelle/HOL. The development of VeriFormal is part of a larger project on verifying safety and security properties of hardware and software systems.

The two most popular hardware description languages are Verilog and VHDL. There have been various attempts at providing a formal semantics for Verilog and VHDL, dating back at least a couple of decades. Gordon [11] defined a simplified version of Verilog, and discussed its formal semantics. The most relevant is the work by Meredith et al. [15] where they have defined an executable semantics of Verilog in the rewriting logic tool Maude [8]. Our work is based on the same model and is discussed in more detail later. Braibant et al. [6] defined a deep embedding of Fe-Si, a simplified version of the higher-level language Bluespec, in the theorem prover Coq. Their work differs from ours in that they investigate high-level verification while VeriFormal deals with design at low level. Love et al. [14] formalized a synthesizable subset of Verilog in Coq, however, the framework does not support asynchronous circuits and multiple register updates are not allowed in a single clock cycle. Another closely relevant work is the verification framework by Slobodova et al. [19] where they model hardware using the formally defined hardware description language E [5] that is deeply embedded in ACL2. They symbolically simulate the design units (modelled as E modules) using AIG as the Boolean function representation. The framework relies on the formal language E, which models a subset of Verilog [20] and as the source is not available, a thorough comparison with VeriFormal can not be made.

In the literature, one can observe that the formal models of VHDL are usually not mechanised. Fuchs and Mendler gave a functional semantics for VHDL delta-delay based on focus and streams [10]. Similar approach was taken by Breuer et al. to formalise unit-delay [7]. Olcoz [16] and Döhmen and Herrmann [9] gave a formal model of VHDL using petri-nets. Börger et al. defined a fragment of VHDL using evolving algebras [4]. The tool-oriented formalisations include Van Tassel's model of a fragment of VHDL called Femto-VHDL in HOL [22], and Reetz and Kropf's flow-graph based semantics of VHDL, also formalised in HOL [18]. Although the above formalisations mainly focus on fragments of VHDL, it is possible to extend them to the full language.

Besides user preferences, Verilog and VHDL are different due to their characteristics. One can think of Verilog as C and VHDL as Ada [11]. Verilog is weakly-typed and less deterministic. The user has to be extra careful when making deterministic Verilog designs. On the other hand, VHDL is strongly-typed and very deterministic. Since both languages are widely used, it is equally important to have formal models for them. However, the differences in Verilog and VHDL make it difficult to cope with the two languages at once. Hence this paper only focuses on Verilog.

We give a formal language VeriFormal[1], by deeply embedding Verilog constructs in Isabelle/HOL, which covers most of the Verilog language. Our semantic model of

[1] All the source code can be found at link `http://securify.sce.ntu.edu.sg/HWVer/VerilogModel/VerilogModel.zip`

Verilog is based largely on the formal model formalized in Maude [15]. An advantage of Maude formalization is that the formal model itself is executable. For our purpose of integrating verification of hardware and software that run on top of them, we found the rewriting framework that Maude is based on to be inadequate in various reasoning tasks that require (co)inductive reasoning principles and other set theoretic methods, e.g., to reason about refinements between hardware designs and instruction set architecture specifications of specific processor architecture. For these reasons and to integrate better with our existing formalization efforts, we have opted to develop a formal model of Verilog using the expressive theorem prover Isabelle/HOL. We develop the operational semantics of Verilog in a functional style, allowing us to also execute the specifications of hardware designs in Isabelle/HOL, much like the Maude framework. The main advantage of VeriFormal, however, is that it allows one to leverage the very extensive reasoning support of Isabelle/HOL. VeriFormal, in addition, is augmented with a type checker predicate to check the correctness of Verilog modules and a translator to encode existing designs automatically.

The main contributions in this work include:

- VeriFormal: an executable formal model of Verilog in Isabelle/HOL;
- a type checker program to check correctness of Verilog modules;
- a prototype translator to encode Verilog designs into the Isabelle/HOL formal model.

The rest of the paper is organized as the following: In next section, a short introduction to Verilog is given. Sections 3 and 4 give detailed explanations of the syntax and operational semantics of VeriFormal, respectively. Section 5 explains the prototype translator to translate existing Verilog designs to the VeriFormal setting. The type checker predicate is explained in Section 6. The two most important applications, execution and reasoning about the designs, are discussed in Section 7. The coverage of the Verilog language in VeriFormal is outlined in Section 8. Finally, the last section concludes the work.

2. Introduction to Verilog

Verilog is one of the most popular Hardware Description Languages (HDL) used by hardware designers to model and simulate hardware functions. HDLs, including Verilog, are different from other standard programming languages in the sense that the latter are sequential while HDLs describe parallel concurrent behaviour. A circuit design described in Verilog consists of several smaller modules to allow code reuse and abstraction, where each module consists of definitions of input and output ports and defines logical relationship between them.

A simple Verilog module, shown in Figure 1, takes two eight-bit integers as input and stores their sum in variable of type `reg`. This example highlights some important features of Verilog. A module starts with keyword `module` followed by module name *addint* followed by a list of input and output ports (line 1). Verilog module declaration is followed by a set of data type declarations (lines 2-4). There are two major groups of Verilog data types: *variable* (such as `reg`) and *net* (such as `wire`). Identifiers of both of these types can either be single bits or bit vectors. Identifiers of type *variable* represent

```
1  module addint(Xi, Yi);
2   input [7:0] Xi, Yi;
3   wire [7:0] X, Y;
4   reg [7:0] Z;
5
6   assign #0 X = Xi;
7   assign #0 Y = Yi;
8
9   always #2
10    begin
11     Z = X + Y;
12     $finish;
13    end
14 endmodule
```

Figure 1. Verilog module

the notion of states while *net* models wires to carry information within a design. The input and outputs to the module are defined using the keywords `input` and `output` with automatically assumed type *wire*. Identifiers of type *wire* can also be explicitly defined using the keyword `wire`.

According to Verilog standard [3], identifiers of type *variable* (e.g., `Z`) may only be driven by procedural assignments within procedural blocks `initial` and `always` (line 11) and nets (e.g., `X` and `Y`) can only be driven by continuous assignments (lines 6-7). The `always` keyword may be followed by delay (e.g., #2) or a trigger of the form @(SL), where the former delays the entire body of the always block by 2 simulation cycles (explained later) and the later puts the body into wait until a condition is met. The `SL`, called *sensitivity list*, is a set of variables or nets, that provides such condition. For example, the body of the always block starting with `always @(X)` gets executed when the value of `X` changes. Any other statement, such as assignments, can be delayed or put to wait for a trigger. A continuous assignment is triggered whenever any value on the right side of assignment (e.g., `Xi` in line 6) changes. An initial block is similar to always block except it does not have delays and triggers (after keyword `initial`) and the execution starts with it.

Procedural assignments can be further divided into `blocking` and `non-blocking` assignments. A blocking assignment *blocks* the execution of statements that follow until the assignment is completely executed, while in case of non-blocking assignment, execution starts immediately with statement following it. A wide range of operators, such as bit reduction, relational, arithmetic, shift so on, can be used to build expressions to make the body of the assignments. An existing module can also be instantiated inside another module.

3. Language Syntax

The language syntax includes formal definitions of Verilog expressions, statements and *top* statements. To ease understanding, simple notations are used here and the details can be found in the source code.

3.1. Expressions

Identifier names, ranged over by q, optionally followed by brackets for bit-select, bit-slice and index, are expressions. Binary word data is represented using bitvector v (as in [15]), which is a pair of integer (value) and natural number (length in bits). An alternative definition would be using Isabelle words which are tedious to deal with. Other expressions include binary, unary, right shift and left shift operations and conditional operator. These are formalised as below.

$e ::=$	*expressions*
$\mid\ q$	identifier names, bitselect, slice, index
$\mid\ v$	bitvectors (value, size) pair
$\mid\ \bullet e$	unary operation
$\mid\ e \circ e'$	binary operation
$\mid\ e \gg n$	right shift
$\mid\ e \ll n$	left shift
$\mid\ b\ \mathbf{?}\ e\ \mathbf{:}\ e'$	conditional operation

3.2. Statements

The definition of statements is given below.

$s ::=$	*statements*
$\mid\ skip$	skip
$\mid\ finish$	finish
$\mid\ disab\ q$	disable name q
$\mid\ le = \#i\ e$	blocking assignment
$\mid\ le = @(sl)\ e$	triggered blocking assignment
$\mid\ le <= \#i\ e$	non-blocking assignment
$\mid\ le <= @(sl)\ e$	triggered non-blocking assignment
$\mid\ if\ e\ then\ s\ else\ s'$	if-then-else
$\mid\ while\ e\ s$	while loop
$\mid\ begin\ q\ s\ end$	code block
$\mid\ case\ e\ lcs\ s\ endcase$	case statement
$\mid\ @(sl)\ s$	triggered statement
$\mid\ \#n\ s$	delayed statement
$\mid\ s1;;s2$	sequence

The statement *skip* is a dummy statement representing a silent transition without making any changes to the state. Statements *finish* and "*disab q*" correspond to Verilog `$finish` and `disable` keywords. The next four statements correspond to Verilog blocking and non-blocking assignments, where the integer i (after #) represents optional delay (-ve value implies no delay). The expressions list le on left is the tuple (e.g., $\{q_2, q_1, q_0\}$) of identifiers (names of type `reg`) and e on the right is the body of assignment. The sl represents the sensitivity (event control) list with elements $signal \in \{posedge\ e,\ negedge\ e,\ e,\ *\}$, with the first three respectively correspond to positive and negative polarities and value change. The $*$ marks all identifiers, in the expression or in the statement that follows, as signals of sensitivity list.

The statements *if-then-else*, *while*, *begin..end*, *case* and $@(sl)s$ correspond to standard Verilog statements, where q, e, s, and lcs are name, expression, statement and list of case statements, respectively. The number n in the delayed statement is a non-negative delay and the sequence statement combines two statements into a sequence.

3.3. Verilog top statements

Expressions and statements make up Verilog statements and keywords (henceforth top statements), which are used to build Verilog modules. These statements exactly correspond to Verilog net and variable declarations, continuous assignment and initial and always block statements[2] . The integer i after # represents optional delay (similar to procedural assignments) and le is the list of names (expressions) modelling the tuple of identifiers on the left. An always block, for example, with time control followed by body would be represented by *always*, followed by triggered/delayed statement which normally would be in a *begin..end* block of statements.

$$\begin{array}{lll} t ::= & & \textit{top statements} \\ \mid & assign\ \#i\ le = e & \text{continuous assignment} \\ \mid & always\ s & \text{always block} \\ \mid & \ldots & \end{array}$$

3.4. Processes

Statements are executed by scheduling events or creating processes. All statements are processes and continuous assignments and always blocks are executed as $palw(sl,\ p)$ (process always) and $pca(d,\ lq,\ e,\ v)$ (process continuous assignment), respectively. The body of an always process p gets executed when any of the signal's value in sl changes. Similarly, the body of continuous assignment process e evaluates to value v whenever any identifier's value in e changes and the assignment to names in lq occurs after delay d.

$$\begin{array}{lll} p ::= & & \textit{processes} \\ \mid & s & \text{statement} \\ \mid & palw(sl, p) & \text{always process} \\ \mid & pca(d, lq, e, v) & \text{continuous assignment process} \end{array}$$

[2]Module instantiation is omitted here. See Section 5 for detail.

3.5. Events

To execute Verilog top statements, different events are registered. An assignment, for example, may be registered as an event and is then executed or it is scheduled as an active process and then converted to event. A variable update with a value (bitvector) is represented with update event *upd*$(e,\ v)$ where the expression e represents the identifier (name, part-select, index or bit-select) on left-hand side and v is the value (bitvector) of the expression.

For tuples on the left-hand side of the assignment, a list of update events (one for each name in left-hand side) is created, which are either scheduled for execution in order (blocking or continuous assignments) or stored as a list in *non-blocking assign update events* in the state for later retrieval (non-blocking assignments). For the latter, all the events are executed by combining them as a list lev in a single event *updl*$(lev,\ p)$, where p represents the rest of process (computation) following the non-blocking assignment. Events may be delayed temporarily (inactive events), scheduled at a specific simulation cycle (future events), or set to wait until some change occurs to signal(s) (listening events).

$ev ::=$		*events*
$\mid$	$upd(e,\ v)$	update
$\mid$	$updl(lev,\ p)$	update list
$\mid$	$ina(p)$	inactive
$\mid$	$fut(n,\ p)$	future
$\mid$	$listn(sl,\ p)$	listening

3.6. Program configuration

A Verilog program is defined as a Verilog module which has a list of input and output ports and list of top statements forming the body of the module. A module is then transformed to a program configuration (10-tuple state FAILED TURN): list of future events, active processes, inactive and listening events, environment (name-value mapping), set of disabled names, current simulation cycle (natural number n), list of update events, finish flag[3], and non-blocking assign update events. Different fields of configuration C are accessed using the dot operator (e.g., C.F). The notation $C\langle f = f'\rangle$ is used to update the value of field f to new value f' while keeping the rest of fields unchanged. A much simplified version $C\langle f'\rangle$ is often used when f' is instantiated to a specific field, such as $C\langle F'\rangle$ changes future events to F' and $C\langle L - L'\rangle$ removes events in L' from listening events L. An initial configuration has all the fields set to empty, T to 0, and the flag R is set to False.

4. Operational Semantics

A Verilog program is scheduled as events and active processes where each in turn is comprised of expressions and statements. To execute these events and processes, the ex-

[3]Letter R is used for finish flag to make the tuple a meaningful word to ease remembering.

pressions and statements in their body need to be executed. In the environment, expressions are evaluated to values, using big-step semantics $E \vdash e \downarrow v$, by retrieving the value of identifiers and evaluating binary, unary, shift and conditional operations. Formal definition of statement execution is defined as a relation $s \parallel C \longrightarrow C'$, which is detailed below.

4.1. Process blocking assignments ($s \parallel C \longrightarrow C'$)

Procedural assignments can be either blocking or non-blocking and they are executed differently. The major difference is that the statements preceded by a blocking assignment are *blocked* until the assignment is completely executed, while statements preceded by a non-blocking assignment are executed *before* the assignment. The semantics include rules for sequence statements with a blocking assignment at head. Rules for single blocking statement are much simpler and not included here and instead are modelled as sequences with blocking statement at head followed by *skip* (rule [P-SINGLE]).

(P-BA)

$$\frac{\begin{array}{c} C.E \vdash e \downarrow v \\ s_1' = (lq = exp(v')) \qquad (v_q, v') = slice(v, q) \qquad L' = triger(E(q), q, v_q, L) \end{array}}{(q\#lq = e);;s_2 \parallel C \longrightarrow s_1';;s_2 \parallel C\langle E@(q, v_q), actle(L')@A, L - L'\rangle}$$

(P-BA-EMP)

$$\frac{}{([] = e);;s_2 \parallel C \longrightarrow C\langle A@[\triangleright s_2]\rangle}$$

(P-ZDBA)

$$\frac{C.E \vdash e \downarrow v \qquad ev = inae(\triangleright(lq = e;;s))}{(lq = \#0\ e);;s \parallel C \longrightarrow C\langle I@[ev]\rangle}$$

(P-DBA)

$$\frac{d > 0 \qquad t = C.T + nat(d) \qquad ev = fut(t, \triangleright(lq = e;;s))}{(lq = \#d\ e);;s \parallel C \longrightarrow C\langle F@[ev]\rangle}$$

(P-TBA)

$$\frac{sl = [*]\ ?\ sl' = sensl(e) : sl' = sl \qquad ev = listn(sl', \triangleright(lq = e;;s))}{(lq = @(sl)\ e);;s \parallel C \longrightarrow C\langle L@[ev]\rangle}$$

(P-SINGLE)

$$\frac{s \in \{blocking\ assignment\} \qquad s;;skip \parallel C \longrightarrow C'}{s \parallel C \longrightarrow C'}$$

The first rule [P-BA] models blocking assignments without delay[4]. First the right-hand expression of the assignment is evaluated to value v. The function $slice(v, q)$ builds a bitvector v_q (of n least significant bits of v where n is the size of right-most identifier q in tuple) and shifts v right by n making the next bits ready for the next assignment. The value of q in environment is updated with new vector v_q. This update may trigger listening events which are removed from listening events and converted to active processes using function $actle()$. A listening event is triggered if the new value is different than the old

[4] For simplicity, # -1 is omitted from statements without delay.

one and the event is listening for change to the identifier (updated variable is in sensitivity list). If the listening event was registered by continuous assignment, only a change in the value can trigger it, while polarity is also taken into consideration for events registered by an *always* block. A blocking assignment may update the environment, active processes, and listening events. If there is a tuple on the left-hand side, the execution starts with the next assignment s'_1 where the remaining bits in bitvector v' (right shifted version of v) are assigned to the next identifier on the left-hand side, otherwise, the execution starts with statement that follows (rule P-BA-EMP). The constructor *exp* converts a bitvector to an expression.

A sequence that has blocking assignment at head with zero delay is added to *inactive* events. The notation $\triangleright$ is used as a constructor to convert a statement to a process. Similarly, a blocking assignment with non-zero positive delay followed by statement is added to future events, where the time of the future event is the current time plus delay in the assignment. Triggered blocking assignment (blocking assignment with right-hand side preceded by sensitivity list) followed by a statement is added to listening event. In case of $*$, the function *sensl* calculates the new sensitivity list.

4.2. *Process non-blocking assignments ($s \parallel C \longrightarrow C'$)*

(P-NBA)
$$\frac{C.E \vdash e \downarrow v \qquad lev = mkuel(C.E, lq, v)}{(lq <= e) \parallel C \longrightarrow C\langle N@lev\rangle}$$

(P-ZDNBA)
$$\frac{ev = ina(\triangleright(lq <= e))}{(lq <= \#0\ e) \parallel C \longrightarrow C\langle I@[ev]\rangle}$$

(P-DNBA)
$$\frac{d > 0 \qquad t = C.T + nat(d) \qquad ev = fut(t, \triangleright(lq <= e))}{(lq <= \#d\ \mathrm{e}) \parallel C \longrightarrow C\langle F@[ev]\rangle}$$

(P-TNBA)
$$\frac{sl = [*]\ ?\ sl' = sensl(e) : sl' = sl \qquad ev = listn(sl', \triangleright(lq <= e))}{(lq <= @(sl)\ e) \parallel C \longrightarrow C\langle L@[ev]\rangle}$$

(P-NBA-SEQ)
$$\frac{rhs = \{e, \#d\ \mathrm{e}\ , [@](sl)\ \mathrm{e}\} \qquad s \parallel C \longrightarrow C' \qquad (lq <= rhs) \parallel C' \longrightarrow C''}{(lq <= rhs);; s \parallel C \longrightarrow C''}$$

A non-blocking assignment is scheduled at a later time depending on the time control and execution immediately starts with the statement that follows. If there is no timing control (delay or sensitivity list), then an update event (list in case of tuple on the left-hand side) is created and stored in non-blocking assign update events. The function *mkuel* makes update event list for the normal non-blocking assignment. Assignments with zero-delay are added to inactive events and those with non-zero positive delay are added to future events. Similarly, assignments with a sensitivity list are added to listening events.

4.3. *Process non-assignment statements ($s \parallel C \longrightarrow C'$)*

Non-assignment statements consist of all single statements except blocking and non-blocking assign statements and sequences with non-assignment statement at the head.

The dummy statement *skip* does not change the state (rule [P-SKIP]). The statement *finish* changes the termination flag R to True, signaling the termination of the program execution (rule [P-FINISH]). The statement *disab* adds the name q to the set of disables if it is not already added (rule [P-DISAB]). Statements with zero-delay are added to inactive events and with non-zero positive delay are added to future events (rules [P-ZD] and [P-NZD]).

$$\text{(P-SKIP)}\ \frac{}{skip \parallel C \longrightarrow C} \qquad \text{(P-FINISH)}\ \frac{}{finish \parallel C \longrightarrow C\langle R = true\rangle} \qquad \text{(P-DISAB)}\ \frac{}{(disab\ q) \parallel C \longrightarrow C\langle D \uplus \{q\}\rangle}$$

$$\text{(P-ZD)}\ \frac{ev = ina(\triangleright s)}{\#0\ s \parallel C \longrightarrow C\langle I@[ev]\rangle} \qquad \text{(P-NZD)}\ \frac{n > 0 \qquad ev = fut(C.T + n, \triangleright s)}{\#n\ s \parallel C \longrightarrow C\langle F@[ev]\rangle}$$

$$\text{(P-TRIG)}\ \frac{sl = [*]\ ?\ sl' = sensl(s) : sl' = sl}{@(sl)\ s \parallel C \longrightarrow C\langle L@[listn(sl', \triangleright s)]\rangle}$$

$$\text{(P-SEQ)}\ \frac{s_1 \in \{assignments\} \qquad s_1 \parallel C \longrightarrow C' \qquad s_2 \parallel C' \longrightarrow C''}{s_1;;s_2 \parallel C \longrightarrow C''}$$

Triggered statements are added to the list of listening events (rule [P-TRIG]). To process a sequence with non-blocking assignment at head, the assignment is scheduled at a later time (using any of the rules above) and the execution starts immediately with the statement that follows the assignment (rule [P-SEQ]).

4.4. Execute process ($p \parallel C \longrightarrow C'$)

A process is executed by processing all the statements and events that comprise the process. A statement is processed by first reducing it, using a big-step semantics $C \vdash s \downarrow s'$, to a format suitable for processing and then it is processed using any of the rules described above. Always and continuous processes, *palw* and *pca*, are executed by processing the process and assignment in the body and all the processes activated as result of the first one. When an always or continuous process is completely executed, it is converted back to a listening event to listen to other updates. As suggested by Gordon [11], the body of the process continuous assignment is executed by processing a delayed blocking assignment statement. Note that, the assignment must use the value of the expression calculated at the time when the event was triggered. The extra argument v is used to do exactly that and the other three arguments are used to re-create the same listening event. As mentioned above, the last rule continues executing all the processes created as side effect of the process.

(EP-STM)
$$\frac{C \vdash s \downarrow s' \qquad C.A = [] \qquad s' \parallel C \longrightarrow C'}{\rhd s \parallel C \longrightarrow C'}$$

(EP-ALW)
$$\frac{C.A = [] \qquad p \parallel C \longrightarrow C'}{palw(sl,p) \parallel C \longrightarrow C'\langle L@listn(sl,p)\rangle}$$

(EP-CA)
$$\frac{C.A = [] \qquad (lq = \#d\ v) \parallel C \longrightarrow C' \qquad ev = listn(sensl(e), pca(d,lq,e,e))}{pca(d,lq,e,v) \parallel C \longrightarrow C'\langle L@[ev]\rangle}$$

(EP-EFFECT)
$$\frac{C.A = p\#lp' \qquad p \parallel C\langle A = []\rangle \longrightarrow C'}{p\#lp' \parallel C \longrightarrow lp' \parallel C'}$$

4.5. *Executing Verilog module*

Starting from the initial configuration, a new configuration is built from the body (list of top statements) of the Verilog module and events are scheduled accordingly. Verilog declarations populate the environment and continuous assignments are converted to either update, listening, inactive or future events, depending on the existence of identifier(s) in the right-hand body and the value of delay. The body of the initial block is scheduled as an active statement process and an always block is scheduled as active delayed process or listening event depending on the value of delay.

Events and processes in the stores are executed in a specific order defined by the rules of the relation $C \Longrightarrow C'$ and given in the Figure 2[5]. The execution starts with update events (if there is any). Each update event is executed by creating a single blocking assignment which eventually updates a variable in the environment. The process p in the update event list *updl* represents the computation that follows. For example, a blocking assignment followed by other statements in a sequence statement generates a list of (singleton list in case of single identifier on the left of assignment) update events combined together in a single *updl* with a statement that follows makes the process p. An *updl* is executed when all the update events in the list are processed and the rest of computation p is added to active processes (rules [SIM-UPDL], [SIM-UPDL-EMP] and [SIM-UPD]).

If there are no update events in the store, active processes are executed one by one followed by first activating (using function *activate*()) inactive events and then executing them (rules [SIM-ACTP] and [SIM-INAP]). When all of the update and inactive events and active processes are executed, execution starts with non-blocking assign update events scheduled by non-blocking assignments. To execute non-blocking assign update events in the store, they are combined together in a single *updl* event and are executed (rule [SIM-NBAUE]).

[5]The outgoing arrow from each circle is followed only when all events/processes of that kind are exhausted.

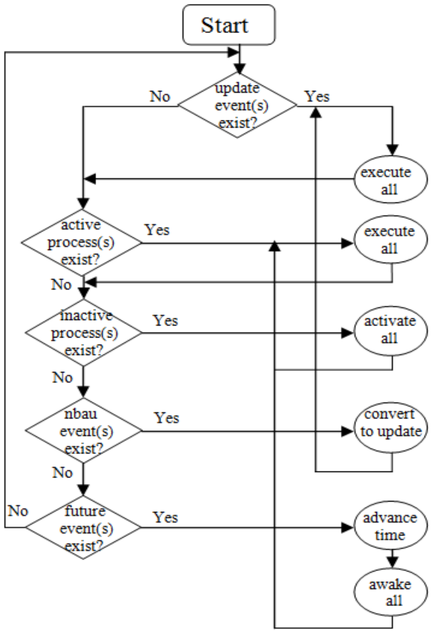


Figure 2. Order of events and processes execution

(SIM-UPDL)

$$\frac{ev = upd(q,v) \qquad ([q] = v) \parallel C \longrightarrow C'}{C\langle updl((ev\#lev),p)\#U'\rangle \Longrightarrow C'\langle updl(lev,p)\#U'\rangle}$$

(SIM-UPDL-EMP)

$$\frac{}{C\langle A, updl([],p)\#U'\rangle \Longrightarrow C'\langle p\#A, U'\rangle}$$

(SIM-UPD)

$$\frac{(q = v) \parallel C \longrightarrow C'}{C\langle upd(q,v)\#U'\rangle \Longrightarrow C'\langle U'\rangle}$$

(SIM-ACTP)

$$\frac{p \parallel C \longrightarrow C'}{C\langle U = [], p\#A'\rangle \Longrightarrow C'\langle A'\rangle}$$

(SIM-INAP)

$$\frac{p = activate(i) \qquad p \parallel C \longrightarrow C'}{C\langle A = U = [], i\#I'\rangle \Longrightarrow C'\langle I'\rangle}$$

(SIM-NBAUE)

$$\frac{ev = updl(N, \rhd skip) \qquad C\langle U = [ev], N = []\rangle \Longrightarrow C'}{C\langle U = A = I = [], N = x\#xs\rangle \Longrightarrow C'}$$

This completes a single pass of simulation. If there are any new events or processes left, which are added by non-blocking assign update events in the last step (right-most last circle in Figure 2), they are not executed. To execute them, the above process is repeated and as this may get into a loop, it is controlled by specifying a number of executions using the natural number *m* in the simulation relation (see below).

4.6. Simulation ($C \overset{m}{\Longrightarrow} C'$)

The parametric simulation relation simulates a state (program) up to *m*th simulation cycle[6] (or *m* passes if there are no future events) or until the program terminates itself (whichever comes first). A program is simulated once if $m = 0$ and the state contains at least a process/event of any kind (except future events) in the store and flag value is False (rule [SIM-ONCE]). The predicate *nextpass* is satisfied if there exists at least a process in any store (except future events) and flag `R` is False and predicate *nextcycle* holds if stores are empty (except future events) and the program has not yet terminated.

(SIM-ONCE)
$$\frac{m = 0 \qquad nextpass(C) \qquad C \longrightarrow C'}{C \overset{\text{m}}{\Longrightarrow} C'}$$

(SIM-PASS)
$$\frac{m > 0 \qquad \neg nextcycle(C) \qquad nextpass(C) \qquad C \Longrightarrow C' \qquad C' \overset{\text{m-1}}{\Longrightarrow} C''}{C \overset{\text{m}}{\Longrightarrow} C''}$$

(SIM-ADVANCE)
$$\frac{m > 0 \qquad nextcycle(C) \qquad C' = newcycle(C) \qquad C' \Longrightarrow C'' \qquad C'' \overset{\text{m-1}}{\Longrightarrow} C'''}{C \overset{\text{m}}{\Longrightarrow} C'''}$$

If the simulation is intended to run for $m > 0$ passes, there must be at least a process/event of any kind or a future event store that is non-empty. If there are no future events in the state C and other stores contain at least a process/event (the predicate $\neg$*nextcycle*(C) $\wedge$ *nextpass*(C) holds), then the state is given a step and the simulation is then continued for the next $m - 1$ passes (rule [SIM-PASS]). This is required that either a process add non-blocking assign update and inactive events to the store and all of them are either executed or simulation continues until *m* passes, or when the program terminates itself.

If the simulation is intended to run for $m > 0$ simulation cycles, there is at least an event in future events store and a program that has not terminated itself, then a new state is prepared for the next cycle and is executed until *m* passes (or when flag `R` changes to True). The new state is prepared for the next cycle (using function *newcycle*()) to execute events scheduled at a future time. To do this, first future events are sorted by time (event with lowest time first) and if an event has been scheduled more than once, only the event with higher time is left and all other occurrences of the same event are *cancelled* [11]. The simulation time is set to the time of event at the head of the sorted list. All the events that were scheduled at this new time are awaken by converting them to active processes and are removed from future events.

[6]A simulation cycle is completed when active processes of any kind (except future events) are executed and the time is advanced. .

5. Verilog Programs Translation

The formal language VeriFormal described in the previous sections accepts Verilog programs written in its formal syntax. To simulate and reason about existing Verilog designs using VeriFormal, one needs to translate them manually to the formal setting. This is a very tedious and error-prone job. This is also true for new designs. To automate this process, a prototype Verilog to VeriFormal translator is developed in C++. The role of the translator is two-fold: it eases the encoding process by translating Verilog design (.v file) into Isabelle theory (.thy file) and translates Verilog constructs, which are not formalized in the language, to equivalent constructs which are formalized. The latter enables VeriFormal to accept a wide variety of Verilog programs while keeping the language simple.

A Verilog module consists of a list of ports, list of (net and/or variable) declarations, continuous assignments, initial and always blocks and module instantiations. Each of these parts is mapped one by one to corresponding VeriFormal statements which then form an Isabelle definition in a theory.

Using the translator, the Verilog module in Figure 1 is automatically translated to a VeriFormal module as shown in Figure 3.

```
1  module ([Xi, Yi])
2  [ input [7:0] [Xi,Yi],
3    wire [7:0] [X,Y],
4    reg [7:0] [Z],
5
6    assign #0 [n X]= n Xi,
7    assign #0 [n Y]= n Yi,
8
9    always ( [#]2
10     BEGIN : [None]
11      ([n Z] [=] [#] -1 (b n X [+] n Y));;
12      ($finish)
13     END ) ]
14 endmod
```

Figure 3. Automatically generated VeriFormal module

The translation is straightforward where each line in Figure 3 corresponds to the same line in Figure 1. The translated module looks similar to original Verilog module with few exceptions: identifiers (separated by comma) in a declaration in Verilog are mapped to a list of identifiers in a VeriFormal declaration (line 2), notations = and # in procedural assignments and operators such as + are enclosed in brackets, constructors precede expressions (e.g., `b` for binary operations, `n` for names and so on). The VeriFormal module generated is then used in Isabelle definition of type `program` and then saved as Isabelle theory *theoryname.thy*.

```
    definition verimodule :: program where
"verimodule = VeriFormal-module"
```

We have included most of the Verilog constructs; however, there are various features which are not directly formalized in the language but mapped to equivalent constructs in VeriFormal. For example, module instantiation is not directly formalized but the instantiation is replaced with instantiated module code inside the parent module as suggested by [11]. The translator currently does not support nested structures such as Verilog module with nested if-then-else, while, case and begin..end blocks.

6. Correctness of Verilog Programs Syntax

The formal syntax described in section 3 establishes rules to write Verilog programs but it does not guarantee that a Verilog program written in it conforms to the Verilog standard. The syntax for continuous assignment, for example, accept identifier of type `reg` on the right-hand side of assignment which is a violation of the standard [3]. To rule out such programs, a correctness predicate `wfprog` is defined which returns True only if the program follows certain rules.

Two sub-predicates, well-formed left-hand side of continuous assignment and well-formed left-hand side of procedural assignment, assert that all the names in the left-hand side of continuous and procedural assignments are declared of type net and variable, respectively. The correctness rules also require that all the names listed in the sensitivity list of triggered statements and expressions are already declared on some type (net or variable). A statement is well-formed if in the sensitivity list, all the expressions and left-hand sides of the assignments in the statements are well-formed. Net declarations are well-formed if the declared names are not declared as variables anywhere in the program. In addition, identifiers declared of type *input* and *inout* must also appear in the port list. Similarly, variable declarations are well-formed if the variables are not declared as net in the program. Continuous assignments, initial and always blocks are well-formed if all the assignments, expressions and statements in them are well-formed.

The objective of this correctness predicate is to check the correctness of the Verilog module before giving it as input to the formal simulator. Even though it is a very simple checker, it can currently capture errors related to type conformance in continuous and procedural assignments, input declarations and ports, duplicate variable (reg) declarations and identifiers in sensitivity list.

We show an example in Figure 4 where a continuous assignment (line 5) derives a variable `y` (of type *reg*). This is not allowed by the standard, but still permitted by VCS online simulator [2]: the online VCS simulator accepted the program in Figure 4 and resulted the value 2 while it was rejected by both, our type checker predicate and another popular simulator Icarus [1].

7. Verification and Simulation

VeriFormal can be used for both reasoning about Verilog designs and simulating them. The main advantage of building rigorous mathematical model of hardware description language is that it enables us to prove theorems about the designs. For example, two simple Verilog designs were developed that multiply a specific number by two: one was implemented using left-shift by one, and the other was a direct multiplication using prod-

```
module increment;
 reg [15:0] x;
 reg [15:0] y;

 assign y = x + 1;

 initial
 begin
   x = 0;
 end

 always @(x[0])
   begin
        x = x + 1;
        $display("y = %d", y);
   end
endmodule
```

Figure 4. Continuous assignment deriving variable

uct operation. Functional equivalence of these two Verilog designs was proved using theorem prover Isabelle/HOL.

In addition to theorem proving about Verilog designs, Verilog programs directly written in (or translated to) VeriFormal can also be executed and the results can be compared with the results of the original Verilog design simulated by any open source simulator. As in [19], this method can be used to discharge many proofs by exhaustive simulation.

To execute Verilog designs, the Isabelle formal model was translated to OCaml using the `export_code` command. A simple Verilog design (Figure 1), which adds two 8-bit integers, was automatically translated to Isabelle definition (Section 5, Figure 3) and then both, the formal model and this definition, were converted to OCaml code. An interface was written in C++ to feed all possible eight-bit input combinations to the Icarus simulator and the OCaml version of VeriFormal and the results were stored in separate files. Through manual checking, it was verified that they both produced the same results. In addition to executing many other simple designs, all the Verilog modules mentioned in [15] were executed using VeriFormal and all the captured errors in Icarus and VCS were confirmed[7].

8. Coverage

We formalize all the features modelled in [15], the richest model in the literature. An exception is `display` which is modelled in VeriFormal differently by storing the result

[7]Verilog designs in [15] were slightly modified to make VeriFormal happy (e.g., Verilog command `$display` is not supported in VeriFormal and instead a register is used to store the result rather than displaying it).

in the environment which is displayed at the end of simulation. Our model also supports multiple always blocks and zero delay. Non-determinism is supported in our model; however, the non-deterministic version of the semantics is not executable.

Even though VeriFormal covers most features in the Verilog language, it still misses some features, which are either not needed for simulation and reasoning, not very important or not directly supported but can be transformed to equivalent constructs by the translator. Features that are not included are `forever`, `for`, `task`, `fork`, `casex`, `casez`, conversion functions, compiler directives, and so on. However, most of these missing features can be encoded using existing constructs (e.g., `for` loop can be modelled using `while`). The model does not support tri-state values and hence does not support `casex` and `casez`. It could be added but then we would loose Isabelle/HOL feature of operating on integers and hence the model would be extremely complicated.

9. Conclusion

To manufacture trusted hardware, the design of hardware described in the description language needs to be verified. However, most of the popular description languages (e.g., Verilog) do not have a complete mathematical foundation and hence we can not reason about hardware designs described in such languages. In this work, we have defined VeriFormal, which is a rigorous formal model of the popular description language Verilog. VeriFormal can be used both to execute Verilog designs and to reason about them. As converting existing or writing new designs in VeriFormal is a tedious job, this process is automated by a prototype translator. VeriFormal is further augmented with a type checker predicate to verify the correctness of designs. The formalisation totals 1528 lines of Isabelle code. The translator is more than two thousands lines of C++ code. This entire work took around 6 months to complete.

Even though VeriFormal establishes a framework towards trusted designs, the translator is a prototype and needs refinement to accept Verilog designs of any complexity. We discussed a very simple proof of equivalence of two Verilog designs, however, proving more general theorems about complex designs would be extremely difficult. A practical example with proof of interesting theorems using VeriFormal would be more interesting to add in future.

References

[1] Icarus Verilog. `http://www.icarus.com/eda/verilog/`. Accessed: 2016-12-06.

[2] VCS Online Simulator. `http://www.edaplayground.com/x/FfR`. Accessed: 2015-12-25.

[3] I. S. Association et al. Ieee standard for verilog hardware description language. design automation standards committee, 2005. *IEEE Std 1364TM-2005*.

[4] E. Börger, U. Glässer, and W. Muller. A formal definition of an abstract vhdl'93 simulator by ea-machines. In C. Kloos and P. Breuer, editors, *Formal Semantics for VHDL*, volume 307 of *The Kluwer International Series in Engineering and Computer Science*, pages 107–139. Springer US, 1995.

[5] R. S. Boyer and W. A. Hunt Jr. The e language. In *Proceedings of the International Workshop on Hardware Design and Functional Languages*. March, 2007.

[6] T. Braibant and A. Chlipala. Formal verification of hardware synthesis. In *Computer Aided Verification*, pages 213–228. Springer, 2013.

[7] P. Breuer, L. FernÃ¡ndez, and C. Kloos. A functional semantics for unit-delay vhdl. In C. Kloos and P. Breuer, editors, *Formal Semantics for VHDL*, volume 307 of *The Kluwer International Series in Engineering and Computer Science*, pages 43–70. Springer US, 1995.

[8] M. Clavel, F. Durán, J. Hendrix, S. Lucas, J. Meseguer, and P. Ölveczky. The maude formal tool environment. In *Algebra and Coalgebra in Computer Science*, pages 173–178. Springer, 2007.

[9] G. Döhmen and R. Herrmann. A deterministic finite-state model for vhdl. In C. Kloos and P. Breuer, editors, *Formal Semantics for VHDL*, volume 307 of *The Kluwer International Series in Engineering and Computer Science*, pages 170–204. Springer US, 1995.

[10] M. Fuchs and M. Mendler. A functional semantics for delta-delay vhdl based on focus. In C. Kloos and P. Breuer, editors, *Formal Semantics for VHDL*, volume 307 of *The Kluwer International Series in Engineering and Computer Science*, pages 9–42. Springer US, 1995.

[11] M. Gordon. The semantic challenge of verilog hdl. In *Logic in Computer Science, 1995. LICS'95. Proceedings., Tenth Annual IEEE Symposium on*, pages 136–145. IEEE, 1995.

[12] X. Guo, R. G. Dutta, and Y. Jin. Hierarchy-preserving formal verification methods for pre-silicon security assurance. In *16th International Workshop on Microprocessor and SOC Test and Verification, MTV 2015, Austin, TX, USA, December 3-4, 2015*, pages 48–53, 2015.

[13] R. Kaivola, R. Ghughal, N. Narasimhan, A. Telfer, J. Whittemore, S. Pandav, A. Slobodová, C. Taylor, V. Frolov, E. Reeber, and A. Naik. Replacing testing with formal verification in intel®coretm i7 processor execution engine validation. In *Proceedings of the 21st International Conference on Computer Aided Verification*, CAV '09, pages 414–429, Berlin, Heidelberg, 2009. Springer-Verlag.

[14] E. Love, Y. Jin, and Y. Makris. Proof-carrying hardware intellectual property: A pathway to trusted module acquisition. *Information Forensics and Security, IEEE Transactions on*, 7(1):25–40, 2012.

[15] P. Meredith, M. Katelman, J. Meseguer, and G. Rosu. A formal executable semantics of verilog. In *Formal Methods and Models for Codesign (MEMOCODE), 2010 8th IEEE/ACM International Conference on*, pages 179–188. IEEE, 2010.

[16] S. Olcoz. A formal model of vhdl using coloured petri nets. In C. Kloos and P. Breuer, editors, *Formal Semantics for VHDL*, volume 307 of *The Kluwer International Series in Engineering and Computer Science*, pages 140–169. Springer US, 1995.

[17] M. Rathmair, F. Schupfer, and C. Krieg. Applied formal methods for hardware trojan detection. In *IEEE International Symposium on Circuits and Systemss, ISCAS 2014, Melbourne, Victoria, Australia, June 1-5, 2014*, pages 169–172, 2014.

[18] R. Reetz and T. Kropf. A flow graph semantics of vhdl: A basis for hardware verification with vhdl. In C. Kloos and P. Breuer, editors, *Formal Semantics for VHDL*, volume 307 of *The Kluwer International Series in Engineering and Computer Science*, pages 205–238. Springer US, 1995.

[19] A. Slobodová, J. Davis, S. Swords, and W. Hunt Jr. A flexible formal verification framework for industrial scale validation. In *Formal Methods and Models for Codesign (MEMOCODE), 2011 9th IEEE/ACM International Conference on*, pages 89–97. IEEE, 2011.

[20] S. O. Swords. A verified framework for symbolic execution in the acl2 theorem prover. 2010.

[21] M. Tehranipoor, H. Salmani, and X. Zhang. *Integrated Circuit Authentication: Hardware Trojans and Counterfeit Detection*. Springer Publishing Company, Incorporated, 2013.

[22] J. Van Tassel. An operational semantics for a subset of vhdl. In C. Kloos and P. Breuer, editors, *Formal Semantics for VHDL*, volume 307 of *The Kluwer International Series in Engineering and Computer Science*, pages 71–106. Springer US, 1995.

[23] J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald. Formal methods: Practice and experience. *ACM Comput. Surv.*, 41(4), Oct. 2009.

A Systems Approach to Cyber Security
A. Roychoudhury and Y. Liu (Eds.)

doi:10.3233/978-1-61499-744-3-37

Cyber Risk Analysis for a Smart Grid: How Smart is Smart Enough? A Multi-Armed Bandit Approach

Matthew SMITH [a,1] and Elisabeth PATÉ-CORNELL[a]
[a] *Management Science and Engineering Department, Stanford University*

Abstract. As electric sector stakeholders make the decision to upgrade traditional power grid architectures by incorporating smart grid technologies and new intelligent components, the benefits of added connectivity must be weighed against the risk of increased exposure to cyber attacks. Therefore, decision makers must ask: how smart is smart enough? We present a probabilistic risk analysis approach to this problem. Central to this approach is a new network security model based on a reformulation of the classic "multi-armed bandits" problem, where instead of projects with uncertain probabilities of success, a network defender faces network nodes that can be attacked at uncertain Poisson-distributed rates. Probing these nodes provides additional information about their vulnerability, but at a cost. Using this new technique, which by similarity we call "multi-node bandits", we compute the net marginal benefits of increased connectivity. We illustrate this model by the quantification of the overall cyber risk to the physical and informational networks of a smart grid in order to identify the optimal degree of "smartness" and the best risk management strategy.

Keywords. Smart Grid, Cyber Security, Cyber-physical, Multi-Armed Bandit, Probabilistic Risk Analysis, Systems Analysis

1. Introduction

The emergence of the smart grid promises to deliver many benefits to the overall operation of the North American electric grid, including increased efficiency, improved reliability, better incorporation of renewable energy sources, and more choice for electricity consumers [1]. However, the same technologies that improve the performance of the smart grid also expose it to digital threats such as denial of service attacks, intellectual property theft, invasion of privacy, and sabotage of critical national infrastructure [2]. Given the integrated nature of these cyber-physical systems, cyber-induced failures of the power grid can cascade to other critical infrastructure sectors such as transportation networks, water treatment, or financial systems, causing extensive physical damage and economic disruption.

While there is growing recognition across government, academia, and the private sector of the cyber vulnerability of the electric grid, the likelihood and consequences of

[1] Corresponding Author: Matthew Smith, Management Science and Engineering Department, Stanford University, Palo Alto, CA. E-mail: msmith7@stanford.edu

a cyber attack are in general difficult to quantify. Therefore, electric sector stakeholders have problems determining which investments to make beyond the minimum required for compliance with mandatory standards, and current risk management approaches are generally qualitative or heuristic in nature [3].

This paper presents a probabilistic risk analysis approach to smart grid cyber security. We develop a network security model that explores how defenders of smart grid information networks can optimally allocate their limited cyber security resources each day, including the adjustment of their strategy based on the feedback they receive from network scans and intrusion detection devices. We then show how this model can be used to identify the optimal level of connectivity where the benefits of increased incorporation of smart grid technologies is weighed against the cyber security risks these new connections entail.

Given the sequential decision nature of this formulation, we draw inspiration from multi-armed bandits, a class of problems where a decision maker must sequentially allocate resources among competing projects with uncertain probabilities of success [4]. In cyber security settings, however, network defenders are often concerned not just with the probability of a compromise, but also the *rate* at which nodes in their network are attacked. Inspired by this notion, we developed a new formulation of the multi-armed bandits model, where instead of gaming machine "arms" or projects with unknown probabilities of success, a decision maker faces unknown Poisson rates of attack against nodes in their network. We refer to this new formulation as a *multi-node bandit.*

Using this multi-node bandit network security model, we use dynamic programming to solve for the optimal network defense strategy, and based on that strategy we quantify the cyber risk to smart grid information networks. This allows us to gain insights into two research questions facing system operators:

1. How smart is smart enough?
2. How much to invest in cyber security, and where to invest it?

This paper is organized as follows. Section 2 reviews the relevant literature and gives an overview of the smart grid, including how the new systems are designed and what makes them vulnerable to cyber threats. After presenting an overview of multi-armed bandits, section 3 develops the multi-node bandit network security model, including a discussion of the assumptions and as well as the development of accurate and efficient solution methods. Section 4 illustrates the application of this security model to a hypothetical, schematic power network, and demonstrates how this approach can enable smart grid stakeholders to prioritize protection efforts given limited security resources. Section 5 discusses the modeling power of this approach, and offers concluding remarks.

2. Smart Grid Cyber Security

2.1. Motivation: Cyber Threat to the Smart Grid

The smart grid can be thought of as a modernization of existing generation, transmission, distribution, and metering infrastructures, in which existing systems have been upgraded to a digital anatomy of microprocessors, software, and network communications channels [5]. In some cases, the new technologies augment existing components that perform the same function as before, but have become “smart” by providing and communicating information about their respective task to a centralized system. In other cases, the smart grid provides completely new functionalities that allow human operators and the grid itself to react intelligently to changing conditions. The resulting architecture reflects a tight coupling of communications and power networks, and an increased interconnectedness of all the domains of the smart grid, as shown in Figure 1.

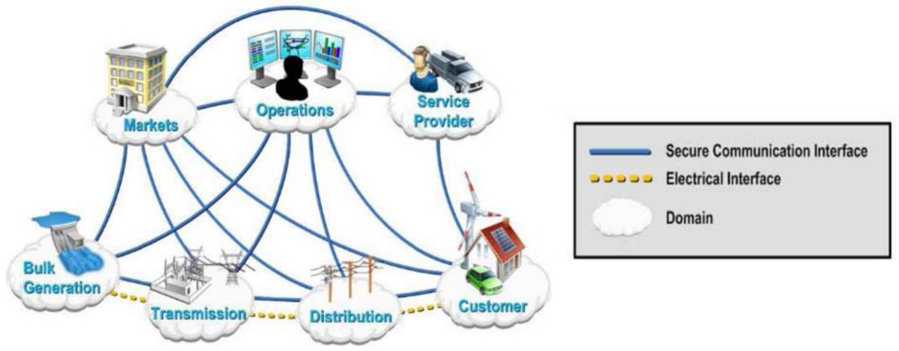


Figure 1: The coupling of communication and power networks in the smart grid. Source: EPRI [5].

In order to reap the benefits of this new level of intelligence, the smart grid involves the addition of an unprecedented level of coupling between communications and power networks, creating new potential attack vectors into power systems and therefore new security challenges for utilities, system operators, and regulators. If they could gain access to them, hackers could manipulate control systems to disrupt the flow of electricity, transmit erroneous signals to operators, block the flow of vital information, or disable protective systems.

Concerns over the threat of such a compromise are exacerbated by the widespread availability of SCADA[2]-specific hacking tools, which has created a lower barrier to entry for malicious cyber activity against the grid [6]. A growing array of threat actors, including nation states, terrorist groups, criminal organizations, disgruntled insiders, or common low-level hackers, have the potential motivation and capability to inflict various degrees of harm on power grids. Their most common motives are:

1. *Theft of Information*. There is an abundance of valuable information in the smart grid, for instance energy consumption data and valuable intellectual property. Stolen information could be used for malicious purposes, ranging from influencing energy trading to the execution of a targeted, weaponized attack.
2. *Denial/Manipulation of Service*. This could be used either as a means to sabotage the grid, or to impede the business process of the utility. Since the smart grid is built upon a foundation of real-time measurement within and between many interconnected systems, denial or manipulation of these readings can hinder the ability of the power network to perform its service of delivering power to end users, or impede the business process of the utility.

[2] SCADA (Supervisory Control And Data Acquisition) systems are computer-based control systems used to monitor and control physical processes, widely used in used in distribution systems such as water distribution networks, oil and natural gas pipelines, electric power grids, and public transportation systems.

Although to date, attacks on the US power system may have been limited to small-scale vandalism by a few individuals or small groups with limited technical sophistication, numerous reports suggest that the cyber threat is on the rise and warrants serious attention and planning. In 2014, the US Industrial Control System Cyber Emergency Response Team (ICS-CERT) reported 245 cyber incidents against industrial control systems, with 32% occurring in the energy sector [7].

2.2. How Smart is Smart Enough?

As electricity sector stakeholders make the decision to upgrade traditional power grid architectures by incorporating smart grid technologies and new intelligent components, the benefits of added connectivity must be weighed against the risk of increased exposure to cyber attacks. Decision makers must thus ask: how smart is smart enough?

To measure "smartness," we consider a physical power grid network, which consists of nodes (generators, customers, and substations), and edges (transmission lines). The system operator may then choose to connect any subset of nodes, creating an overlaid information network, as illustrated in Figure 2. Each information node enables a smart grid technology, but also becomes a potential cyber-attack vector.

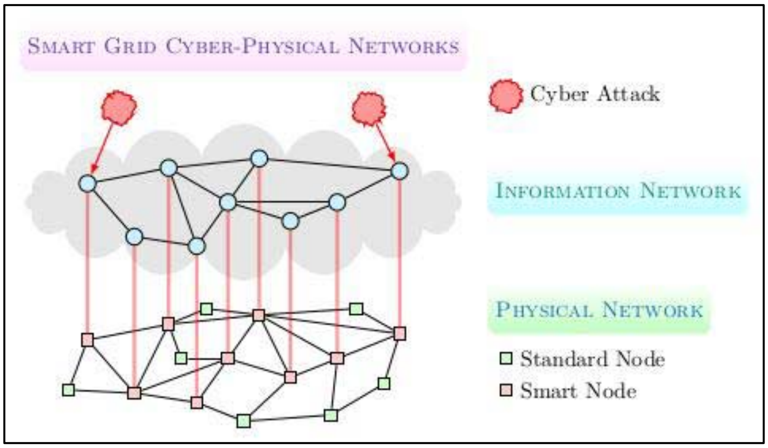


Figure 2: Smart grid cyber-physical networks.

Using this framework, we then define smartness as the number of nodes that have been connected to the information network. Therefore, smartness conceptually represents the degree to which a particular smart grid technology and its external links has been integrated into the utility's power network. Notionally, as a grid becomes smarter, we expect two things to happen:

1. **The marginal benefit will decrease**. For many smart grid technologies, most of the benefit can be achieved with a moderate level of deployment. For example, a recent report found that installing conservation voltage-reduction technology on 40 percent of distribution feeders achieves 80 percent of the benefit of upgrading all feeders [8]. Beyond that point the marginal benefit per added component decreases. A similar trend has been demonstrated for smart meters [9].

2. **The cyber security risk will increase**. While the effects of the dependency are not immediately apparent, increased connectivity should cause an increase in both the probability of cyber attacks (due to increased number of attack paths) and the consequences (due to a tighter coupling between control components in the power network).

The recent cyber-induced power outage in Ukraine on December 23, 2015 – the first known instance of a cyberattack causing a power outage – demonstrates the value of quantifying the risks and the benefits of a "smart" interconnected electric network. In this incident, three Ukrainian electricity distribution companies were the victim of a coordinated attack that included a spear phishing campaign to gain a foothold in the corporate networks; keystroke loggers to obtain credentials to access ICS networks; and remote SCADA client software to hijack human machine interfaces and issue commands to open breakers. Initial analysis suggests that these three distribution companies may have been targeted *because* they were the most connected [10]. Specifically, the level of automation and connectivity within their distribution networks enabled the attackers not only to pivot from the enterprise network to the control network, but to also gain remote control of the SCADA systems needed to open substation breakers. This enabled the attackers to disconnect 30 distribution substations and cause approximately 225,000 citizens to lose power for several hours, despite the strong cyber security countermeasures the distribution companies had in place, including a control network that was well-segmented from the business network behind a robust firewall.

2.3. Probabilistic Risk Analysis Approach

The current cyber threat to the smart grid presents unique challenges that are not satisfactorily addressed by current risk management approaches, which tend to be qualitative or empirical in nature. For example, the Australian and New Zealand Standard for Risk Management, which is used by critical infrastructure owners in those countries and mirrors many industrial standards around the world, is "not mathematically based" and has "little to say about probability, data, and models." [11]

We propose a probabilistic risk analysis (PRA) framework to smart grid cyber security. PRA is a quantitative method of risk analysis that has been applied to the safety of complex engineering systems, and is thus well suited to managing cyber risk in the smart grid. Our approach to PRA is highly interdisciplinary in nature, relying on systems analysis, probabilistic analysis, stochastic modeling, simulation, dynamic programming, human factors, economics and decision analysis to quantify risks, support decisions under uncertainty, and set priorities given limited resources.

Our PRA approach to the cyber security of a smart grid is depicted in Figure 3. In the first step, we perform systems analysis to identify classes of failure scenarios that can be induced by exploiting cyber vulnerabilities in a smart grid network, as drawn from industry expert opinion and known incidents. We then use an economic dispatch model to compute the benefit of increased smartness as well as the impact of each failure scenario, based on the physics and economics of their effect on the power system. The outputs of the economic analysis in step 2 are then inputs to the probabilistic model of step 3, which uses a Markov decision process framework to determine a strategy for the sequential allocation of cyber security resources. Finally, a decision analytic framework is used to compare the effectiveness of candidate risk mitigation strategies, from which we can compute the optimal degree of connectivity as well as other insights for smart grid system operators.

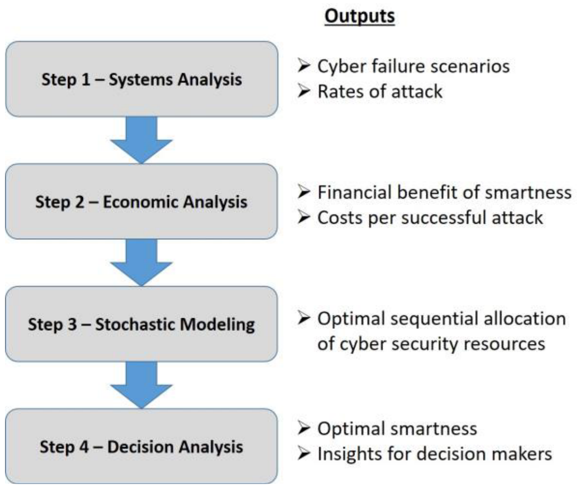

Outputs
Step 1 – Systems Analysis
➤ Cyber failure scenarios
➤ Rates of attack
Step 2 – Economic Analysis
➤ Financial benefit of smartness
➤ Costs per successful attack
Step 3 – Stochastic Modeling
➤ Optimal sequential allocation of cyber security resources
Step 4 – Decision Analysis
➤ Optimal smartness
➤ Insights for decision makers

Figure 3: Overview of probabilistic risk analysis approach to smart grid cyber security

This approach has many advantages: it provides tools to combine statistical data with expert opinion and engineering models; it also allows decision makers to conduct sensitivity analysis for key parameters, including their own risk tolerance and several other organization-specific security factors; it is well suited to handle the many levels of uncertainty introduced by the complexity of the engineering systems and the influence of human behavior; and it uses analysis to identify the focus of additional modeling efforts, thus keeping a reign on model complexity.

In addition, one of the key contributions of this work is the definition and use of a multi-node bandits (MNBs) framework, a novel Bayes-adaptive network security model that can help network defenders optimally allocate their limited cyber security resources. This model, which is described further, is used in step 3 of the analysis.

2.4. Related Work

Current efforts to analyze and implement risk management options for a smart grid and critical infrastructure in general are still in the early stages given the complexity of the task. Numerous mathematical tools and modeling disciplines have been explored as the basis for a quantitative risk model of smart grid cyber security. A small sample includes the use of attack trees/attack graphs [12], game theory [13], stochastic Petri nets [14], and network science approaches [15].

Our model focuses on the risk posed by false data injection attacks, whereby devices continues to function and appear normal, but are in fact under the control of a malicious actor. These types of attacks, which were a key aspect of the Stuxnet attack against the programmable logic controllers (PLCs) of centrifuges at Iran's Natanz nuclear enrichment facility [16], can cause operators to not take corrective actions, and could even prevent automated protective mechanisms like reclosers or automatic generation control from working properly. Research in the years following Stuxnet has focused on the risk posed by similar false data injection attack against nearly all aspects of smart grid operation, including microgrid management [17], price-directed energy utilization [18]; communication networking [19]; or against critical smart grid control components such as FACTS devices [20] or automatic generation control [21].

Despite the resurgence of research about classic multi-armed bandit (MAB) problems in recent years [4], its application to cyber security modeling is scarce. Relevant to this paper is recent work which uses a "restless multi-armed bandit" model to address the optimization of probing strategy of intrusion detection systems in a large dynamic cyber network [22], as well as work that applies MAB and reinforcement learning techniques to help a defender identify adaptive network defense strategies when knowledge about a cyber attacker is limited [23]. These efforts, however, all use the standard MAB framework to analyze situations with unknown probabilities of compromise. To the best of our knowledge, the extension of multi-armed bandit models to handling uncertain Poisson attack rates is a new contribution.

3. Multi-Node Bandits: A Bayes-Adaptive Network Security Model

3.1. Model Framework and Scope

To assess the benefits and risks of smart grid integration, we consider an electric utility that wishes to maximize its expected daily benefits. The justification for this choice is that utilities are typically the entities actually making decisions about how to invest in cyber security countermeasures, and they will only do so if they can provide the business case for their financial investment.

We consider a utility company faced with the following situation: starting from a baseline version of their power grid network, they have to decide on the degree to which they want to integrate some smart grid technology into their network. We then consider the following network defense setting. Each connected node enables a smart grid technology, but also becomes a potential cyber-attack vector, and is subject to successful cyber-attacks at an uncertain rate λ_i, as depicted in Figure 4.

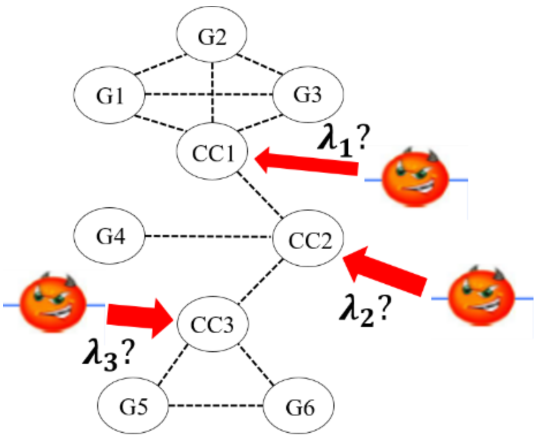


Figure 4: Network security setting where each control center (CC) is connected and subject to cyber-attacks at uncertain Poisson distributed rates. Generators (G) are not directly connected.

We then assume that a network manager has the resources to probe one node in the network per day. This action thwarts any attempted cyber attacks to that node on that day, and also provides information that the defender can use to update his belief about the uncertain rates of attack.

Given the sequential decision making nature of this formulation, this setting lends itself to a Bayes-Adaptive Markov Decision Process (BAMDP) formulation, a powerful tool for studying sequential decision problems where there is model uncertainty. Here, we use a class of BAMDPs known as multi-armed bandits.

3.2. Multi-Armed Bandits

Consider the following problem: given a fixed budget, a decision maker must sequentially allocate resources among competing projects, whose probabilities of success are unknown at the time of allocation, but which may become better understood as time passes. This describes a class of problems known as multi-armed bandits (MABs), a name which refers to the canonical example of a gambler choosing a sequence of plays on a collection of slot machine which, on average, take the gambler's money. Originally formulated in the 1930s, MABs at first proved exceptionally challenging to solve due to their computational complexity. However, with recent advances in computational power and reinforcement learning, MABs have seen a resurgence and are now used in a wide variety of applications including clinical trials, managing research projects in a large organization, and web site AB testing [4].

Taking a Bayesian approach, we can model our prior belief about each uncertain probability p_i using a Beta distribution with shape parameters α_i and β_i, as depicted in Figure 5. Due to Beta-Binomial conjugacy, our belief over the uncertain probabilities remains a sequence of Beta distributions after observing the results of pulling a given arm, but with updated parameters (if we pull the arm of machine i and that machine wins, then $\alpha_i \to \alpha_i + 1$; if arm i loses, then $\beta_i \to \beta_i + 1$). We can thus treat the sequence of belief parameters as the state of the system, e.g. $x_t = (\alpha_1, \beta_1, \alpha_2, \beta_2, \alpha_3, \beta_3)$ in the case of three arms. The Bayesian updating rules provide the system dynamics equations, i.e. the probabilities for transitioning to a new belief state as the system evolves. This effectively converts the problem into a Markov Decision Process, which we can solve using dynamic programming techniques in the case of finite time horizons, or allocation indices in the case of discounted infinite time horizon problems [4].

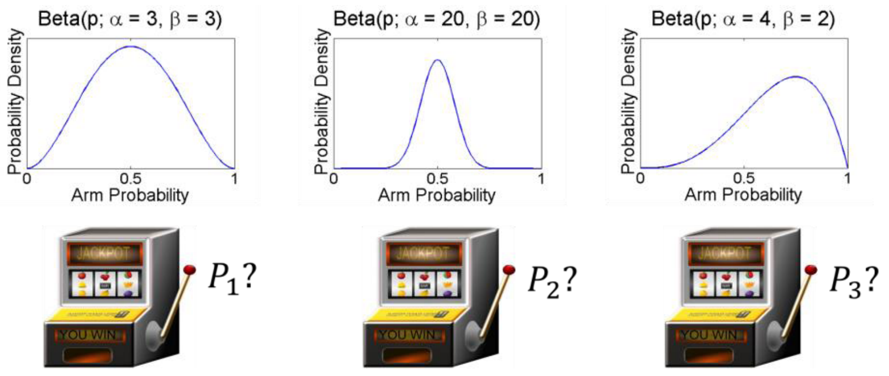


Figure 5: Bayesian approach to modeling multi-armed bandits

One of the main features of MABs is the fundamental tradeoff between exploitation (choosing the arm that we believe is best based on the information that we have gathered so far) and exploration (choosing an arm about which we are uncertain, but that may have a higher potential upside). An optimal control policy must balance these often-competing objectives based on the risk attitude of the decision maker. However, due to the complexity of the state space, which increases exponentially with the number of arms and the planning horizon, exact solutions are only possible for the simplest of problems. For more complex MAB problems, one must rely on heuristics or approximate solution methods to compute near-optimal strategies.

3.3. Multi-Node Bandits

In cyber networks, system defenders are often concerned not only with the probabilities of attack, but also the *rate (frequency)* at which nodes in their network are under attack. Based on this notion, we develop a new formulation of the multi-armed bandits model, where instead of a collection of slot machine arms or projects with uncertain probabilities of success, a network defender operates in an environment where nodes in the network are under attack with an uncertain Poisson distributed rate. To retain the Bayesian formulation, we model the prior probability distribution for each attack rate as a Gamma distribution, which is a conjugate distribution of a Poisson. A Gamma distribution for the uncertain rate λ is given by $Gamma(\lambda; \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} e^{-\beta\lambda}\lambda^{\alpha-1}$, where α is the shape parameter, β is the rate parameter, $\Gamma(\alpha)$ is the Gamma function, and the expectation is $\mathbb{E}[\lambda] = \alpha/\beta$. Due to Gamma-Poisson conjugacy, the probability distribution of an uncertain Poisson rate remains a Gamma distribution as the system evolves with new information. After observing k arrivals in time t, our posterior probability distribution for λ is given by $Gamma(\lambda; \alpha + k, \beta + t)$. Therefore, as in the standard multi-armed bandit case, we can consider our current belief parameters to be the "state" of the system.

By similarity with the original model, we refer to this new formulation as "multi-node bandits", as depicted in Figure 6.

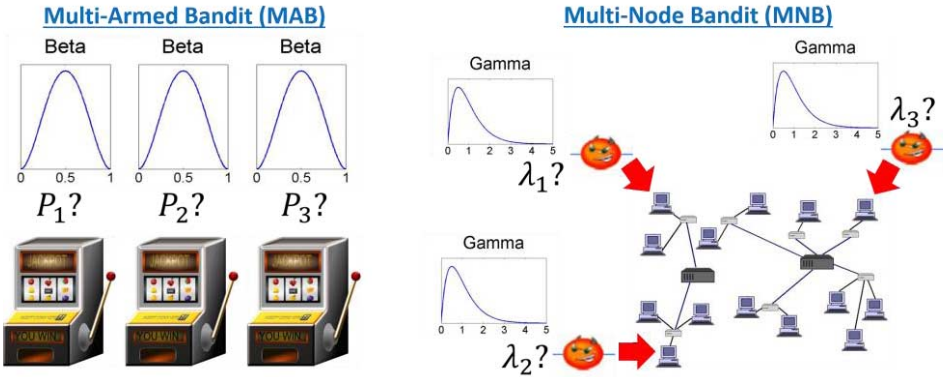


Figure 6: Conceptual representation of the multi-node bandits model

3.3.1. Basic Formulation

We apply the multi-node bandit model to cyber security settings as follows:

- A network has N nodes connected to the internet
- For node i:
  - The rate of *successful cyber-attacks* is a Poisson process with a constant uncertain rate λ_i
  - The prior probability distribution of λ_i is $Gamma(\lambda_i; \alpha_i, \beta_i)$
  - Each successful attack incurs an expected cost of c_i
- At each time period, we probe exactly 1 node. When we probe a node:
  - We observe the number of successful cyber attacks $k_i\ (t)$ to that node in that time period
  - We thwart any successful attack to that node in that time step, thus resulting in a cost savings (or a reward) of $c_i \cdot k_i\ (t)$
- The goal is to minimize the overall expected cost over the finite horizon T

As in the case of standard multi-armed bandits, there is a fundamental tradeoff between exploitation and exploration. Since probing a node thwarts any current cyber-attacks against that node, a pure exploitation policy would be to choose the node where the expected cost inflicted by past cyber-attacks, $\lambda_i c_i$, is the highest. Intuitively, this corresponds to investing security resources where the network is most heavily attacked. However, since probing a node yields information that updates our information about λ_i, we also have incentives to explore nodes about which we have less information. Once again, the optimal policy is to find a balance between exploitation and exploration.

Given the dynamic nature of cyber threats, we choose a finite time horizon. A short time horizon ($T \leq 30$) not only simplifies the computation, but also justifies the assumption that the rates of attack remain constant (though unknown) throughout the planning horizon. Other key assumptions are discussed in the next section.

3.3.2. Assumptions and Justification

We provide further discussion here on some of the key assumptions that bound the scope of our analysis and ensure that the multi-node bandit network security model remains both reasonable and tractable.

- Assumption 1: *The rate of successful cyber-attacks against connected nodes can be modeled as a set of independent Poisson processes*

Empirical analysis of cyber security incident data has shown that the arrival of successful cyber attacks are well modeled by a Poisson process. For example, analysis of recently-released security logs of 1,131 cyber intrusions against the US Department of Energy from 2010-2014 [24] shows that the distribution of inter-arrival times are exponential, and therefore arrivals follow a Poisson process, as shown in Figure 7. Additional theoretical justification comes from the Palm-Khintchine Theorem, which states that the aggregate arrivals of from many (possibly non-Poisson) sources approach a Poisson distribution in the limit.

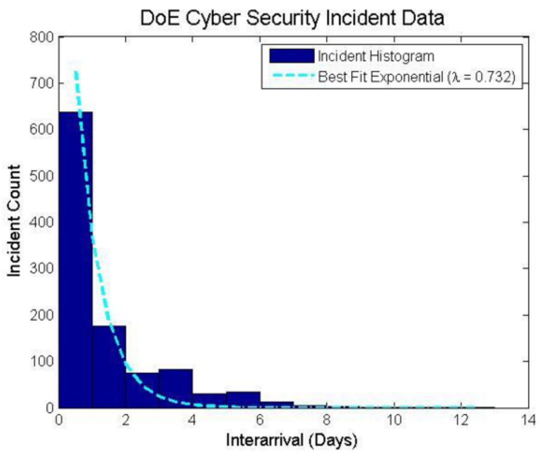


Figure 7: Analysis of cyber security logs of 1,131 total cyber intrusions against US Department of Energy (DoE) between 2010 and 2014. Inter-arrival times are well modeled by an exponential distribution. Inter-arrivals from 3 to 6 days are likely over-represented because incidents are only recorded on weekdays.

Our model also requires that the rate of attack to different nodes in the network be independent. While it is reasonable to think that these rates may be dependent (for example in the case where an attacker is launching a coordinated attack against multiple attack vectors), we justify this assumption by considering geographically dispersed networks where each connected node also has its own unique network configuration, therefore each node is relatively isolated, both physically and logically.

- Assumption 2: *The prior probability distribution of each attack rate can be modeled as a Gamma distribution*

In addition to the computational convenience of using Bayesian conjugate distributions, we can further justify the use of Gamma distributions by noting that they are a good modeling fit for cyber security settings. First, they effectively capture the "fat-tail" nature of cyber attack intensity, as Gamma distributions are *leptokurtic* (they fall to zero more slowly than a normal distribution). Additionally, the two distribution parameters α and β allow us to effectively model a wide range of prior distributions, which may be informed by historical baselines and cyber threat reporting.

- Assumption 3: *Network defenders only have the cyber security resources to probe one node per day. We assume that the organization already has a satisfactory cyber security posture in place, including standard firewall, anti-virus software, access control, and perimeter physical security measures. Hence, this daily probing represents an* extra *protection effort beyond the basic security posture.*

Our model can only apply to the allocation of cyber security resources that (1) are resource constrained, hence must be judiciously allocated in each time step, and (2) reveal information about the rate of attacks in a way that can be used to update the defender's state of information. Due to requirement (1), this model is not a good fit for Intrusion Detection Systems (IDS), which are generally applied equally to all nodes in a network, thus not subject to a sequential allocation decision. An alternative cyber security resource that can only be applied to one node per day is Endpoint Posture Assessment [25]. However, this technique only yields information about the security configuration of each endpoint, but not about the current rate of attacks. Therefore, this technique does not meet requirement (2).

A better fit is provided by the cyber security technique known as Digital Forensic Incident Response (DFIR) [26]. This technique was born out of efforts to investigate cyber security breaches after they occurred, but has since emerged as a useful proactive tool to assess the vulnerabilities and threats facing a network. DFIR is resource-intensive, requiring a team of three working a full day to complete, and therefore can only be applied to one node in the network per day. Also, because it entails a deep analysis of network logs, DFIR reveals information on the current number of attacks against that endpoint. DFIR is thus a good fit for the multi-node bandit model, and we consider in the rest of this paper the problem of a sequential allocation of DFIR teams to the different nodes of an information network.

We further assume that only attacks targeting the node currently probed can update the defender's belief. While attacks to other nodes still inflict a cost, they are not discovered unless the defender probes that node. This is justified by the nature of false-data injection attacks, whereby devices continue to function and appear normal even when currently subject to a successful cyber attack.

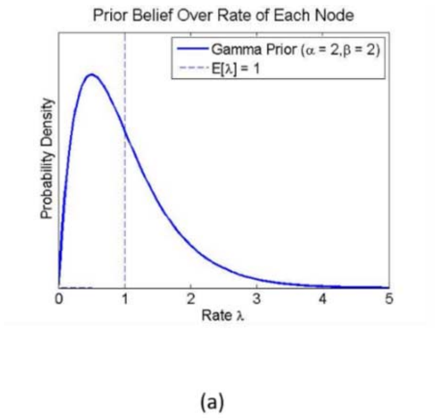
3.3.3. Solving the Multi-Node Bandits Problem

As with standard multi-armed bandits, we can solve multi-node bandits by considering the state of the system to be the current values of the Gamma distribution parameters, $x_t = (\alpha_1, \beta_1, \dots, \alpha_N, \beta_N)$. However, solving the resulting Markov Decision Process presents additional challenges beyond what was required to solve standard multi-armed bandits. First, belief updating (i.e. the probability of transitioning from one belief state to the next) is non-trivial. We must find the probability of observing k arrivals when our belief over the rate of the Poisson process is $Gamma(\lambda; \alpha, \beta)$. This can be shown to be

$$\Pr(k \text{ arrivals} \mid \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)k!} \frac{(\alpha+k-1)!}{(\beta+1)^{\alpha+k}} \tag{1}$$

Second, a more significant computational challenge is the fact that in one time unit, we can observe any integral number of attacks from a Poisson process, meaning that, in theory, we must keep track of an infinite number of transition possibilities to solve for the optimal strategy. To mitigate this, we place a cap on the number of arrivals, with m being the most arrivals we allow for a single node in a single time step, and $M = m \cdot T$ representing the most arrivals allowable to a single node in the entire time horizon. We choose m so as to balance model accuracy with computational complexity, as low values of m may cut off the high impact/low frequency events in which a node is under severe attack, but higher values of m will result in exponential increases in computational complexity.

Given these computational challenges, we used dynamic programming to solve an illustrative multi-node bandit problem with $n = 2$ nodes, a time horizon of $T = 14$, a cost per attack of $c_i = 10$ for each node, and parameters of the prior distribution of the rate of attack of each node as $\alpha_i = 2$ and $\beta_i = 2$, as shown in Figure 8a. The resulting optimal control policy results in an overall maximum reduction of the costs of attack of 173.8. The intuition of this result is that it represents the value of thwarting cyber-attacks over that time period. The results show that if we did not probe any node, cyber attacks would inflict an expected cost of 280 on our network. As illustrated in Figure 8b, if we were to use a naïve strategy and picked one node at random during every time unit, we would stop, on average, half the attacks, achieving a value (cost savings) of 140. The optimal probing strategy thus performs significantly better than the random one, saving 173.8, because it adapts the choice of node to be probed as system parameters become better understood.



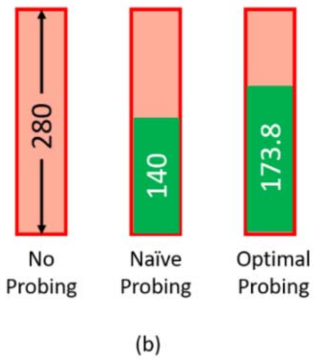


Figure 8: Illustration of the multi-node bandit problem, including (a) the prior probability distribution of the rate of attack against each node, and (b) a visual depiction of the total cost inflicted by attacks before probing, as well as the savings from random probing and optimal probing strategies.

3.4. Approximate Solution Methods

Although we are able to solve for the optimal probing policy in simple versions of the multi-node bandit problem, in practice it is rarely possible to find exact solutions for larger versions as the "curse of dimensionality" causes an exponential increase in computational complexity. Therefore, we explore in this section the effectiveness of various approximation solution methods (also referred to as exploration strategies) for the simple problem described in section 3.3.3 where the exact solution is known. We then identify the best techniques that we can apply to more complex settings.

The first class of approximate solution methods are heuristic-based exploration strategies. Based on a review of multi-armed bandit literature as well as intuition on the unique characteristics of multi-node bandits, we evaluated the solution methods described in Table 1. The parameters of each method are tuned to ensure the best match with the known optimal policy. The variance bonus method does not appear in multi-armed bandit literature, but it is intuitively a useful heuristic because when selecting a node, we have an incentive to probe nodes with more uncertainty and therefore more potential upside, and this incentive is greater as we have more remaining time.

Table 1: Heuristic-based exploration strategies for multi-node bandit problems. For each strategy, $X_i = \lambda_i c_i$, which represents the expected cost of attacks against node i.

Strategy	Choose Node u	Intuition
Greedy	$u = \underset{i}{\operatorname{argmax}}\, \mathbb{E}[X_i]$	Pick "best" node based on information so far
Boltzmann	$\Pr\{u = i\} \propto e^{E[X_i]/\tau}$ τ is Boltzmann Temperature	Start out exploratory (high temperature), then get more greedy (cooler temp)
Best Quantile	$u = \underset{i}{\operatorname{argmax}}\, F_{X_i}^{-1}(q)$ q is quantile point in CDF	Better measure of "potential upside"
Variance Bonus	$\mathrm{u} = \underset{i}{\operatorname{argmax}}\{\mathbb{E}[X_i] + \eta(t_R - 1)\cdot \mathrm{var}(X_i)\}$ t_R is time remaining; η is bonus factor	Encourages exploration of nodes with more uncertainty

While the heuristic methods attempt to circumvent the need to address the dynamic programming problem, another class of solution methods known as approximate dynamic programming attempts to solve the original dynamic programming problem by making simplifying assumptions that make it more tractable. We explore here the use of a one-step look-ahead algorithm. This means at any time, rather than performing value iterations from the terminal horizon back to the current time, we only consider the expected cost of the next transition, and use an approximate value function to estimate the remaining value until the end of the time horizon. In order to find a good approximate value function, we used the concept of Value of Perfect Information, defined as follows:

- Value of Perfect Information (VoPI) = Value we would get if we were clairvoyant and knew which node had the highest rate of attacks
- Value of No Information (VoNI) = Value we would get if we couldn't adapt as we got further information by probing each node.

Intuitively, with one time unit remaining, the best value that we can achieve is VoNI, since it would be too late to adapt after getting new information. With more time remaining, there is more time to explore, and we expect the probing value to increase toward VoPI. Indeed, the exact value of various states of information shows this pattern, as presented in Figure 9, from which we can fit an approximate value function for use in a one-step look-ahead algorithm.

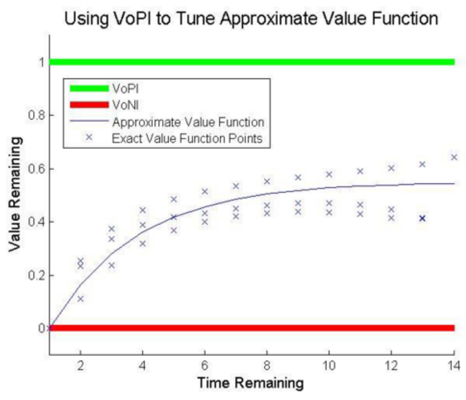


Figure 9: Use Value of Perfect Information to calibrate an approximate value function

A cross-comparison of all the approximate solution methods (the four heuristic-based strategies as well as the one-step look-ahead strategy) is shown in Figure 10. We compare methods based on three factors: (1) Accuracy (how often do we take the correct action), (2) Value of implementing that strategy, as determined from Monte Carlo simulation, and (3) Computational Speed.

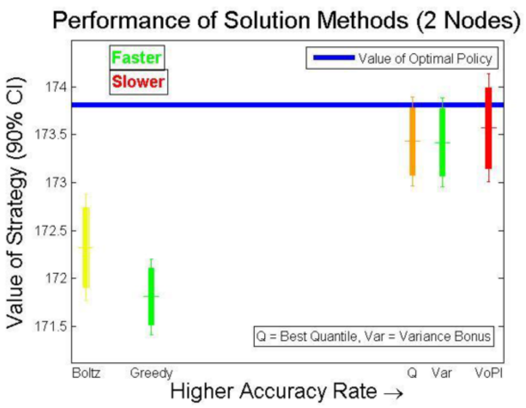


Figure 10: Comparison of approximate solution strategies for multi-node bandits

We observe that the one-step look-ahead strategy with a VoPI-based approximate value function provides the best value and accuracy, although it is the slowest one to compute. The Quantile-Selection and Variance Bonus strategies perform well, which we can attribute to the fact that they capture the "potential upside" of probing a node better than pure greedy, short-sighted strategies. In the rest of this paper, we use the VoPI method to gain insights into more complex multi-node bandit problems, although it should be noted that the variance bonus method would be especially useful in settings where a solution is needed quickly.

4. Insights to Smart Grid System Operators

We illustrate the multi-node bandit network security model and the cascading effects of failures using a schematic 16-node interconnected power network of the Western United States, as presented in [27]. As the power system becomes smarter, physical nodes are augmented with communication nodes to provide monitoring and control functions needed for a more efficient operation of the grid. In particular, we consider the incorporation of distribution grid management (DGM) technologies, one of the six principal domains of smart grid operations [28]. We use DGM because it is a key smart grid technology that enables a response to demand response (using price signals to influence energy usage) as well as the integration of renewable energy sources. Additionally, DGM technologies can be implemented in all three-node types (generators, customers, and transformers), meaning that the smartness of this network can take on any integer value from 0 nodes connected, to all 16.

4.1. Economic Dispatch Model: Risks and Benefits of Added Connectivity

Benefits – We use an economic dispatch model to compute the daily operating costs and benefits of a fictional utility in charge of this entire illustrative network. Based on recent price and energy usage data [29], an economic dispatch algorithm uses an optimal power flow (OPF) calculation to compute the cheapest way to dispatch enough generation capacity to meet the daily demand, subject to system constraints such as generator flow, transmission line flow, and stability limits. This type of calculation is used by power system operators on a daily basis. Therefore, this approach has the advantage of introducing cyber security considerations into the economic calculus that electricity providers use to make daily operational decisions. All our computations are performed using MATPOWER®, an open-source tool that operates on the MATLAB® platform and can be used to perform static and dynamic power system analyses [30]. MATPOWER® was chosen because its open design allowed us to customize simulations by directly changing system parameters in the MATLAB® files.

As DGM integration increases, the following benefits are achieved:

- The load profile flattens due to improved demand response capability, resulting in a lower *load factor* (ratio of peak to average demand)
- Potential for increased integration of renewables
- Routine failures and outages are detected and fixed more quickly

The first two of these effects are captured by the economic dispatch model. In order to capture the third effect, the dispatch model was extended to account for the random occurrence of routine outages due to natural disasters and technical failures, for which well-established baseline rates and costs can be inferred from the literature [31]. The net result is a benefit curve with decreasing marginal returns, (see Figure 11).

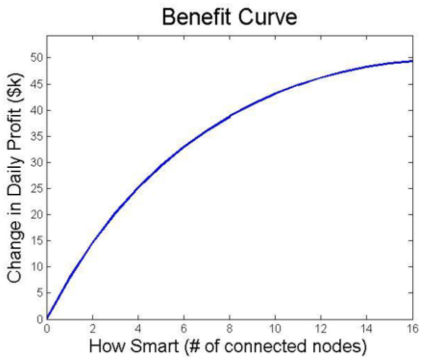


Figure 11: Benefit of increased incorporation of distributed grid management technology

Risks – To model the risks associated with increased smartness, we considered five classes of cyber security failure scenarios related to the incorporation of DGM technologies, based on those identified in a recent analysis by the National Electric Sector Cyber security Organization Resource (NESCOR) [32]. The five failure scenarios are: (FS1) Disrupted Distribution Management System Communications, (FS2) Outage Triggered Remotely, (FS3) Loss of Situational Awareness, (FS4) Demand Response Blocked, and (FS5) Customer Data Breach. The costs of the first four are computed by rerunning the economic dispatch model, subject to the additional constraints imposed by each failure scenario. The costs of a customer data breach are obtained from recent reports [33]. The average cost of all failure scenarios then becomes the average cost per successful attack to that node, c_i, which range from \$2.1k to \$2.8k.

4.2. Insights: How Smart and How Much is Enough?

The outputs of the economic dispatch model, specifically the benefit curve of Figure 11 and the costs of a successful cyber-attack on each node c_i, are used as inputs to the multi-node bandit (MNB) network security model. At each level of smartness, we use the MNB model to determine the optimal probing strategy, and the residual risk (cost inflicted by all incoming cyber attacks minus the value of savings from thwarting attacks), becomes the Cyber Risk facing the network. We set the prior probability distribution of the rate of attacks for each node as $Gamma(\lambda_i; \alpha_i = 2, \beta_i = 2)$ in accordance with baseline rates such as the DoE incident data from Figure 7, which indicates a rate of 0.73 attacks per day. Combining this risk with the benefit of increased smartness, we observe the tradeoff shown in Figure 12.

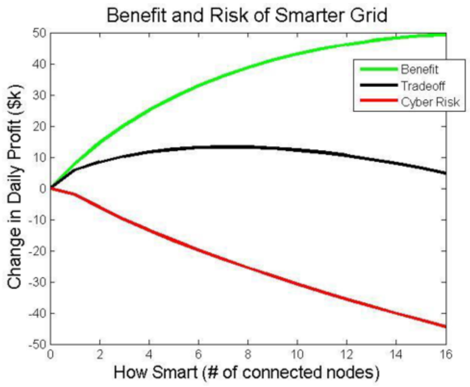
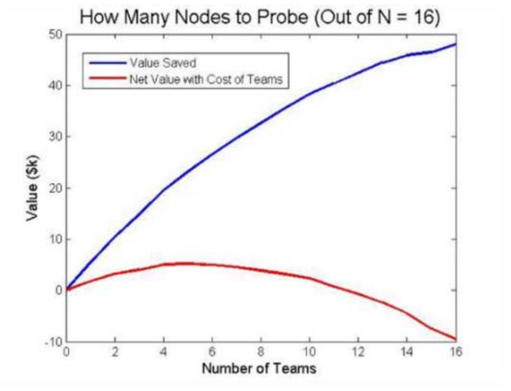


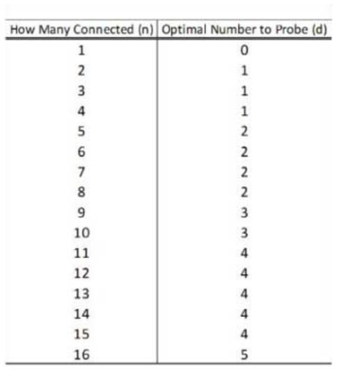
Figure 12: Tradeoff between the benefits and cyber risk of a smarter grid and optimum of "smartness"

These computations show an optimal smartness level, in this case 7 connected nodes, beyond which the marginal risk of increased connectivity outweighs the marginal benefit. The MNB network model thus allowed us to address our first research question: how smart is smart enough?

We can also use the MNB model to address our second research question: how much to invest in cyber security, and where to invest it? To do so, we consider a reformulation of the MNB model where instead of probing exactly one node per time unit, we have the option to probe d nodes per time unit, with $d \in \{0,1,\dots,n\}$ when there are n connected nodes. While hiring more teams allows a network defender to stop more cyber attacks and achieve a higher savings value, this value must be weighed against the cost of hiring more teams to perform the probing, at a cost of \$3k per team per day [34]. Applying these modified techniques to the case of all sixteen nodes connected, we find that it is optimal to hire five teams, as shown in Figure 13a. We notice that there is decreasing marginal returns on the value saved by additional teams, the intuition being that the first team, if proceeding in an optimal way, will focus on the most costly node, the second team will tend to focus on the second most costly, etc.



(a)



How Many Connected (n)	Optimal Number to Probe (d)
1	0
2	1
3	1
4	1
5	2
6	2
7	2
8	2
9	3
10	3
11	4
12	4
13	4
14	4
15	4
16	5

(b)

Figure 13: Finding how many probing teams to hire, (a) for N = 16 connected nodes, and (b) for any number of connected nodes

Finally, we can use the MNB model to combine the two research questions, where we simultaneously ask: how smart is optimum and how many nodes to probe? To do so, we consider a two-dimensional optimization problem with two decision variables: the number of nodes to connect (n), and the number of probing teams to hire (d), where $d \leq n$. The net value at each point is a combination of

1. The benefit of a smarter grid
2. The cost inflicted on the network by cyber attacks (before any probing)
3. The value saved by an optimal probing strategy using d teams
4. The cost of hiring d probing teams

In this example, we find that the optimal point is to connect $n = 8$ nodes, and hire $d = 2$ teams, as shown in Figure 14.

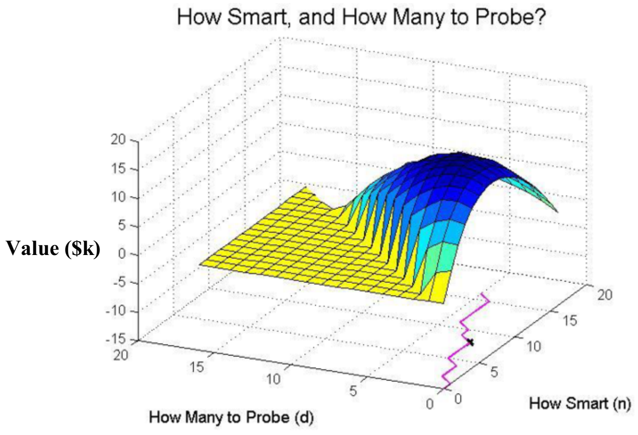


Figure 14: A 2-dimensional optimization problem showing the net value of a risk management strategy as a function of n and d. The pink trace shows the optimal number of nodes to probe for each level n. ‘x’ is the optimal point. The flat, yellow area is the infeasible region (we can’t probe more nodes than are connected).

These insights can be valuable for a wide variety of electric sector stakeholders. In addition to utilities looking to maximize expected daily profit, other entities such as market regulators, load service entities, or independent system operators can use this model to evaluate the implications of different policies. For example, if a regulator were to implement a policy which mandated that utilities implement the maximum amount of cyber security resources, we see that this can force utilities into negative profit situations (the "valley" in Figure 14) where utilities spend more on countermeasures than the value they get back.

5. Conclusions

We have presented here a probabilistic risk analysis approach to smart grid cyber security, focusing on the level of connectivity and the value of the information gained by probing for node vulnerability. This probabilistic model allows a systematic evaluation of the tradeoff between the benefit of smart grid technologies, and of increased connectivity, and the cyber security risks that these new connections entail.

The Multi-Node Bandits (MNBs) model – a new Bayes-adaptive approach to network security – provides a powerful framework to find an optimal network defense strategy given the uncertainties about the rates of cyber attack. We found that the "smartness" in the electrical grid is beneficial, but only up to a point. Above a certain level of connectivity, the marginal risk of connecting an additional node exceeds the marginal benefit of the increased smartness. We further showed that this optimum can be assessed through a risk analysis based on statistics, engineering models, and expert opinion.

Smart grid networks indeed pose unique cyber security challenges. The Congress of the United States is currently considering requiring in some cases a decrease in the grid's connectivity level, essentially unplugging parts of it from the Internet [35]. The question is: up to what point? A quantitative analysis such as that presented here can provide a more robust support to that decision than the current approaches. While it is infeasible to protect against every cyber-attack vector in systems as complex as the smart grid, this approach can help identify critical assets and enable smart grid stakeholders to prioritize protection efforts given limited security resources.

References

[1] S. Blumsack and A. Fernandez, "Ready or not, here comes the smart grid!," *Energy*, vol. 37, no. 1, pp. 61–68, Jan. 2012.

[2] A. Narayanan, "The Emerging Smart Grid: Opportunities for Increased System Reliability and Potential Security Risks," *Dissertations*, 2012.

[3] M. Hayden, C. Hebert, and S. Tierney, "Cybersecurity and the North American Electric Grid : New Policy Approaches to Address an Evolving Threat," 2014.

[4] M. O. Duff, "Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes," University of Massachusetts at Amherst, 2002.

[5] A. Lee, "Cyber Security Strategy Guidance for the Electric Sector," Palo Alto, CA, 2012.

[6] R. S. E. Knapp, *Applied Cyber Security and the Smart Grid*. Waltham, MA: Syngress, 2013.

[7] DHS ICS-CERT, "ICS-CERT Year in Review 2014," p. 21, 2014.

[8] K. P. Schneider, J. C. Fuller, F. K. Tuffner, and R. Singh, "Evaluation of Conservation Voltage Reduction (CVR) on a National Level," *Pacific Northwest Natl. Lab.*, no. July 2010, p. 114, 2010.

[9] M. G. Morgan, L. Lave, J. Apt, M. Ilic, and M. Sirbu, "The many meanings of 'Smart Grid .'" 2009.

[10] R. Lee, M. J. Assante, and T. Conway, "Analysis of the Cyber Attack on the Ukrainian Power Grid,"

2016.
[11] M. Leitch, "ISO 31000:2009 - The new international standard on risk management: Perspective," *Risk Anal.*, vol. 30, no. 6, pp. 887–892, 2010.
[12] T. Sommestad, M. Ekstedt, and L. Nordström, "Modeling security of power communication systems using defense graphs and influence diagrams," *IEEE Trans. Power Deliv.*, vol. 24, no. 4, pp. 1801–1808, 2009.
[13] Z. M. Fadlullah, Y. Nozaki, A. Takeuchi, and N. Kate, "A survey of game theoretic approaches in smart grid," *2011 Int. Conf. Wirel. Commun. Signal Process. WCSP 2011*, 2011.
[14] T. M. Chen, J. C. Sanchez-Aarnoutse, and J. Buford, "Petri net modeling of cyber-physical attacks on smart grid," *IEEE Trans. Smart Grid*, vol. 2, no. 4, pp. 741–749, 2011.
[15] M. Ouyang, L. Dueñas-Osorio, and X. Min, "A three-stage resilience analysis framework for urban infrastructure systems," *Struct. Saf.*, vol. 36–37, pp. 23–31, 2012.
[16] K. Zetter, *Countdown to Zero Day: Stuxnet and the launch of the world's first digital weapon*. Broadway Books, 2015.
[17] M. Talebi, C. Li, and Z. Qu, "Enhanced protection against false data injection by dynamically changing information structure of microgrids," *Proc. IEEE Sens. Array Multichannel Signal Process. Work.*, pp. 393–396, 2012.
[18] D. Niyato, L. Xiao, and P. Wang, "Machine-to-machine communications for home energy management system in smart grid," *Commun. Mag. IEEE*, vol. 49, no. 4, pp. 53–59, 2011.
[19] D. Kundur, X. Feng, S. Mashayekh, S. Liu, T. Zourntos, and K. L. B. Purry, "Towards modelling the impact of cyber attacks on a smart grid," *Int. J. Secur. Networks*, vol. 6, no. 1, p. 2, 2011.
[20] S. Sridhar and G. Manimaran, "Data integrity attack and its impacts on voltage control loop in power grid," *IEEE Power Energy Soc. Gen. Meet.*, pp. 0–5, 2011.
[21] Y. W. Law, T. Alpcan, M. Palaniswami, and S. Dey, "Security Games and Risk Minimization for Automatic Generation Control in Smart Grid," pp. 1–11.
[22] K. Liu, "Intrusion Detection in Resource-Constrained Cyber Networks : A Restless Multi-Armed Bandit Approach," pp. 1–14, 2012.
[23] M. Zhu, Z. Hu, and P. Liu, "Reinforcement Learning Algorithms for Adaptive Cyber Defense against Heartbleed," *Proc. First ACM Work. Mov. Target Def.*, pp. 51–58, 2014.
[24] US Department of Energy, "Freedom of Information Request #: HQ-2015-00126-F." 2015.
[25] IETF RFC 5209, "Network Endpoint Assessment (NEA): Overview and Requirements."
[26] P. Henry, J. Williams, and B. Wright, "Digital Forensics and Incident Response," *SANS*, no. July, 2013.
[27] C. Larrosa, L. Kaneshiro, J. Zhao, and M. Elisabeth Pate-Cornell, "Effect of severe space weather on cascading power grid failure: An illustrative model and policy implications," *11th Int. Probabilistic Saf. Assess. Manag. Conf. Annu. Eur. Saf. Reliab. Conf. 2012, PSAM11 ESREL 2012*, vol. 7, pp. 5253–5262, 2012.
[28] U.S. National Institute of Standards and Technology, "Guidelines for Smart Grid Cybersecurity NISTIR 7628 Revision 1," *U.S. Dep. Commer. NISTIR*, vol. 1, no. September, p. 668, 2014.
[29] "Energy Information Administration - Commercial Energy Consumption Survey," 2015. [Online]. Available: http://www.eia.gov/electricity/data.cfm.
[30] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, "MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, 2011.
[31] E. Office and P. August, "Economic Benefits of Increasing Electric Grid Resilience To Weather Outages," *Exec. Off. Pres.*, no. August, pp. 1–28, 2013.
[32] NESCOR, "Electric Sector Failure Scenarios and Impact Analyses," 2013.
[33] Ponemon Institute, "2016 Cost of Data Breach Study : Global Analysis," 2016.
[34] K. J. Soo Hoo, "How much is enough: A risk management approach to computer security," *ProQuest Diss. Theses*, no. June, p. 104-104 , 2000.
[35] The Hill, "Lawmakers look to 'dumb down' smart grid," 2015. [Online]. Available: http://thehill.com/policy/cybersecurity/286538-lawmakers-look-to-dumb-down-smart-grid.

A Systems Approach to Cyber Security
A. Roychoudhury and Y. Liu (Eds.)
© 2017 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/978-1-61499-744-3-57

A Deep Convolutional Neural Network for Anomalous Online Forum Incident Classification

Victor POMPONIU [1] and Vrizlynn L. L. THING
Cyber Security and Intelligence (CSI) Unit
Institute for Infocomm Research (I^2R)
*Agency of Science Technology and Research (A*STAR), Singapore*

Abstract. Web forums are a frequent way of sharing useful information among people. They are becoming the main source of up-to-date information and marketplaces pertaining to different domains, including criminal content and zero-day security exploits. Analyzing the web forums of the existing discussion threads is an alternative method to understand the exploits and fraud modalities a law breaker will most likely make use and how to defend against them. However, in many cases, it is hard to capture all the relevant context of the forums which is needed for classification. In this paper, we introduce a data-driven technique to mine the web forums and provide policy recommendations to the defender. A neural network (NN) is used to learn the set of features for forum classification. Furthermore, we present the evaluation and results from employing our method, with various system configurations, on real-world datasets collected form the web.

Keywords. analytics, classification, forum, cyber security intelligence, neural networks, word embedding.

1. Introduction

The Internet has become the place where the users research, purchase, socialize and learn about the world by a few clicks of a mouse or keystrokes. In this cyberspace users, leave behind rich trails of behavioral information [6] and activity intent, which can be mined by third-party trackers such as law enforcement agencies, social networking websites, and data analytic providers.

However, the web has also a less known secretive component [4] where dark users can socialize, access and share concealed services [5]. Regardless of how the information is expressed, can we extract deep insights form this alternate network? What are the most efficient techniques to analyze criminal activities, the modalities of committing them and discover who make use of this hidden networks? This part of the Internet started to enter in the spotlight of the public due to the media that raised awareness of several cases such

[1]Corresponding Author: Victor Pomponiu, Cyber Security and Intelligence Unit, Institute for Infocomm Research, Agency of Science Technology and Research, 1 Fusionopolis Way, Singapore 138632, Singapore; E-mail: v-pomponiu@i2r.a-star.edu.sg.

as releasing of NSA hacking tools [1], card cloning services [24] and online illegal drugs selling stores [2].

The most common method to systematically extract data from the websites is to use an automated program, called crawler [7]. It requires, for each website, custom parameter definitions of the elements of interest and a database to store the crawled data. The aim of security intelligence analytics is to proactively detect the pattern of the fraud activities by processing the vast amount of data (text, image. etc.) collected by the crawler.

Social network analysis [30] and machine learning are the most used tools to mine web data, with a tremendous impact on our everyday life. This new direction infused by artificial intelligence (AI) is driven by a paradigm shift in system design. Instead of learning hand-crafted features, which involves deep domain knowledge, the current approach is able to capture hierarchical feature representation automatically extracted from vast amounts of example data which leads to significant performance improvement in fields likes speech understanding, computer vision, and human language processing.

In this paper, we are investigating the feasibility of AI for security intelligence analysis of the web forums. In particular, a neural network is trained with the phrases of the post for forum classification. The classification performance of the method attains good accuracy and shows that it is able to understand the fraudulent forum posts created by manipulators.

Our contribution to the field are twofold and summarized below:

- We propose a methodology that employs a neural network to learn deep features from the forums threads which can be later used for security intelligence analysis. Furthermore, we investigate the suitability of adapting the knowledge of the models pretrained on different domains to the peculiar the domain of illicit content detection.
- Extensive experiments are carried out to illustrate that the proposed method outperforms the state-of-the-art baselines algorithms.

The remaining sections of the paper are structured as follows. Section 2 reviews web forum analysis methods, with a focus on those that are targeting those with fraudulent content. In section 3 we formalize the problem that we are addressing, while section 4 presents in detail our proposed system. Section 5 presents the experiments and results obtained. Finally, Section 6 discusses the challenges of the system and concludes the paper by highlighting several future research directions.

2. Related Works

The extraction and analysis of the web discussion forums has acquired a lot of attention and currently is an active research field. Nowadays, the available solutions share many characteristics and the only differences are the level of automation, the type of pattern recognition (whether is classification or retrieval) and the domain from where the data is collected. In this section, we give a brief overview of the recent state-of-the-art methods, while paying a particular attention to those that are devoted to detect illicit activities patterns in web forums.

The problem of searching similar threads to a given thread, have been first addressed by Singh et al. [9]. Their framework represents the thread structure via a graph model, built from forum posts, to which they incorporate heuristics that can capture the thread similarity. It is worthwhile to point out that, the model is able to cope with the issues that arise when analyzing the forums such as the drift of the post's subject and their dimensionality within the threads.

Some of the earliest works [10] on web forum classification tried to adapt the Latent Dirichlet Allocation (LDA) approach for modeling the topic of the threads. The topic distribution inferred from the contextual data was later used as a feature descriptor for thread classification.

[11] introduces a method that analyzes the forum posts [12] in order to detect patterns of terrorist activities. At the core of the method is a hybridized feature selection algorithm which takes into account the standard features used for text mining such as term frequency (TF), document frequency (DF), term frequency-inverse document frequency (TF-IDF) and entropy. After extracting the feature sets, the scheme employs a feature selection algorithm based on the union combination and symmetric difference functions. To reduce the dimensionality, these functions weighs the features according to a hybrid criterion. The experimental results performed on the Web Forum dataset confirms the ability of the proposed feature selection to identify a reduced set of discriminant features that can be used for forum classification.

Based on the naive Bayes (NB) classifier, in [13], another feature weighting approach was proposed. In a nutshell, the idea is to integrate the feature weights learned from the training data as prior information to aid in the estimation of the conditional probabilities of naive Bayes. The experimental simulations carried out on several datasets from the UCI repository show that the modified NB improves the performance when compared to other state-of-the-art NB text classifiers. However, the NB classifier is quite rigid since it assumes linearity of the model and statistical independence of the features.

Diab et al. [14] devised a system that collects information used to early identify threats from the Internet websites such, forum discussion boards and marketplaces that sell illegal goods. The system focuses more on the integration and deployment issues rather than on novel content features for forum classification. The main classification features are bag-of-words and the n-grams extracted from both the title and description of the posts (concatenated in a single feature vector). A preprocessing operation is employed to remove non-alphabetic characters and misspelled words. To cope with the insufficient labeled samples, which are difficult to obtain, the method merges supervised training with semi-supervised techniques, such as label propagation [15] and co-training[16]. During the evaluation, the model was trained and tested on 10 marketplaces, using 3 types of classifiers, i.e., NB, logistic regression and SVM.

In [17], a system targeting the deep dark web forums was devised. The paper analyzed the forums from a network perspective, gathering temporal data and further generating off-line analytics of the social network communities. In a subsequent work [18], they adopted a systematic approach to test the data mining techniques suitable for these forums.

[19] following a game theoretic approach, shows that an attack can be anticipated and, in case it occurs, the damages minimized if the cyber defense system incorporates updated information from the illicit Web marketplaces. The main assumption is that an adversary will search through the available tools and vulnerabilities in the marketplace, and therefore will initiate attacks by leveraging this knowledge. Thus, the security turns into a game, where the defender tries to predict the possible attacks by analyzing the

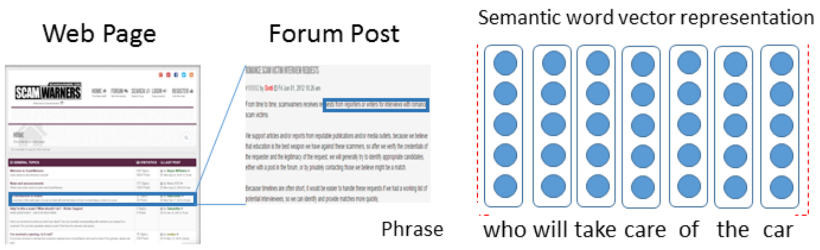

Web Page
Forum Post
Semantic word vector representation
Phrase
who will take care of the car

Figure 1. The main components of a web forum page: threads and posts. We represent the post content by a series of sentences and their words.

same base knowledge used by the attacker. In the same spirit but from a socio-economical perspective, other works [20,21] studied the ‘gadgets’ available in the hackers forums and their implied associated risk.

Other studies [22,23,24,25,26,27,28] chose a completely different approach to tackle the problem, by modeling the online communities (i.e., a social network) as a network graph [29] where each node represents a user and edges are the connections among them, weighted by a similarity feature. Using tools from social network analysis [30], these methods are able to extract, without supervision, insights about the dynamics of the social relationships, the information sharing rate, virality, the illicit content (such as cards, hacking materials etc.), user importance (centrality) within the network and the trust relationships emerged between mutually parties.

Law enforcement agencies are also leveraging on analyzing social networks at large scale to identify, track and counter criminal activities such as money laundering and human trafficking (e.g., the MEMEX program [3] run by DARPA).

3. Problem Statement

A website forum **R** consists of a set of discussion threads **T**, and in each thread, there are several posts **P**. We define a post as the smallest piece of communication generated by a user in online social networks. An illustration of the thread and post of the website is shown in **Figure 1**. Each post has embedded metadata like date and time of posting, the owner of the post etc.

Generally a thread commences with a main post (i.e., the entry post), whose title is associated with the thread title, and comprises all posts that were placed in reply to the main post. It is worth to point out that, the posts in reply to a post contains all the posts already in the thread post. For a forum thread, a graph can be constructed, in which the vertices represent the posts in the thread and edges the links between a post and all responses to it.

The goal of the study is to devise a system that identifies to which set of predefined classes **K** the posts belong. Given a set of **P** and their associated class labels l, the system construct a classifier which predicts the label of an unseen post P_i as follows:

$$\hat{l} = \text{argmax}_k F_k(P_i) \tag{1}$$

where $\hat{l}$ denotes the predicted label, F_k is the classifier and $k \in 1,\ldots,K$.

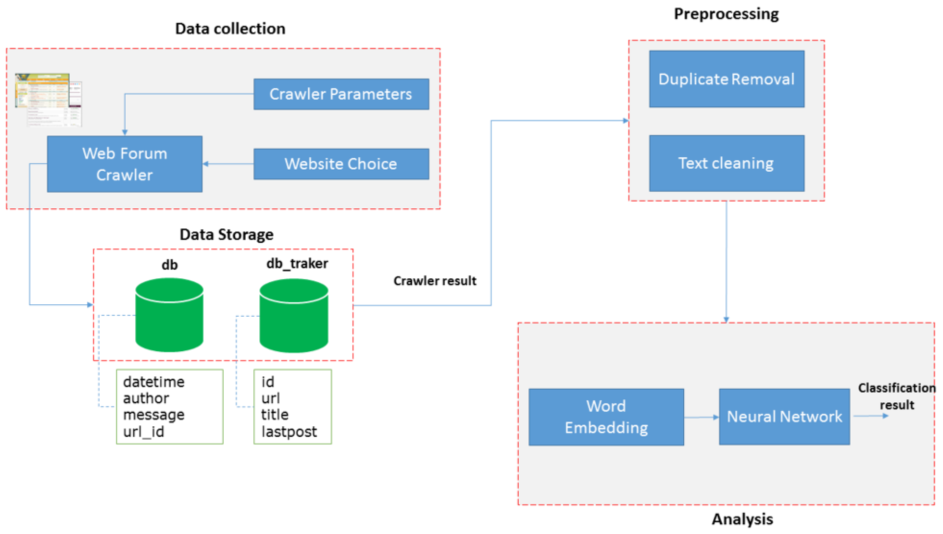


Figure 2. The flowchart of the web crawler system which incorporates the proposed analysis method using a neural network module.

4. Proposed Method

It is challenging to achieve accurate classification of web forums due to their inherent unstructured data. Instead of using hand-crafted features, we propose a modular approach to classify the forums with a combination of thread representation and a deep convolutional neural network.

More precisely, with the aid of custom web crawlers, we collect relevant information from a selection of websites and accumulate the resulting data in structured databases. Then, to simplify the processing and remove any noise from the data we, employ a preprocessing step to the crawled data. The filtered output is transformed to another representation and fed into a neural network for forum classification task.

An illustration of entire the system is depicted in **Figure 2**. In the next sections, we provide a detailed description of the main modules of the proposed system.

4.1. Data Collection

A crawler is a piece of software used to visit and retrieve websites, being a desirable tool which facilitates data collection. In order to initiate a targeted data extraction campaign from the web, we need to specify for the crawler:

1. The pool of websites from where we are interested to collect data. For each website, we do not need to define the web pages since the crawler is able to find and traverse them by using as the starting point the initial website.
2. A set of parameters which defines the elements that need to be localized and retrieved from the selected websites. It is worth pointing out, this set of parameters are custom for each website, since the structure and context of the elements vary among the websites.

3. For each website, we maintain two relational databases (*db* and *db_tracker*) to store the information such as data and time of the post, author, the entire post message, the url from where the post was retrieved, the title of the thread, and other log data.

4.2. Preprocessing

Usually, thread title and the post descriptions on web forums are flooded with unrelated text and graphics which may perturb the analysis, since they are acting as noise. To tackle this issue, we apply a text cleaning procedure to remove all the non-alphanumeric characters from the title and post. In addition, after this step, we removed all the words that are less then two characters in length.

Another problem that we encounter in processing the thread and posts was the duplicates that frequently occur on forums. To avoid any bias, we decided to remove all the duplicated from data. The last measure that we adopted was to find a suitable representation [31] of the post which is able to cope with the misspellings and word variations.

4.3. Analysis

In our system, each post is described by a set of sentences that are comprised of an ordered list of words, the smallest unit of the post. Each word is transformed to a vector of numbers, called *word vector*. A word vector denotes a word‘s meaning as it refers to other words, by means of a single array of numbers.

To embed the word in vector space representation, a shallow neural network is used which learns the context through recurrent guesses. One example of such a network is word2vec [31] - a two-layer neural net. The input of the network are words and its output is a dictionary in which each word has a vector affiliated to it. The vector which represents the words are called *neural word embeddings*, i.e., a word is mapped to a number. In the vector space generated, close semantical related words have similar vector representations.

The training is done by comparing the words, belonging to the input, against each other. Thus, the target word is predicted using the context (i.e, the *bag-of-words* technique) or another word (i.e., the *n-gram* technique). If a word context can not be predicted by the feature vector, due to reconstitution error, then its components are updated.

At the core of the model that we are using to analyze the forums is a convolutional neural network integrated into the work-flow of the system depicted in **Figure 2**.

Let’s consider $s_j \in R^m$ the j^{th}-phrase of a sentence from a forum post and m is the number of words in the phrase. Then, the sentence is represented as:

$$s_j = v_1^j \parallel v_2^j \parallel ... \parallel v_m^j \tag{2}$$

where $v_i^j \in R^n$ is the continuous word vector representation of the i^{th}-word of the phrase s_j, n is the length of the word vector, and $\parallel$ denotes the concatenation operator. This continuous feature representation of the words fits properly with the NN since the non-linear activation of the units is also continuous. We assign the label of the post to all the sentences generated.

We investigated two possible premises for the input of to the NN. In the former setting, suitable for a supervised learning mode, each word vector was randomly initialized from a Gaussian distribution, i.e., $v_i \sim \mathcal{N}(0, \sigma^2)$, stacked into a matrix and subsequently changed by the network. In the latter initialization approach, we are using a pretrained set of word vectors [31] learned in an unsupervised fashion over a large dataset. These word vectors grasp the semantic and, to a limited extent, syntactic information from their co-occurrence statistics.

A convolution in the word vector space involves employing a filter $w \in R^{n \cdot b}$ on overlapping block of words of size b to generate a new feature representation:

$$\phi_i = f(w \cdot [v_1^j \parallel v_2^j \parallel \dots \parallel v_{b-1}^j] + \beta) \tag{3}$$

where ϕ_i is the new generated feature, β is the bias term with real values and$f$ is the non-linearity function such as hyperbolic tangent. After applying the filter on all combinations of overlapping word blocks, we obtain the feature map $\phi = [\phi_1, \phi_2, \dots, \phi_{m-b+1}]$. Inspired by the convolutional neural networks applied to object detection, the models enrich the feature representation in the first convolutional layer by generating multiple feature maps using different filters on the input word vectors.

To increase the robustness of the algorithm, we are employing a max pooling operation over the feature map, that is $\hat{\phi} = max(\phi)$, to compute the final feature. Furthermore, in order to capture multiple hierarchical features, we make use of a bank of filters with various parameters, each of the filter generating its own feature map.

A fully connected network layer takes these features and combine them, and output the probability of distribution over the classes. Computing the reconstruction error at this layer will backpropagate and affect both the parameters of the NN and the vector word representations.

When learning using the scam forums, the word embeddings are adapted to characterize the intent and less the syntactics. The hierarchical structure generated by the NN aims to capture most of the information contained in the words, without the need of taking into account the syntactic constraints.

Therefore, the neural networks that we are using, has the following architecture:

1. Input layer, of size m x n, where m is the number of words in the sentence and n is dimension of the word vector.
2. Convolution layer, which filters a window of word vectors with filters of different widths.
3. Max-pooling layer, which captures the relevant features, by taking the maximal value over the feature map.
4. Fully connected layer, which computes the probability distribution over the classes.

In order to prevent overfitting, at the penultimate layer of the network, we employ regularization with dropout by constraining the l_2-norm of the weight vectors. The dropout operation randomly sets to zero (i.e., no weight) a percentage η of hidden units during forward backpropagation step. Thus, for the penultimate layer $\zeta = [\hat{\phi}_1, \hat{\phi}_2, \dots, \hat{\phi}_m]$, we modify the output y during the forward phase with

$$y = w \cdot (\zeta \circ \kappa) + \beta \tag{4}$$

where $\circ$ denotes the element-wise multiplication operator and κ is random vector with probability η being 1, which is used to cancel the contribution of the weight vectors.

Table 1. An overview of the datasets used for our experiments. We mostly collected data from English language forum related to security intelligence and general topics.

Dataset	Source	Number of samples	Target classes
Fraud	allscamsforum	1930	7
	scamwarners	2000	
	complaintboard	393	
	ripoffreport	383	
	general forums	5331	
SPAM	spam messages	5574 (with 747 spam)	2

It is worth to notice that, during the training, the network gradients are backpropagated merely via the non-zero units. At inference step, to assess the unseen samples, we will use a scaled version of the learned weight, like $\tilde{w} = \eta \cdot w$, without applying the dropout. Furthermore, after a gradient descent step, we re-scale the weight vector if the following condition is met: $\| w \|_2 = \sigma$ if $\| w \|_2 > \sigma$.

5. Evaluation and Results

5.1. Dataset

The main dataset, used in this study to evaluate the proposed scheme, called **Fraud**, consisting of approximative 13k posts, is obtained by crawling several security related forum websites containing scam, fraud and phishing information. All the webpages have undergone preprocessing and posts found in different webpages but belonging to the same thread were identified. For each thread, the crawler captured information such as the title, first post, other posts, author information and time stamp. The total number of phrases extracted from the **Fraud** dataset was 46784.

In addition, we added to this dataset 5331 forum posts collected from general websites (e.g, movie reviews [33], car forums etc.) without being directly related to any criminal content.

Beside this dataset, we also included a dataset containing a collection of email messages [32] classified into two categories. The main choice of including this dataset was driven by the assumption that there is a relationship between the scam and spam messages, since often scam messages are disguised via spam campaigns. More broadly, we are interested to explore whether the learned patterns (i.e., the semantical words vectors and the syntactics) of the model will enable to detect the illicit (or undesired) character of a sentence, paragraph or a document.

A detailed description of the datasets used for the evaluation and experiments is shown in **Table 1**.

5.2. Selection of the Parameters

For all the simulations we use the following parameters:

1. *General parameters*. We split each sentence of the post in a block of maximum length of 55 words, and the size of the word vector was set to 300. For classification

Table 2. Classification accuracy of the proposed system on the considered dataset. The $Fraud_6$ dataset considers only the post pertaining to scam forums (i.e., six classes) while $Fraud_7$ includes also the general posts. The values in bold correspond to the best performance.

Model	$Fraud_6$	$Fraud_7$
CNN-R1	0.7720	0.8008
CNN-WV1	0.7951	0.8213
CNN-WV2	**0.8073**	**0.8341**

we use 4-folds cross-validation, i.e., we split entire dataset in 4 folds that are used for training and the remaining data is used for testing the model.

2. *NN parameters*. For the neural network, we set the weight decay to 0.95, the number of epochs to 15, the units used were rectified linear (i.e., ReLU), the batch size to 50, the dropout rate to 0.5 and hidden units to 100. In addition, we use filters of size $\{3,4,5\}$ during the convolution and we rescale the l_2-norm of the weights by 9. These parameters values were selected through a grid search procedure on a subset from the **Fraud** dataset. The training of the NN is done through stochastic gradient descent using shuffled mini-batches with the Adadelta update optimization rule [34].

We classify the content of the **Fraud** dataset in 8 classes: *complaints* (posts in these class are related to complains issues), *phishing* (posts in these class are related to phishing attempts), *business scam*, *e-scam* (posts in these class are related to scams that originate from fraudulent emails), *phone-scam* (posts in these class are related to scams that originate from fraudulent phone calls), *prevention* (posts in these class are related to prevention) and *general* (posts that are related to general topics). Instead, for the **SPAM** dataset, we are using just two classes (binary classification): spam and not-spam.

In order to evaluate the classification results, we use several standard measures such as sensitivity (i.e., Sens = TP/(TP + FN)), specificity (i.e., Spec = FN/(FN + FP)) and accuracy (i.e., Acc = (Sens + Spec)/2).

5.3. Classification results

We tried different set-ups for the NN, especially by changing the input representation and the internal parameters of the net. For instance, in two of the configurations we use pretrained features vectors to initialize the neural network. The reason of adopting this approach is due to the lack of labeled data during training, and the assumption that the pretrained vectors have learned a representation that can be transferred to similar tasks.

The pretrained vectors were generated using the word2vec model trained over the Google News dataset [31]. The model has feature vector dimensionality of 300 and uses the bag-of-words technique for training. In case a word vector is not covered by the set of pretrained vectors, it will be randomly initialized with the same variance as the pretrained ones.

Therefore, the model configurations that were tested are:

- *CNN-R*1. This is the baseline method which has all the word vectors randomly initialized, and updated during the training. In this scenario, the convolutional NN model is used both for training (learning the feature vectors from the dataset) and testing (inference).

Table 3. Classification result of the proposed system against all other evaluated classifiers on the considered **SPAM** dataset. The values in bold correspond to the best performance.

Model	**SPAM**
EM	0.8554
MDL	0.9626
Boosted NB	0.9750
Linear SVM	0.9764
Linear SVM-P	0.9582
CNN-R1	0.9573
CNN-WV1	0.9769
CNN-WV2	**0.9822**

- *CNN-WV*1. The model adopted uses the pretrained vectors, and merely updates the parameters of the NN during inference.
- *CNN-WV*2. In this setting, the model starts with the set of pretrained vectors, but fine-tunes the pretrained vectors to better adapt them to the specific domain.

To estimate the generalization error of the classification models based on the selected features, we employed the 5-fold cross validation technique on the set of samples in the datasets. For 5-fold, in each trial, 4 folds are held out and used for training. The remaining fold is used for testing.

The classification results of the evaluated models are shown in **Table 2** on the **Fraud** dataset for different number of classes. The model *CNN-R*1 which is characterized by having all feature words randomly initialized has satisfactory accuracy. However, by using pretrained vectors, we achieved important performance improvements on all datasets.

For the **SPAM** dataset evaluation, which is a binary classification problem, we compared our model with well-known machine learning algorithms such linear SVM, SVM with pretrained word vectors (called linear SVM-P), Minimum Description Length (MDL), boosted NB and Expectation-Minimization (EM). Since this is an imbalanced dataset we also included among the analyzed methods the trivial rejector, as a baseline, applied to the spam class. The comparison results for the NN models against all other evaluated classifiers are presented in **Table 3**.

Although we design a simple and fast architecture, with any sophisticated pooling techniques, the model shows promising results and potential to scale to very large datasets. Overall, the experiments prove that the pretrained vectors are valuable features which can aid in better classification over the datasets. It is worth pointing out that, better performance can be obtained by adapting the pretrained vectors for the task at hand, like is done by the model *CNN-WV*2.

5.4. Discussion

The proposed method is generating word vector representation which is able to infer the contexts in which the words occur. To adapt to a specific NLP task, the neural word model fine tunes these vectors via supervised learning.

The way we represent the feature word vectors plays an important role in the overall model design. Our option for the pretrained vectors was those generated by the word2vec model trained on the Google News dataset [31].

We stress out that this model, which preserves the word order, has a limited ability is measure the interaction between the input vectors, and is weak at understanding the meaning of longer phrases.

It will be interesting to investigate also with other set of pretrained vectors trained on datasets different from the Google News dataset, which can properly acquire more complex linguistic manifestations. Furthermore, another important aspect which may affect the training is the way we initialize the words vectors which are not found within the word2vec model, by taking into consideration the distribution of pretrained vectors.

For the gradient optimization technique, we adopted the Adadelta rule, which is less offensive in the gradient update step than Adagrad, but is faster since it requires less epochs.

In addition, the employed dropout in the network layer is a good option for regularization. Finally, the NN is able to accommodate more features due to the use of a filter with various widths which generate multiple feature maps.

6. Conclusions and Future Work

In this paper, we study the problem of classifying the discussion forum threads related to fraud and other criminal activities into different classes for security intelligence gathering and analysis. Forum post classification has numerous applications such as enabling security experts to quickly filter the elements of interest from the clutter that abound forum threads, to filter harmful contents, to detect cyber-threats and discover emergent exploitation capabilities.

The core of our forum analysis method resolves around a convolutional neural network, which was tested in different configurations, and using several types of word feature vectors. Through a series of experiments, on real world datasets we prove the efficiency of our technique. These results confirms the general idea that using pretrain word vectors is an efficient asset in deep learning for various text classification tasks.

There are several research directions that we are investigating. For instance, adapt the model to extract complex linguistic phenomena from the scam forum. This weak signal could represent the well-crafted illicit intent of exploiting an innocent person, disguisedly into a forum post. To achieve this level of intelligence the model needs to go beyond the word-level, by learning vector representations of paragraphs and sentences, and their hierarchical structure.

We believe that the complexity of the illicit intent, which manifests in the forum posts, can not be full characterized by one dimensional scale. Fine-grained classification of the scams forum can help to better segregate the compositional semantic effects used by this criminals to deceive innocent people.

In the current proposal, we neglect the multimedia content that is attached to the forum threads. We believe that more rich insights can be identified by a multi-modality approach that integrates in a holistic manner both the text and the multimedia content of the forum post.

Acknowledgment

This material is based on research work supported by the Singapore National Research Foundation (NRF) under NCR Award No. NRF-2014NCR-NCR001-034.

References

[1] `https://www.washingtonpost.com/world/national-security/powerful-nsa-hacking-tools-have-been-revealed-online/2016/08/16/bce4f974-63c7-11e6-96c0-37533479f3f5_story.html`

[2] D. Bradbury. Unveiling the dark web. *Network Security* **2014(4)**, 14-17, 2014.

[3] MEMEX program. `http://opencatalog.darpa.mil/MEMEX.html`

[4] Threats Report *McAfee Lab Technical Report*. Available at `http://www.mcafee.com/hk/resources/reports/rp-quarterly-threats-may-2016.pdf`. Retrieved on August 23, 2016.

[5] T. Jordan and P. Taylor. *A sociology of hackers*. The Sociological Review, **46(4)**, 757-780, 1998.

[6] F. Roesner, T. Kohno, and D. Wetherall. Detecting and defending against third-party tracking on the web. *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (NSDI'12)*, Berkeley, CA, USA, 12-12, 2012.

[7] F. Menczer, G. Pant, and P. Srinivasan. Topical web crawlers: Evaluating adaptive algorithms. *ACM Transactions on Internet Technology (TOIT)*, **4(4)**, 378-419, 2004.

[8] J. Healey. Winning and Losing in Cyberspace, *In Proceedings of the 8th International Conference on Cyber Conflict: Cyber Power*, NATO, pages 37-51, 2016.

[9] A. Singh, P. Deepak, and D. Raghu. Retrieving similar discussion forum threads: a structure based approach. *In Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval (SIGIR '12)*, pages 135-144, 2012.

[10] X.-H. Phan, L.-M. Nguyen, and S. Horiguchi. Learning to Classify Short and Sparse Text and Web with Hidden Topics from Large-scale Data Collections. *In Proceedings of WWW Conference*, 2008.

[11] T. Sabbah, A. Selamat, Md. H. Selamat, R. Ibrahim, and H. Fujita. Hybridized term-weighting method for Dark Web classification. *Neurocomputing* **173**, 1908-1926, 2016.

[12] Y. Zhang, S. Zeng, L. Fan, Y. Dang, C. A. Larson, and H. Chen. Dark web forums portal: searching and analyzing Jihadist forums. *In Proceedings of the 2009 IEEE international conference on Intelligence and security informatics (ISI'09)*, pages 71-76, 2009.

[13] L. Jiang, C. Li, S. Wang, and L. Zhang. Deep feature weighting for naive Bayes and its application to text classification. *Engineering Applications of Artificial Intelligence* **52**, 26-39, 2016.

[14] E. Diab, A. Gunn, A. Marin, E. Mishra, V. Paliath, V. Robertson, J. Shakarian, J. Thart, A. Shakarian. Darknet, Deepnet Mining for Proactive Cybersecurity Threat Intelligence. *ArXiv e-prints*, 2016.

[15] H. Cheng, and Z. Liu. Sparsity induced similarity measure for label propagation. *In Proceedings of ICCV*, pages 317-324, 2009.

[16] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. *In Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 92-100, 1998.

[17] T. Fu, A. Abbasi, and H. Chen. A focused crawler for dark web forums. *Journal of the American Society for Information Science and Technology* **61(6)**, 1213-1231, 2010.

[18] H. Chen. *Dark web: Exploring and data mining the dark side of the web*. Springer Science and Business Media, volume 30, 2011.

[19] J. Robertson, V. Paliath, J. Shakarian, A. Thart, and P. Shakarian. Data driven game theoretic cyber threat mitigation. *In Proceedings of Innovative Applications of Artificial Intelligence Conference*, pages 4041-4046, 2016.

[20] S. Samtani, R. Chinn, and H. Chen. Exploring hacker assets in underground forums. *In Proceedings of Intelligence and Security Informatics Conference*, pages 31-36, 2015.

[21] E. Marin, A. Diab, and P. Shakarian. Product Offerings in Malicious Hacker Markets. *ArXiv e-prints*, arXiv:1607.07903, 2016.

[22] E Ferrara, W-Q Wang, O Varol, A Flammini, and A Galstyan. Predicting online extremism, content adopters, and interaction reciprocity. *SocInfo 2016: 8th International Conference on Social Informatics*, 2016

[23] C. Fachkha. Security Monitoring of the Cyber Space. *ArXiv e-prints*, arXiv:1608.01468, 2016.

[24] A. Haslebacher, J. Onaolapo, and G. Stringhini. All Your Cards Are Belong To Us: Understanding Online Carding Forums. *ArXiv e-prints*, arXiv:1607.00117, 2016.

[25] L. Le, E. Ferrara, and A. Flammini. On Predictability of Rare Events Leveraging Social Media: A Machine Learning Perspective. *In Proceedings of the 2015 ACM on Conference on Online Social Networks (COSN '15)*, pages 3-13, 2015.

[26] M. Motoyama, D. McCoy, K. Levchenko, S. Savage, and G. M. Voelker. An analysis of underground forums. *In Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 71-80, 2011. reference

[27] T. J. Holt, D. Strumsky, O. Smirnova, and M. Kilger. Examining the social networks of malware writers and hackers. *International Journal of Cyber Criminology* **6(1)**, 891-903, 2012.

[28] D. Lacey and P. M. Salmon. Its dark in there: Using systems analysis to investigate trust and engagement in dark web forums. *In Engineering Psychology and Cognitive Ergonomics, volume 9174 of Lecture Notes in Computer Science*, 117-128, 2015.

[29] A. Rajaraman, and J. D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA, 2011.

[30] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. *In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '09)*, pages 497-506, 2009.

[31] T. Mikolov, I. Sutskever, and J Dean. Distributed Representations of Words and Phrases and their Compositionality. *In Proceedings of NIPS*, 2013.

[32] T.A. Almeida, J.M. Gmez Hidalgo, and A. Yamakami. Contributions to the study of SMS Spam Filtering: New Collection and Results. *In Proceedings of the 2011 ACM Symposium on Document Engineering (ACM DOCENG'11)*, 2011.

[33] B. Pang, and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *In Proceedings of ACL*, 2005.

[34] M.D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *ArXiv e-prints*, 1212.5701, 2012.

A Systems Approach to Cyber Security
A. Roychoudhury and Y. Liu (Eds.)

doi:10.3233/978-1-61499-744-3-70

Mind the Gap: Security Analysis of Metro Platform Screen Door System

Luying Zhou[1], Huaqun Guo, Dong Li, Jun Wen Wong and Jianying Zhou
*Institute for Infocomm Research, A*STAR, Singapore*

Abstract. A smooth operation of the Platform Screen Door (PSD) system is critical to the Metro system, and any disturbance to it may disrupt the train's normal operation. The paper presents the security analysis of a Metro Supervisory Control and Data Acquisition (SCADA) system, specifically the cyber security vulnerabilities in its PSD system. The PSD system includes control subsystem and signaling subsystem, and its operation can be controlled from the moving train and the station as well. The security features of communication protocols that are employed in the SCADA system and PSD system operation mechanisms are discussed in this paper. The weak security features render the PSD vulnerable to cyber attacks. Countermeasures, from both technical and human aspects, to protect the PSD system are studied. An experiment is conducted on a testbed of simulating Metro supervisory control system to test the system vulnerabilities. The results demonstrate that the PSD control system could be compromised by an attacker who gains physical access to the control network and launches forged message or replay message attacks. A firewall cyber security countermeasure is evaluated to show that it can prevent some of the attacks but has limitations due to its rule-based mechanism. Thus, it is necessary to mind the gap for the security of metro PSD system.

Keywords. SCADA, Platform Screen Door, Vulnerability, Cyber Security, Firewall

1. Introduction

Supervisory Control and Data Acquisition (SCADA) systems are Industrial Control Systems (ICS) which are widely used for monitoring and controlling geographically distributed operations, including power plants, manufacturing and processing industries, transportation (e.g., Metro railway) systems, and so on. SCADA systems interconnect remote physical processes, monitor and collect data from remote facilities and control the physical process. Availability and integrity are the most important aspects in the ICS. Real-time access to data assets is critical to control system operations. Unavailable or interrupted control operations result in the loss of facility functions. Manipulated data by unauthorized parties causes false actions. While unlike in the Information Technology (IT) system, confidentiality plays a less important role in the SCADA system [1].

Over the past two decades, SCADA systems have evolved from closed, proprietary systems to open networks comprising common platforms running common operating systems and standard TCP/IP stacks. Presently, the system is generally connected to the

[1] Corresponding author, Institute for Infocomm Research, A*STAR, 1 Fusionopolis Way, #21-01 Connexis, Singapore 138632; E-mail: Lzhou@i2r.a-star.edu.sg.

corporate intranet which may have connections to outside networks. The open architecture and increased connectivity provide more functionalities and reduce operation costs, but they significantly increase the exposure to cyber security threats. Those threats can exploit vulnerabilities uniquely in the design and implementation of SCADA and communication protocols to attack the system or launch attacks already known in the IT world but with a higher impact on the process managed by the SCADA system [2, 3]. Attacks on availability intend to deny access to system assets as well as operations. In SCADA system, Denial of Service (DoS) refers to deny of access to the components of a system, operator workstations, and communication channel as well. Attacks on integrity modify the content of a message or the content of system assets. In the SCADA system, it refers to modify acquired data or control commands transiting through the system. SCADA communication protocols, such as Modbus, and DNP3, lack authentication features to prove their origins and sequence of network traffic [4, 5], which lead to potential attacks of forged commands or false response messages injection into a SCADA system either through direct generation of such messages or replaying messages.

In this paper, we study a specific Metro SCADA system and examine its security vulnerabilities, in particular the vulnerabilities from cyber attacks against the Platform Screen Door (PSD) system operation. The PSD doors are located along the trackside at the station, and are designed for passenger safety and efficient train operation. Any cyber attack on the PSD system to disrupt or disable its operation will result in service unavailability and train delay, causing damage to operator reputation and financial loss as well. The countermeasures to such cyber attacks can be done from both technical and human aspects. Technical approaches such as developing secure communication protocols and secure firewalls could enhance the system capacity against cyber attacks, however human factors play an equal important role in providing system security.

The remainder of this paper is organized as follows. First, the Metro SCADA system is described in Section 2, which includes the station supervisory control system, communication system and signaling system. Modbus protocol employed in the SCADA network is also introduced. In Section 3, the operation of PSD system is specifically presented and its vulnerabilities are further discussed in details. In Section 4, the security countermeasures to protect the PSD system operation against cyber attacks are proposed and discussed. In Section 5, experiments of cyber attack exploiting the SCADA system vulnerabilities and the countermeasures of preventing the attack are demonstrated and discussed. Finally, Section 6 outlines our conclusions.

2. Background of Metro SCADA System

The Metro SCADA system is composed of an Integrated Supervisory Control System (ISCS) and a train signaling system. ISCS provides facilities for integrated, centralized and localized control and supervision of electrical and mechanical subsystems remotely located at the passenger stations, power substations, depots, and tunnels. Through Communication Backbone Network (CBN) the whole Metro system can be remotely monitored, communicated and controlled from the Operation Control Centre (OCC). The signaling system supports communications between trackside controllers and trainborne controllers, and controls trackside equipment, e.g., PSD doors, and necessary mechanisms for train position localization. The standard Modbus protocol is employed in the ISCS for the information transfer among the devices.

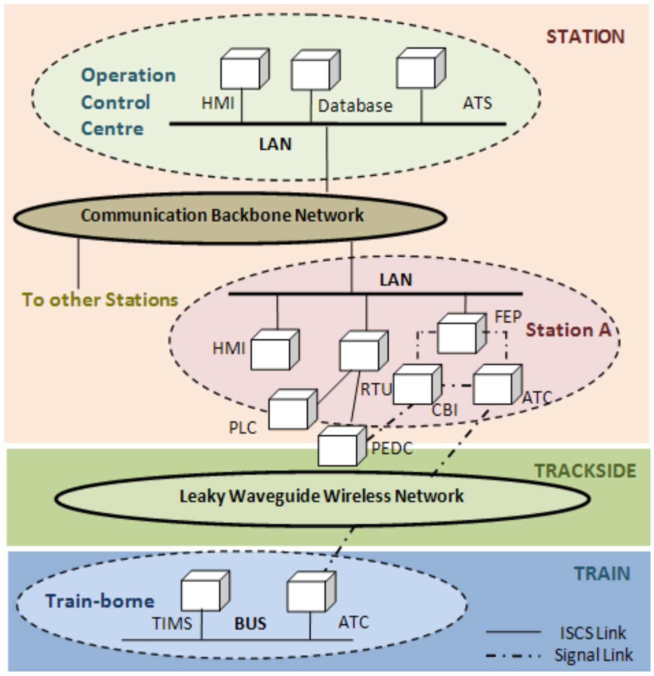


Figure 1. A typical Metro SCADA system

A typical Metro SCADA system is schematically drawn in Fig. 1. The abbreviations appear in this paper and corresponding definitions are summarized in Table 1 below.

2.1. Control and Signaling System

The main subsystems of the SCADA system, i.e., ISCS system and the signaling system, are briefly described as follows.

ISCS system

The ISCS system consists of Station Management Systems (SMS) and a Central Management System (CMS). SMS, located at each train station, handles all the monitoring and control activities within each station. It handles internal/external interface with field devices, and acquiring data from and sending commands to field devices. The field devices include controllers of PSD doors, elevators, escalators, station and tunnel lighting, and so on. Standard Modbus protocol is employed for data exchange between Remote Terminal Unit (RTU) and Programmable Logic Controller (PLC) where sensors and controllers are connected to. CMS is installed at the OCC, provides control and monitoring covering all the stations. Human Machine Interface (HMI) allows human operators to monitor the state of a process under control, and to manually override automatic control operations in the event of an emergency.

Table 1. Abbreviations and Definitions

Abbreviation	Definition	Abbreviation	Definition
ATC	Automatic Train Controller	ICS	Industrial Control System
ATS	Automatic Train Supervision	ISCS	Integrated Supervisory Control System
CAN	Controller Area Network	OCC	Operation Control Centre
CBI	Computer Based Interlocking	PEDC	Platform Edge Door Controller
CBN	Communication Backbone Network	PLC	Programmable Logic Controller
CMS	Central Management System	PSD	Platform Screen Door
DCU	Door Control Unit	RTU	Remote Terminal Unit
DoS	Denial of Service	SCADA	Supervisory Control and Data Acquisition
DPI	Deep Packet Inspection	SMS	Station Management System
FEP	Front End Processor	TDS	Training and Development System
HMI	Human Machine Interface	TIMS	Train Information Management System

Communication system

CMS and SMS subsystems have their own Local Area Networks (LAN) respectively, which may use dual 10/100 Mbps Fast Ethernet built on switches and employ TCP/IP standard protocols. OCC and all stations communicate with each other through CBN, which actually is a Wide Area Network (WAN). The communication between trainborne and trackside devices in signaling system can be done through leaky waveguide wireless channel, where microwave signals are transmitted in a trackside hollow rail.

Signaling system

The signaling system is responsible for the train communication and control. It includes a central Automatic Train Supervision (ATS) server located at OCC, trackside Automatic Train Controller (ATC) and Computer Based Interlocking (CBI) devices located at stations. The CBI device at each station also has a connection to the corresponding station LAN via a Front End Processor (FEP) and manages control signals on the trackside. Trackside ATC communicates with trainborne ATC through wireless channel, which operates in certain frequency band carrying control messages between the moving train and the station [6]. Trainborne Train Information Management system (TIMS) provides central management of control and monitoring of a range of devices in the moving train and issues instructions for emergency situation handling.

2.2. Modbus Protocol

In the Metro SCADA system, Modbus protocol is used for the communication between various field devices, e.g., RTUs and PLCs. Modbus protocol was developed for industrial automation systems, and has become a common industrial standard to transfer digital or analog I/O information, diagnostic commands, system logs, and register values between industrial control and monitoring devices. Modbus protocol based devices establish the communication by using a master-slave technique in which only one device (the master) can initiate transactions. The other devices (slaves) will respond by taking the action requested in the query, or by supplying the requested data to the master.

Modbus protocol has two variants: Modbus/RTU and Modbus/TCP. Modbus protocol starndard defines the message structure and communication procedures [7, 8]. Modbus/RTU messages are transmitted between a master and slave devices over serial lines using the ASCII or RTU transmission modes. These messages usually have four main components: slave ID, function code, application data, and an error checking field. Comparing to Modbus/RTU, Modbus/TCP or Modbus over TCP protocol provides connectivity within an Ethernet LAN-based network as well as for IP-interconnected network, enabling a Modbus master to have multiple outstanding transactions, and a Modbus slave to engage in concurrent communications with multiple masters. Modbus/TCP utilizes TCP to carry the data of the Modbus message structure between compatible devices.

From the angle of cyber security, Modbus protocol lacks mechanisms for verifying the integrity of messages sent between a master and slaves. Modbus protocol does not authenticate the master and slaves. Moreover, the protocol does not incorporate any anti-repudiation or anti-replay mechanisms. The security limitations of Modbus can be exploited by attackers to compromise industrial control systems [5].

3. PSD Operation Mechanism and Security Vulnerability

The PSD serves as a safety barrier to prevent objects or passengers from falling onto, or gaining unauthorized access to the track, and in the case of underground track, protect passengers from strong tunnel draughts accompanying an approaching train. PSD usually has a strong frameless structure that aesthetically blends with the architectural design of the stations.

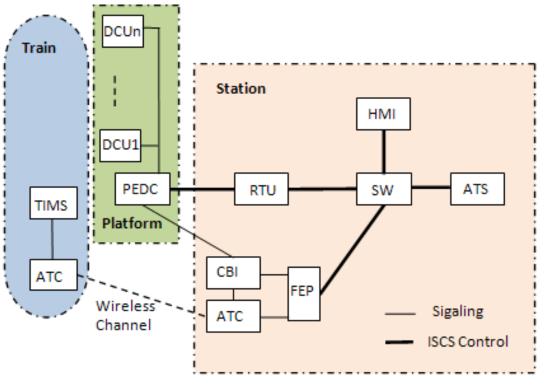


Figure 2. A typical PSD system

The PSD system is designed to provide an efficient and safe door operation, and vital circuits are hardwired and based on safety logic with fail-safe principle. This means that any single fault will not put passengers in danger. The PSD system includes two subsystems: **signaling** and **control** systems. The PSD signaling system consists of an ATC located in the train, an ATC and CBI at the station, and the Platform Edge Door Controller (PEDC) at platform, as shown in Fig. 2. The communication between ATCs is taken over wireless channel and proprietary protocols, and the channel between CBI and PEDC is hardwired serial line. The PSD control system, being a part of the ISCS, consists of in HMI, ATS, RTU and also the PEDC, and uses standard communication protocols, such as Modbus/RTU and RS485.

3.1. Platform Screen Door Operation

The train controls the open and close of the platform doors via the PSD system when it arrives and leaves the platform, in addition to simultaneously controlling the opening and closing of train doors through an internal train control system.

The vital commands, e.g., open and close the PSD doors, are communicated through the CBI in the signaling system. Other commands, such as PSD doors state setting, are transmitted in the control system. In CBI, the interlocking is an arrangement of signals and signal appliances which are interconnected such that their movements must succeed each other in proper sequence, and an action would not be performed without predefined conditions satisfied. For example, when a train is arriving at a station, the command of opening PSD doors will be issued only after the train has fully stopped and also stopped at the pre-defined position. Trainborne ATC issues open/close PSD door commands through wireless channel to ground station ATC, and through wired line to CBI for interlock checking and then to the trackside PSD controller - PEDC. PEDC executes these commands to open or close the doors via hardwired lines to the Door Control Units (DCUs). The train's operation is also affected by doors' states fed back from DCUs to PEDC and to CBI. The doors' open/close states are detected or monitored by sensors, and these can be mechanic touch sensors, and the sensed door state information is processed in CBI. Only all doors are closed or opened will the CBI acknowledge the state of the accomplished command to the train [9, 10, 11].

PSD doors can also be controlled/monitored through PSD control system, i.e., through ISCS. The main functions performed by the PSD control system are to collect platform door state information, and enable/disable the door operation. ATS in the ISCS can issue PSD open/close commands, but the commands have to pass through the signaling system via FEP and to CBI. The data collecting and state setting commands are transmitted in ISCS among HMI, the RTU and the PEDC. E.g., when being informed that certain train doors are inoperative, the ISCS sends state setting commands to PEDC via RTU to disable the corresponding PSD doors. PEDC links with DCUs via Controller Area Network (CAN) bus to communicate control system commands received by the PEDC from the ISCS to the DCUs, and status reports from DCUs to the PEDC and en route to the ISCS.

3.2. PSD System Security Vulnerabilities

The availability and reliability of the PSD system are directly relevant to the train's smooth operation. Any disturbance to the operation of the PSD system will disrupt the train's schedules, e.g., failure to open the PSD doors will block passengers from exiting or entering the train. Although such disruption may not jeopardize the safety of passengers due to the fail-safe mechanisms enforced in the system, it affects train service availability, results in revenue loss, and damages operator's reputation.

The normal PSD operation could be disrupted by cyber attacks, such as forged message injection, and message replay attacks. The PSD signaling system employs closed and proprietary communication protocols and interfaces, which are secretly kept and is thus harder for the cyber attacker to penetrate to the system. However, it is different for the ISCS system, where openly accessible standard protocols and interfaces are employed. With known protocols and message format, and compounded with weak security protections, the ISCS is more vulnerable to cyber attacks. In the following sub-sessions, the PSD control system's security vulnerabilities are analyzed.

Forged Message Attack

In the Modbus data units there is no command sequence number. The master device associates the first response received with its command, and responses which are not associated with a command will be discarded by the master. There is no digital signature or authentication mechanism in the Modbus data units to allow the master to confirm the response is from the addressed slave. An attacker could insert a response to a command sent to another slave before the addressed slave can respond. This vulnerability allows an attacker to eavesdrop on Modbus communications and forge a response to a command before the addressed slave responds. For example, a forged response to a state collecting command would report false PSD doors state information which causes the operator to make wrong assessment and take wrong action.

Message Replay Attack

Due to the lack of message authentication and integrity checking, the message receiver would fail to identify the replaying messages. The control commands or responses to/from the PSD system could be captured over the transmission process and later simply replayed by an attacker. The replayed control command could set wrong PSD door states, and the replayed response messages could lead the operator to make wrong control decision. Besides the PSD system, other electrical and mechanical subsystems in the Metro system are controlled and monitored by operators via the communication networks in ISCS, and they are vulnerable to the message reply attacks due to the weak security measures in the ISCS system.

Human Factors

State-of-the-art technology cannot secure the Metro system alone, minimizing human error is even more crucial. Errors from operators —violations of standard procedures, wrong configured settings, lack of vigilant and prompt investigation of suspicious communications —open the door to successful attacks. For example, using a personal USB thumb drive on the ISCS equipment may introduce malware, virus into the system that compromises the PSD system, and not strictly following the procedure when operating the system or handling any anomaly could let the attacker bypass the security wall as well. Human factors play an important role in securing the Metro system, since the system is still vulnerable to human error even with strong security measures enforced.

4. Countermeasures to PSD System Attacks

The PSD system vulnerabilities could be exploited by adversary to disrupt the system operation. We study the available security approaches to counter the attack, and to secure the PSD system.

4.1. Secure Modbus protocol

To counter the message modification and replay attacks due to the lack of authentication controls in the protocol, i.e., to identify and prevent forged messages from an attacker, one approach is to develop and apply a secure industrial protocol, e.g., secure Modbus protocols [12, 13]. The secure protocol implements an authentication mechanism that provides a verifiable and secure way of indentifying the source of all data transmission. The authentication feature can be enhanced by employing symmetric cryptography and hashed message authentication codes. In the protocol both sender and receiver share a common secrete key from which session key is generated. The approach is an ideal one as it solves the problems at their origin, but to implement such a secure industrial communication protocol faces challenges because of the introduced significant changes to the long lifecycle legacy SCADA system and configuration.

4.2. SCADA Firewall

An approach that effectively addresses the forged message injection attack issues while without significant changes to the legacy system is to deploy SCADA firewalls. A SCADA firewall can be an independent device that locates between the sender and receiver and examines the packets passing through it. It filters the packets based on pre-defined rules, e.g., on IP address, protocols, port numbers and Modbus function codes. It could filter out the packets with irregular formats or unspecified instruction [14, 15]. However a pure firewall could not block the replaying packets and forged packets which have normal formats and contents, and comply with the pre-defined rules, as it does not perform message authentication and message sequence checking.

4.3. Security Gateway

The PSD system does not demand very stringent reaction time to the door opening/closing action. As the default time to execute the opening/closing commands are in a few seconds [9], an extra time in the milliseconds for processing security schemes would be acceptable.

The approach of deploying a security gateway could enforce the security measures such as message authentication and integrity. The security gateway, located before the field devices, e.g., before RTU or PLC as they generally interface with multiple field devices, can be an independent device that has the computation power and storage capacity to perform key management, authentication and message integrity functions in the place of RTU or PLC. A secure channel could be setup between the HMI and the gateway. There is no change required to RTU or PLC, but one of drawbacks of this approach is the extra processing time introduced in implementing the enhanced security features, which may not be feasible for time critical control process. As pointed out above, the PSD system can tolerate milliseconds delay, thus the gateway approach can provide a secure communication channel from the server and HMI up to the front of RTU or PLC.

4.4. Human Factor

Despite all the countermeasures enforced in the technique domain, an error contributed to human factors could break up the defense line, resulting in the PSD operation disruption. The importance of technology, procedures and people must be emphasized in equal order. The countermeasures to prevent human errors should involve the awareness education of the human vulnerabilities, commitment to operational principles, compliance to operation standards, and clear communication of responsibility and accountability.

5. Experiment Study

Experiments are conducted to demonstrate the cyber attacks to the ISCS system and evaluate the feasible countermeasures to prevent such attacks. We present the experiments in which an attacker exploits security vulnerabilities of the protocol employed in the ISCS system and launches message replay and forged message attacks, and these attacks could compromise the PSD system and other systems related to station equipment operation.

The experiments are conducted on a Training and Development System (TDS) -- a testbed of the ISCS system of a Metro system, which is built to test new software and system, and train new staff. The experiments could not be conducted on the real Metro system, for fear of disturbing its operation. The hardware and software configuration and operation of TDS are similar to those of the ISCS system in a train station. The hardware and software components include HMI, RTU, Ethernet switch, HMI software and Modbus protocol. The TDS is a small scale implementation of the ISCS system, so the experiments of cyber attacks and countermeasures on the TDS are equally applicable to the real operating Metro system, without facing the risk of interfering with train revenue-producing operation. There are some cases of SCADA testbed, but not Metro SCADA testbed, can be used by us as reference [16]. Our experiments assume that attacker can gain physical access to the ISCS network, e.g., access a switch by an insider or a visitor in the control or station room. The testbed and experiment setup is shown in Fig. 3.

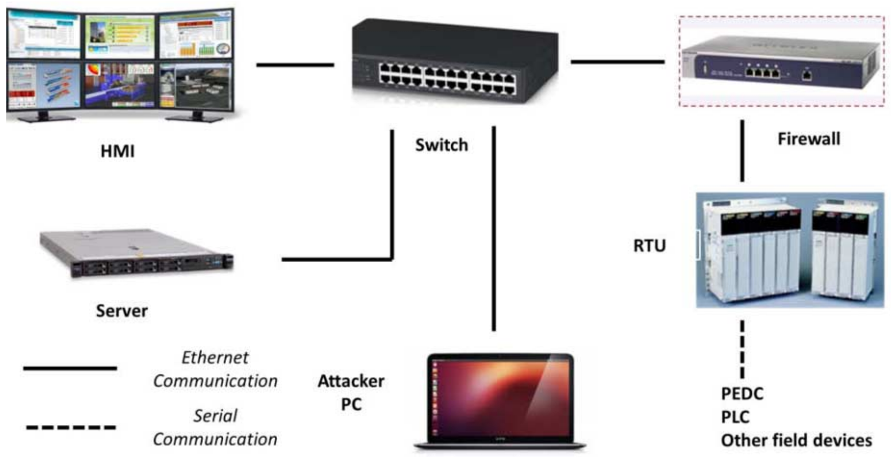


Figure 3. Testbed and experiment setup

The testbed consists of a HMI, a server, and a RTU connected by an Ethernet switch. An attacker's laptop PC is connected to the switch. A firewall device is later inserted in the testbed between the switch and RTU to counter the cyber attacks. The HMI is a software application and an important part of ISCS system that presents the real-time information to an operator about the states of multiple processes, and to implement various control instructions. For example, as shown in Figure 1, an operator sits in a train station can use the HMI to remotely monitor the real-time conditions of field devices. The operator can remotely retrieve RTU status information, read register values of certain PLC, control station lighting systems, open or close station power systems, disable one or multiple platform screen doors, and many more. Typically, the above information is displayed in a graphic format or Graphic User Interface (GUI). Meanwhile, the operator can also use pre-defined buttons and hyperlinks in the HMI to implement commands to control the above systems.

The RTU provides intelligence in the field, and allows the operator or central SCADA server to communicate with the field instruments. Its function is to control process equipment at the remote site, and acquire data from the equipment. A RTU may be interfaced to multiple servers and field devices with different communication media. In the typical Metro implementation, RTU connects PEDC and multiple PLCs with RS485 serial interface, and the commands to field devices will go through RTU and be relayed to the field devices. If forged messages could be accepted by RTU, then they would be accepted as well by the connected field devices, as over the RS485 line and in the field devices there is no cyber security mechanism to examine whether a seemingly normal message is an authentic one.

Various cyber-attack techniques can be applied to attack a SCADA system, such as Distributed Denial of Service (DDoS) attack, Man in-the-middle attack, forged message attack, virus and Trojans, and social engineering. In the testbed environment, to test the attack on a certain subsystem without affecting others, forged message attack or packet replay attack technique is selected. The experiments demonstrate that control commands to the field devices could be captured, modified or replayed by an attacker, and be injected into the ISCS, and be accepted and processed by the RTU. In the following, due to the project confidentiality requirements, some descriptions of raw data and specific protocols employed in the SCADA system are not disclosed.

5.1. Replay/Forged Message Attack

The objective of replay attack is to control the field devices in SCADA system by sending "legal" commands. However, this "legal" only indicates the packet frame, because the attacker can generate the command packets by using his/her knowledge of the industrial protocol. Both the original command and forged command can be accepted and executed by SCADA device, such as RTU and PLC. The difference is that the forged command is sent to the RTU at a wrong time. For example, the attacker sends a forged command to shut down the station lighting during peak hours.

To successfully send a forged single packet that can be accepted by RTU and the field device, the process can be divided into the following stages:

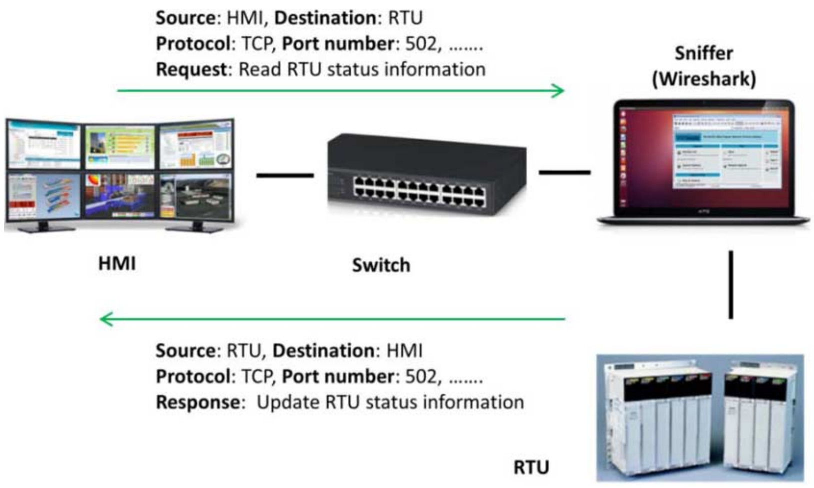
- Packets sniffing
- Packets analyzing
- Packets impersonating

Packets sniffing

We assume that an attacker can find a way to physically connect his/her attacking tools into the SCADA network. This assumption is necessary if the SCADA network is a closed network without connecting itself to public Internet. One of the sniffing methods has been described below.

The attacker connects a device, such as a Linux Ubuntu OS laptop, to the switch located in the middle of communication channel between the HMI and RTU, and sniffs the control commands destined to field devices. We use a Cisco 2960 switch, which is a popular module and employed in many real systems. For example, the attacker can connect his/her laptop into an empty port of station switch, and configure that port as a mirror port. Hence, all the network traffic between HMI and RTU will be copied into the attacker's laptop. This step is easy for insider attack. To capture and view the packets on the laptop, the attacker can install Wireshark and make sure that the network card is running on promiscuous mode. As the network traffic is not encrypted, all network traffic is in plain text and readable by the attacker's sniffer. The attacker eavesdrops and records the network traffic and then further analyses the control command for its corresponding functions and contents.

If the switch administrator sets a password to log in the switch configuration mode, the attacker may face certain challenges when he/she tries to configure the mirror port. The attacker can use other hacking techniques such as Address Resolution Protocol (ARP) spoofing. Another easy option is to unplug the RTU Ethernet cable from the switch, and plug attacker's laptop in between. In such case, the laptop works in network bridge and sniffer mode. In reality, the attacker also needs to pay attention on the security mechanisms employed in the SCADA system, such as Intrusion Dedection System, which may alert the administrator any irregular operations. However many legacy SCADA systems do not employ good protection mechanism as they were built in more than 20 years ago.


Source: HMI, **Destination**: RTU
Protocol: TCP, **Port number**: 502, ...
Request: Read RTU status information
Sniffer (Wireshark)
HMI
Switch
Source: RTU, **Destination**: HMI
Protocol: TCP, **Port number**: 502, ...
Response: Update RTU status information
RTU
Figure 4. Packets sniffing

Packets analyzing

The next step is to perform a deep analysis of the captured packets. The attacker tries to find the packets that carrying important SCADA control instructions. Only after gathering sufficient knowledge about the network traffic and the control command functions, the attacker can further forge those control instructions according to their targeting objectives. As described above, the field devices are connected to RTU through serial communication. A control command is always transmitted over TCP/IP to the RTU first. Modbus/TCP (TCP port 502) is the most common industrial protocol to transmit such commands. Then the RTU plays as a protocol converter to re-package the packet payload based on serial communication standard, and sends the command to its destination. To recognize the control instruction related packets, the attacker can use the following characteristics to match:

- Protocol -- Typically, most SCADA commands will be transmitted over Ethernet by TCP.
- Port number -- Typically, each control command will be assigned a unique and static TCP port number. Some commands will use standard Modbus/TCP, which is TCP port 502.
- Packet timing -- Based on traffic volume, more than 90 percent of packets are periodic, because they are used to keep the SCADA system alive. The control commands may appear non-periodic.
- Other packet details -- For a TCP/IP packet, a lot of information can be used to identify itself, such as MAC/IP addresses, TCP flags, SEQ/ACK numbers, and payload length.

Once control instruction related packets are identified, the attacker needs to analyze its payload frame. This is also a critical but time-consuming reverse-engineering step. The complexity and possibility of payload frame may vary depending on the controlled functions and scale of on-site devices. If the SCADA system utilizes standard industrial protocol such as Modbus, most of them will be exactly same or very similar to standard Modbus payload frame. Statistics is the most direct and efficient methodology to analyze and summarize the payload frame so far. Hence, a successful payload analysis needs enough sniffed packets for each operation.

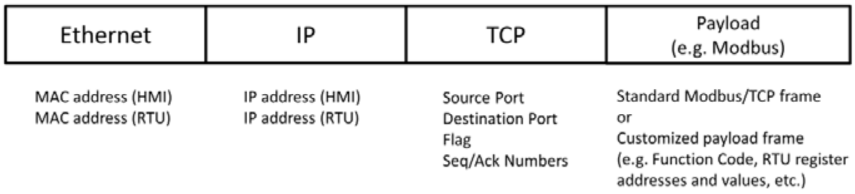


Figure 5. Standard Modbus/TCP packet frame

Packet impersonating

Let assume the attacker has already identify a single packet that carrying the instruction to disable the PSD system, then he/she can forge it and re-send it to RTU at any time he/she wants. If the packet were sent during peak hour, the passengers could be blocked from entering or leaving the train.

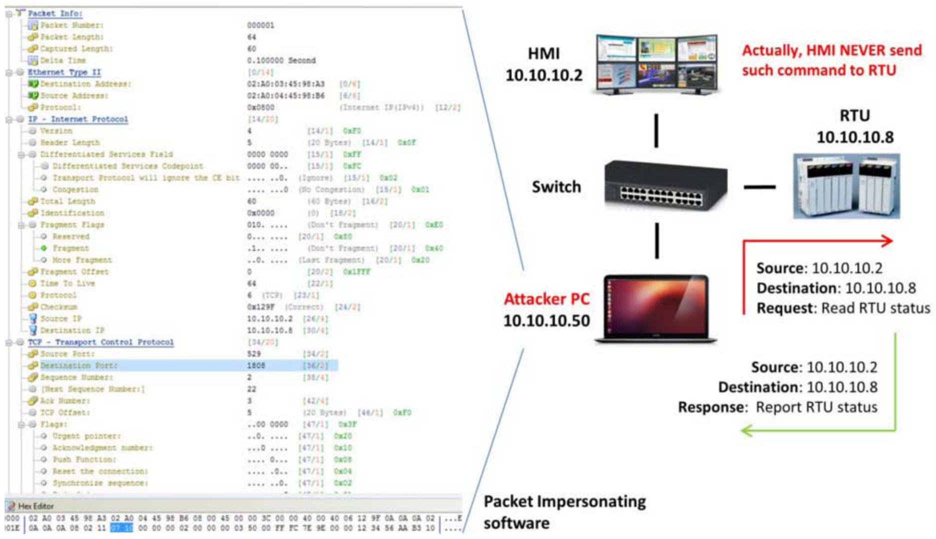
As shown in Fig. 5, the attacker can forge the data part based on original packets. Some examples are shown below. All of them may disrupt the normal operation of PSD or other systems behind RTU.

- Change the function code of Modbus
- Read the register values of RTU
- Write new/invalid values into RTU registers
- Remote restart PSD devices
- Clear device memory, reset counters in memory

There are many available packet generators can be used to simulate legal SCADA traffic as well as generate forged packets for attacking purpose, such as

- Colasoft Packet Builder (Windows OS) [17], an example shown in Fig.6.
- PackEth (Linux Ubuntu OS) [18]

The forged packets can be sent out from any laptop that is connected to the switch (Fig. 3). The attacker could further modify the packet payload or the packet header such as IP address and MAC address of legitimate sender, and fabricate and send a false command to RTU. The forged command could be accepted by RTU without distinguishing whether it is an authentic original command or a replay command. The RTU could execute a forged command that requests state information without realizing the ongoing attack and report its state information in a response packet.



```
Packet Info:
  Packet Number:                                000001
  Packet Length:                                64
  Captured Length:                              60
  Delta Time                                    0.100000 Second
Ethernet Type II                                [0/14]
  Destination Address:                          02:A0:03:45:98:A3   [0/6]
  Source Address:                               02:A0:04:45:98:B6   [6/6]
  Protocol:                                     0x0800              (Internet IP(IPv4))  [12/2]
IP - Internet Protocol                          [14/20]
  Version                                       4             [14/1]  0xF0
  Header Length                                 5             (20 Bytes)  [14/1]  0x0F
  Differentiated Services Field                 0000 0000     [15/1]  0xFF
    Differentiated Services Codepoint           0000 00..     [15/1]  0xFC
    Transport Protocol will ignore the CE bit   .... ..0.  (Ignore)  [15/1]  0x02
    Congestion                                  .... ...0  (No Congestion)  [15/1]  0x01
  Total Length                                  40            (40 Bytes)  [16/2]
  Identification                                0x0000        (0)  [18/2]
  Fragment Flags                                010. ....     (Don't Fragment)  [20/1]  0xE0
    Reserved                                    0... ....  [20/1]  0x80
    Fragment                                    .1.. ....     (Don't Fragment)  [20/1]  0x40
    More Fragment                               ..0. ....  (Last Fragment)  [20/1]  0x20
  Fragment Offset                               0             [20/2]  0x1FFF
  Time To Live                                  64            [22/1]
  Protocol                                      6  (TCP)  [23/1]
  Checksum                                      0x129F  (Correct)  [24/2]
  Source IP                                     10.10.10.2  [26/4]
  Destination IP                                10.10.10.8  [30/4]
TCP - Transport Control Protocol                [34/20]
  Source Port:                                  529          [34/2]
  Destination Port:                             1808         [36/2]
  Sequence Number:                              2            [38/4]
  [Next Sequence Number:]                       22
  Ack Number:                                   3            [42/4]
  TCP Offset:                                   5            (20 Bytes)  [46/1]  0xF0
  Flags:                                        ..00 0000    [47/1]  0x3F
    Urgent pointer:                             ..0. ....  [47/1]  0x20
    Acknowledgment number:                      ...0 ....  [47/1]  0x10
    Push Function:                              .... 0...  [47/1]  0x08
    Reset the connection:                       .... .0..  [47/1]  0x04
    Synchronize sequence:                       .... ..0.  [47/1]  0x02

Hex Editor
000 | 02 A0 03 45 98 A3 02 A0 04 45 98 B6 08 00 45 00 00 3C 00 00 40 00 40 06 12 9F 0A 0A 0A 02 | ...E
01E | 0A 0A 0A 08 02 11 07 10 00 00 00 02 00 00 00 03 50 00 FF FC 7E 9E 00 00 12 34 56 AA B3 10 | ....
```

Figure 6. Customized packets generator (Colasoft) in Windows OS

5.2. Firewall Countermeasure

To protect the SCADA system against cyber attack, a firewall approach is evaluated. A firewall is a network security device that monitors and controls network traffic passing through it based on predefined security rules, and it incurs less interference to the legacy system than other approaches, such as upgrading communication protocols and installing security gateways. In the experiment, a specific firewall device is chosen [14], which has features such as plug-n-protect installation, no requirement on network configuration and routing table's change, and no disruption to the existing control system. It can perform deep packet inspection (DPI), which inspects deeper into the application content fields of a packet, specifically to common industrial protocols, to examine command message structure or Modbus function code and determine whether the intended action makes sense. The DPI functions of the commercial firewall product do not work for non-standard proprietary protocols, which unfortunately is the case of the system under study. The firewall is placed between the switch and the RTU to filter traffic comes into or from the RTU, illustrated in Fig.3. In the experiment, the firewall can successfully block the forged packets if the attacker uses an IP address and a port number that are not existed in the pre-defined firewall rules. In a SCADA firewall, all existed legal network devices must be recorded in its rule list. Hence, the threats from unknown devices can be reduced but not eliminated. In Fig. 7, Rule 1 and Rule 2 are very basic examples to put legal SCADA field devices into whitelist. On the other hand, each SCADA command is always assigned a static TCP port number. Those port numbers will be in standby status and wait for incoming commands. All other unrelated ports are suggested to be put into blacklist of firewall. Hence, the attacker cannot invade through those ports. Most SCADA commercial firewall products are able to achieve above functions. But it does not mean perfect protection at all.

If the attacker can generate forged packets with IP addresses and port numbers which are allowed by the firewall rules, then the firewall may failed to block such packets. As described in Section 5.1, Fig. 6 and Fig. 7, a 3-step forged packet method was applied during the experiment to send attacking packets and pass the firewall successfully. Besides IP addresses and port numbers, the attacker can also calculate the correct Seq/Ack number, checksum and flags. This is relatively easy work for experienced attacker.

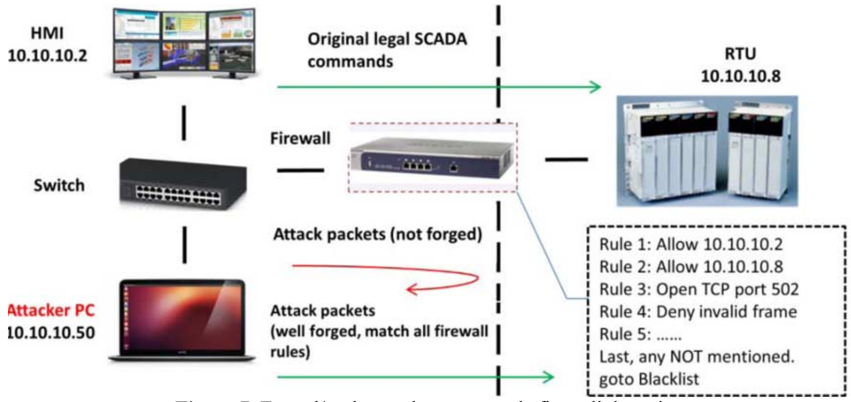

HMI
10.10.10.2
Original legal SCADA commands
RTU
10.10.10.8
Firewall
Switch
Attack packets (not forged)
Attacker PC
10.10.10.50
Attack packets (well forged, match all firewall rules)
Rule 1: Allow 10.10.10.2
Rule 2: Allow 10.10.10.8
Rule 3: Open TCP port 502
Rule 4: Deny invalid frame
Rule 5: ......
Last, any NOT mentioned. goto Blacklist

Figure 7. Forged/replay packets can evade firewall detection

The firewall could filter out the forged packets carrying undefined IP address, port number, protocols, or with abnormal packet format, but it failed to prevent the replay/fabricated message attacks as long as the packets comply with the predefined firewall rules. Most SCADA firewall products will work on single packet inspection by checking whether their packet frame match all the pre-defined rules.

The experiments show that once an attacker is able to access the ISCS network, he could launch a series attacks on the Metro system, besides on the PSD system. It is due to the weak security measures of the Metro system, as there are no message authentication and integrity mechanisms enforced in the communication protocols. Applying firewall can block the attack packets which are not allowed by the pre-defined policy rules, but the approach has its limitations. However, a SCADA firewall can be more efficient and advanced when it performs the following functions:

- Restrict unknown devices to connect to SCADA network -- If the SCADA network uses static IP addresses instead of Dynamic Host Configuration Protocol (DHCP), it will be harder for an attacker to connect his/her attacking tools to the network.
- Only allow connections on known ports, restrict traffic from new ports -- Ports look like the gates of a network. Unnecessary gates should always be kept locked. Then the firewall can only inspect the traffic pass the known ports.
- Able to do Deep Packet Inspection (DPI) [19] on non-standard industrial protocol -- However, available commercial SCADA firewall products, e.g., [14, 15] are designed to inspect the payload of standard industrial protocols.

The firewall technology alone could not completely protect the SCADA system from cyber attack. A consolidated security approach will be more effective which offers tightly integrated multiple detection mechanisms including rule based packet filtering and sender's authenticity verification.

6. Conclusion

We described a specific PSD system in detail, and analyzed its security vulnerabilities. The attacks, in the form of forged message and message replay, could exploit the vulnerabilities of weak security communication protocols in the ISCS system and disrupt the PSD system and other equipment operation. Countermeasures were studied that include employing secure communication protocols but faces challenges as it requires significant changes to the legacy system. The secure gateway approach, requiring less change to the legacy system, could perform the security functions otherwise to be done in RTU or PLC, and apply a wide range of security technologies to protect the system. Another approach, easier to be adopted, is to deploy firewall but it has limitations. The firewall approach could filter out packets based on pre-defined rules, but fail to block packets which comply with the rules. Due to the concern of disrupting the revenue generation Metro system operation, the experiments were not performed on the working PSD system, instead on a testbed of train station SCADA system. The experiment successfully demonstrated the forged and replay packet attacks to the system devices and the effectiveness of firewall protection in the SCADA system. This work points out that we should mind not only the platform gap, but the cyber security gap in the PSD system as well.

Acknowledgements

This work was supported by the National Research Foundation (NRF), Prime Minister's Office, Singapore, under its National Cybersecurity R&D Programme (Award No. NRF2014NCR-NCR001-31) and administered by the National Cybersecurity R&D Directorate. The special thanks are also given to SMRT Corporation Ltd to provide the TDS for the experiments and provide domain knowledge and technical support.

References

[1] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn, Guide to Industrial Control Systems (ICS) Security, NIST Special Publication 800-82 Revision 2, May 2015, http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf

[2] C. Ten, et al, Vulnerability Assessment of Cyber Security for SCADA Systems Using Attack Tree, Power Engineering Society General Meeting, 2007. IEEE, 1-8.

[3] M. Ozturk, and P. Aubin, SCADA Security: Challenges and Solutions, June 2011, http://www2.schneider-electric.com/documents/support/cybersecurity/WP-SCADASecurity-schneider-electric.pdf

[4] A. Shahzad, "Cryptography and authentication placement to provide secure channel for SCADA communication", International Journal of Security, Vol. 6 (3), 2012, 28- 44.

[5] W. Gao, T. Morris, B. Reaves, and D. Richey, On SCADA Control System Command and Response Injection and Intrusion Detection, eCrime Researchers Summit (eCrime), 2010, 1-9.

[6] A. Bertout, and E. Bernard, Next Generation of Railway and Metros Wireless Communication Systems, Aspect-IRSE, 2012, 1-8.

[7] Simply Modbus, http://www.simplymodbus.ca/index.html

[8] Modbus Application Protocol Specification, V1.1b3, http://www.modbus.org

[9] Fersil, Copp Platfrom Screen Door Control System, http://www.fersil-railway.com/en/systems-software/platform-screen-doors-control-psdc/copp-system/

[10] NRT, Platform Screen Door System, http://www.platform-screen-doors.com/platform-screen-doors/overview

[11] ST Electronics, Platform Screen Door System, http://www.stee.stengg.com/pdf/railway_systems/Taipei_PSD-Eng.pdf

[12] I. Fovino, A. Carcano, M. Masera, and A. Trombetta, Design and Implementation of a Secure Modbus Protocol, Third Annual IFIP WG 11.10 International Conference on Critical Infrastructure Protection, Hanover, New Hampshire, USA, March 2009, 83-96.

[13] G. Hayes, and K. El-Khatib, Securing Modbus Transactions Using Hash-Based Message Authentication Codes and Stream Transmission Control Protocol, Third International Conference on Communication and Information Technology, Beirut, 2013, 179-184.

[14] Tofino, https://www.tofinosecurity.com/products/Tofino-Firewall-LSM

[15] Checkpoint, https://www.checkpoint.com/products-solutions/next-generation-firewalls/

[16] S. Jung, J. Song, and S. Kim, Design on SCADA Test-bed and Security Device, International Journal of Multimedia and Ubiquitous Engineering, Vol. 3, No. 4, October, 2008

[17] Colasoft Packet Builder, http://www.colasoft.com/packet_builder/

[18] PackEth Linux Software, http://packeth.sourceforge.net/packeth/Home.html

[19] H. Schulze, Deep packet inspection tutorial, https://ipoque.com/sites/default/files/mediafiles/documents/iss2012/ISS-Tutorial-DPI1.pdf

A Systems Approach to Cyber Security
A. Roychoudhury and Y. Liu (Eds.)

doi:10.3233/978-1-61499-744-3-86

Attribute-Based Secure Messaging in the Public Cloud

Zhi Yuan Poh, Hui Cui, Robert H. Deng, and Yingjiu Li
School of Information Systems, Singapore Management University

Abstract. Messaging systems operating within the public cloud are gaining popularity. To protect message confidentiality from the public cloud including the public messaging servers, we propose to encrypt messages in messaging systems using Attribute-Based Encryption (ABE). ABE is an one-to-many public key encryption system in which data are encrypted with access policies and only users with attributes that satisfy the access policies can decrypt the ciphertexts, and hence is considered as a promising solution for realizing expressive and fine-grained access control of encrypted data in public servers. Our proposed system, called Attribute-Based Secure Messaging System with Outsourced Decryption (ABSM-OD), has three key features: enabling expressive and fine-grained access control of encrypted messages by users, supporting outsourced decryption to the cloud while without compromising confidentiality of decrypted messages, and allowing server-aided revocation to provide effective and instant user revocations.

Keywords. Attribute-Based Encryption, Secure Messaging, Outsourced Decryption

1. Introduction

Messaging systems such as WhatsApp, Facebook Messenger, WeChat, Line, Viber, etc are becoming very popular. Users from different localities have their preferred choices of messaging services[1]. Since these messaging systems reside in the public domain and are subjected to threats on the Internet, security savvy users might be reluctant to trust the service providers to protect the privacy of their messages and there is a growing demand to provide end-to-end encryption in public messaging services. Furthermore, in a threat landscape study, instant messaging platforms are becoming attack vectors which can result in further damages[2].

Messages can be in the form of Electronic Mail (Email), Short Message Service (SMS), Instant Messaging (IM), etc which allow users to share information and collaborate effectively. Previous messaging services focus on functionality over security and the threats on the Internet poses challenges on these messaging services. In 1991, Phil Zimmermann introduced the Pretty Good Privacy (PGP)[3] to protect the confidentiality and authenticity of emails. Later in 1995, Secure/Multipurpose Internet Mail Extensions (S/MIME) (currently version 3.2[4]) was introduced to provide a standard way to protect

[1] https://www.similarweb.com/blog/worldwide-messaging-apps
[2] https://www.fortinet.com/content/dam/fortinet/assets/white-papers/Executive-Summary-CTAP.pdf
[3] PGP https://en.wikipedia.org/wiki/Pretty_Good_Privacy
[4] S/MIME version 3.2 Message Specification https://tools.ietf.org/html/rfc5751

Multipurpose Internet Mail Extensions (MIME) email messages. However, due to the complexity of the solutions and the need for user involvement, both PGP and S/MIME are not widely adopted today [14]. In 2014, Google started implementing the OpenPGP[5] standard (IETF RFC4880[6]) as a Chrome Extension (End-To-End)[7] to enhance the messaging security within the browser. However, PGP and S/MIME are based on the traditional public key encryption hence have a drawback in scalability. Transport messaging protocols like Secure Sockets Layer (SSL), Transport Layer Security (TLS), and Secure Shell (SSH), etc were introduced and enhanced to protect the delivery of messages. However, transport protection is inadequate in protecting message security and privacy as these messaging systems assume that the servers are trusted.

The Signal protocol is used in Signal [7], WhatsApp[8], Facebook Messenger[9] and Allo[10] to provide end-to-end encryption. These systems rely on trusted servers to exchange users' public keys during communication. With the threats in the public domain, key management and transport protection are insufficient in protecting the data security of messaging systems. In addition, these messaging systems assume that the servers residing in the public domain are trusted. Public threats and malicious insiders can reduce the data security and user privacy of these messaging systems.

ABE is a one-to-many public-key encryption where private keys of users and access policy of encrypted data are based on user attributes. ABE allows a sender to embed access policy with encrypted data and only authorized users will be able to gain access to the original data and is widely considered as a promising technique for providing expressive and fine-grained access control of end-to-end encrypted data.

Some studies have considered secure messaging from an access control perspective by integrating ABE with existing messaging systems [4][11][17]. However, these systems require the users to perform full decryption which can be resource intensive for mobile users, especially if the ciphertexts have complex access policies. In [4] and [11], revocation is achieved by issuing decryption keys with expiry date, however a direct revocation approach might be more desirable.

In this work, we propose an Attribute-Based Secure Messaging System with Outsourced Decryption (ABSM-OD) which provides end-to-end message security on the cloud. ABSM-OD is designed to operate in environments where the messaging servers reside in a public untrusted domain. Messages are stored on the cloud and only authorized users will be able to obtain the original data. Specifically, ABSM-OD possesses the following three features.

Fine-grained Access Control of Encrypted Messages The use of ABE allows expressive and fine-grained access control to be enforced on encrypted messages which ensures end-to-end message confidentiality.

Outsourced Decryption The computation of ABE decryption is offloaded to the cloud, hence keeping the resource requirements on the users to the minimal and without exposing users' messages to the cloud.

[5]MIME Security with OpenPGP https://tools.ietf.org/html/rfc3156
[6]OpenPGP Message Format https://tools.ietf.org/html/rfc4880
[7]End-To-End https://github.com/google/end-to-end
[8]https://whispersystems.org/blog/whatsapp-complete/
[9]https://whispersystems.org/blog/facebook-messenger/
[10]https://whispersystems.org/blog/allo/

Effective User Revocation Compromised users can be directly revoked from the system with the use of server-aided revocation technique. Users and attributes can be managed effectively within the system.

With the features of ABSM-OD, enterprises operating their messaging systems in the cloud will be able to preserve message security. Since only ciphertexts are available on the cloud, cloud service providers will have no access to the underlying messages. If the messaging systems are compromised, adversaries will only obtain the ciphertexts and not the original messages. Furthermore, enterprises will be able to manage the access control of messages effectively within a cloud environment.

2. Preliminaries

This section describes the notions that are to be used in the construction of ABSM-OD.

Symmetric Key Encryption: The scheme consists of the following:

- Key Generation $k \leftarrow \mathsf{Gen}^{SE}$: It outputs a random key k.
- Encryption $\mathsf{CT}_{\mathsf{M}} \leftarrow \mathsf{Enc}^{SE}(k, \mathsf{M})$: Takes a key k and message M. It encrypts message M with key k and outputs ciphertext CT_{M}.
- Decryption $\mathsf{M}/\perp \leftarrow \mathsf{Dec}^{SE}(k, \mathsf{CT}_{\mathsf{M}})$: Takes a key k and ciphertext CT_{M}. It decrypts ciphertext CT_{M} with key k and outputs message M or $\perp$ indicates error.

Digital Signature: The scheme consists of the following:

- Key Generation $(pk, sk) \leftarrow \mathsf{Gen}^{DS}$: It outputs a signing key sk and verifying key pk.
- Message Signing $\sigma_{\mathsf{M}} \leftarrow \mathsf{Sign}^{DS}(sk, \mathsf{M})$: Takes a signing key sk and message M. It signs message M with signing key sk and outputs message signature σ_{M}.
- Message Verification $1/0 \leftarrow \mathsf{Verify}^{DS}(pk, \mathsf{M}, \sigma_{\mathsf{M}})$: Takes a verifying key pk, message M and message signature σ_{M}. It verifies message M with verifying key pk. It outputs 1 when the message is valid, 0 if otherwise.

Ciphertext-Policy ABE with outsourced decryption: An ABE with outsourcing decryption scheme proposed by Green et al. [9] that we denote as ABE-OD consists of the following:

- $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathsf{Setup}^{ABE}(\kappa, \mathbb{U})$: Takes a security parameter κ and generates master secret key MSK and master public key MPK.
- $\mathsf{CT}_{A_{\mathsf{M}}} \leftarrow \mathsf{Encrypt}^{ABE}(\mathsf{MPK}, \mathsf{M}, A_{\mathsf{M}})$: Encrypts M using MPK and access structure A_{M} and outputs an ABE ciphertext $\mathsf{CT}_{A_{\mathsf{M}}}$.
- $(\mathsf{DK}_{A_u}, \mathsf{TK}_{A_u}) \leftarrow \mathsf{KeyGen}^{ABE}(\mathsf{MSK}, A_u)$: Generates transformation and decryption keys using input master secret key MSK and user attributes A_u. It outputs a decryption key DK_{A_u} and a transformation key TK_{A_u}.
- $\mathsf{CT}_{out} \leftarrow \mathsf{Transform}^{ABE}(\mathsf{TK}_{A_u}, \mathsf{CT}_{A_{\mathsf{M}}})$: Partially decrypts ciphertext $\mathsf{CT}_{A_{\mathsf{M}}}$ using transformation key TK_{A_u} and outputs a partially decrypted ciphertext CT_{out}.
- $\mathsf{M}/\perp \leftarrow \mathsf{Decrypt}^{ABE}(\mathsf{DK}_{A_u}, \mathsf{CT}_{out})$: Decrypts partially decrypted ciphertext CT_{out} with decryption key DK_{A_u} and outputs a message M or $\perp$ where $\perp$ indicates error.

3. Proposed System Overview

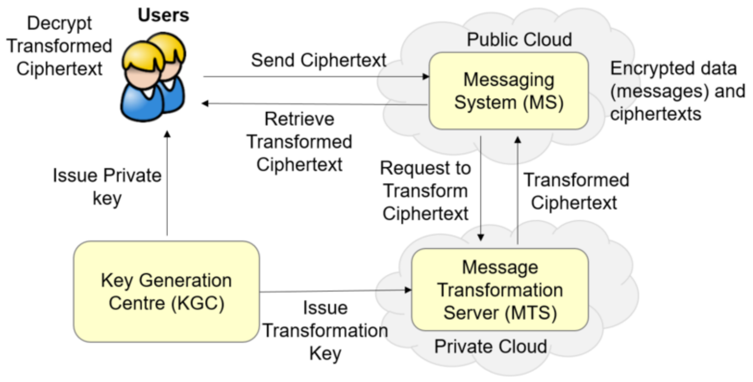

Decrypt Transformed Ciphertext
Users
Send Ciphertext
Public Cloud
Messaging System (MS)
Encrypted data (messages) and ciphertexts
Retrieve Transformed Ciphertext
Issue Private key
Request to Transform Ciphertext
Transformed Ciphertext
Key Generation Centre (KGC)
Message Transformation Server (MTS)
Issue Transformation Key
Private Cloud

Figure 1. Attribute-Based Secure Messaging System with Outsourced Decryption

Figure 1 illustrates the overall proposed system architecture of ABSM-OD. The system comprises of four parties namely Key Generation Centre (KGC), Message Transformation Server (MTS), Messaging System (MS) and users. The KGC manages user attributes, issues decryption keys to users based on their attributes and generates transformation keys to the MTS. The MTS determines recipients using the access policies of the ciphertexts and transforms the ciphertexts for each recipient. The MS stores two types of ciphertexts, ciphertexts received from the users and ciphertexts transformed by the MTS. Users interact with the MS to send/retrieve ciphertexts.

The KGC generates the public parameter and the master private key, and issues MTS with a signing and verifying key pair. Before a user can send/retrieve ciphertexts, a user needs to be registered in the system. The user generates a signing and verifying key pair and provides the verifying key to the KGC. The KGC registers the user with the verifying key, a set of user attributes and an user identifier, and generates ABE attribute keys (i.e. transformation and decryption keys) for the user. The KGC issues the decryption key to the user and distributes the user identifier, user verifying key, user attributes and transformation key to the MTS. Thereafter, the user will be able to send/retrieve ciphertexts.

When the user wants to send a message, the user encrypts the message with an access policy and signs the ciphertext with a timestamp and the user identifier. The user then sends the ciphertext, timestamp, user identifier and signature to the MS. When the MS receives the ciphertext, the MS requests the user verifying key from the MTS using the user identifier. If the user is unauthorized or revoked, no user verifying key is returned and the MS will discard the ciphertext. If the user is valid, the MTS will return the user verifying key to the MS for ciphertext verification. If MS fails to verify the ciphertext, the ciphertext will be discarded. Once the ciphertext is verified, the MS will forward the verified ciphertext to the MTS. As the MS is subjected to attacks in the public domain, the MTS will verify the ciphertext again. If the verification fails, MTS will discard the ciphertext. Once the ciphertext is verified, MTS resolves the recipients based on the access policy of the ciphertext, transforms the ciphertext for each recipient and signs on the transformed ciphertexts. Thereafter, the MTS requests for MS to store the transformed ciphertexts. When the users request for the ciphertexts, the MS will send the transformed ciphertexts to the users. Users will verify the transformed ciphertexts using the MTS verifying key. If verification fails, the transformed ciphertexts are discarded. If the ciphertexts are valid, users will use their decryption keys to decrypt the ciphertexts to obtain the original messages.

When a user is compromised, KGC will remove the user verifying key, user attributes and transformation key from the MTS. Subsequently, the compromised user will not be able to retrieve existing and new ciphertexts and any new ciphertexts generated by the compromised user will be rejected by the MS or MTS.

3.1. Threat Model

We assume that the KGC is trusted. MTS is honest-but-curious such that it will honestly follow the protocol as required but will attempt to obtain sensitive information such as the message encrypted in a ciphertext. MTS operates within a private cloud and is assumed not to collude with users. MS is untrusted and resides in the public cloud.

Any adversary will be able to compromise and gain access to the MS within the public domain. If the adversary gain access to the MS, the ciphertexts and the transformed ciphertexts should not reveal sensitive information.

3.2. Design Objectives

- **Confidentiality**
  The main objective is to design a secure messaging system that can provide end-to-end message confidentiality. The sender determines the access policies of the encrypted messages and only privileged users can access the underlying messages.

- **Efficient Decryption**
  As mobile users may have limited resources, it is desirable to reduce the computational cost of users in decryption. Towards this end, we adopt an ABE with outsourced decryption [9] to offload decryption operations to the cloud.

- **Message Authenticity**
  With the prevalent of spam and malicious messages, it can be challenging for users to recognize if received messages are valid, hence it will be desirable that received messages are indeed authentic. As the sender will sign on the message ciphertexts and MTS will sign on the transformed ciphertexts, the signing of ciphertexts will provide the authenticity of the messages.

- **Traceability**
  The source and changes of ciphertexts can be crucial for audit purposes, hence it is desirable for the system to maintain the traceability of the ciphertexts. To achieve traceability, user identifiers are included into the ciphertexts and digital signatures are used by the sender and MTS to sign the ciphertexts.

- **Revocation**
  Another consideration is to provide an effective user and attribute revocation, any

revoked users should no longer be able to access the system to send/retrieve messages. The use of direct revocation of users with server-aided technique achieves this objective.

4. ABSM-OD Construction

In this section, we describe a concrete construction of ABSM-OD based on ABE-OD [9], symmetric encryption and digital signature.

Assuming that the KGC keeps a list $KGCSTORE$, storing the user identifier oid_u, user verifying key pk_u, user attributes A_u. A tuple in $KGCSTORE$ is represented as $\langle oid_u, pk_u, A_u \rangle$.

The MTS keeps a list $MTSSTORE$, storing the user identifier oid_u, user verifying key pk_u, user attributes A_u, transformation key TK_u. A tuple in $MTSSTORE$ is represented as $\langle oid_u, pk_u, A_u, \mathsf{TK}_u \rangle$.

The MS keeps two types of ciphertexts (message ciphertexts and transformed ciphertexts) in the list $MSGSTORE$. A message ciphertext includes a message identifier mid_{M}, user identifier oid_u, access structure A_{M}, message timestamp ts_{M}, message ciphertext $\mathsf{CT}^{ABE}_{\mathsf{M}}$ and ciphertext signature $\sigma_{\mathsf{CT}_{ABE}}$. A tuple of message ciphertext in $MSGSTORE$ is represented as $\langle mid_{\mathsf{M}}, oid_u, A_{\mathsf{M}}, ts_{\mathsf{M}}, \mathsf{CT}^{ABE}_{\mathsf{M}}, \sigma_{\mathsf{CT}_{ABE}} \rangle$. A transformed ciphertext includes a new message identifier mid_{TM_i}, sender user identifier oid_s, recipient user identifier oid_{r_i}, a creation timestamp ts_{TM_i}, transformed ciphertext $\mathsf{CT}^{ABE}_{\mathsf{TM}_i}$ and transformed ciphertext signature σ_{T_i}. A tuple of transformed ciphertext in $MSGSTORE$ is represented as $\langle mid_{\mathsf{TM}_i}, oid_s, oid_{r_i}, ts_{\mathsf{TM}_i}, \mathsf{CT}^{ABE}_{\mathsf{TM}_i}, \sigma_{T_i} \rangle$.

System Initialization $(\mathsf{MPK}, \mathsf{MSK}, pk_{mts}, sk_{mts}) \leftarrow \mathsf{Setup}(\kappa, \mathbb{U})$: During the initialization, KGC runs the ABE setup to create the master secret key MSK and public parameters MPK (i.e $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}^{ABE}(\kappa, \mathbb{U})$). In addition, KGC issues a pair of signing key and verifying key to the MTS (i.e. $(pk_{mts}, sk_{mts}) \leftarrow \mathsf{Gen}^{DS}$ for MTS).

User Registration $(\mathsf{DK}_u, \mathsf{TK}_u) \leftarrow \mathsf{RegisterUser}(\mathsf{MSK}, oid_u, pk_u, A_u)$: To allow a user u to use the system, KGC needs to register the user u in the system. First, the user u generates a signing and verifying key pair $(pk_u, sk_u) \leftarrow \mathsf{Gen}^{DS}$ and provides the verifying key pk_u to the KGC. KGC registers the user u in the system with the user identifier oid_u, user verifying key pk_u, user attributes A_u and provides the user identifier oid_u, user verifying key pk_u to the MTS. KGC performs the *Attribute Key Generation* (i.e $(\mathsf{DK}_u, \mathsf{TK}_u) \leftarrow \mathsf{KeyGen}(oid_u, \mathsf{MSK}, A_u)$) to update transformation key TK_u and user attributes A_u of user u to the MTS and issues the decryption key DK_u to user u.

Attribute Key Generation $(\mathsf{DK}_u, \mathsf{TK}_u) \leftarrow \mathsf{KeyGen}(oid_u, \mathsf{MSK}, A_u)$: When a user u requires a set of attribute keys, KGC performs the ABE Key Generation (i.e $(\mathsf{DK}_u, \mathsf{TK}_u) \leftarrow \mathsf{KeyGen}^{ABE}(\mathsf{MSK}, A_u)$) to obtain the decryption key DK_u and transformation key TK_u of user u.

Attribute Key Update $(\mathsf{DK}'_u, \mathsf{TK}'_u) \leftarrow \mathsf{KeyUpdate}(oid_u, \mathsf{MSK}, A'_u)$: When the user attributes of user u changes from A_u to A'_u, KGC updates the user attributes from A_u to A'_u.

KGC revokes the transformation key TK_u of user u on the MTS. Also, KGC performs *Attribute Key Generation* (i.e $(\mathsf{DK}_u^{'}, \mathsf{TK}_u^{'}) \leftarrow \mathsf{KeyGen}(oid_u, \mathsf{MSK}, A_u^{'})$) and updates MTS with the new transformation key $\mathsf{TK}_u^{'}$ and new user attributes $A_u^{'}$. KGC issues the new decryption key $\mathsf{DK}_u^{'}$ to user u.

User Revocation $1/0 \leftarrow \mathsf{RevokeUser}(oid_u)$: When a user u is compromised, the user u will need to be revoked in the system. KGC and MTS will need to revoke the user verifying key pk_u, user attributes A_u and transformation key TK_u of user u. The tuples in $KGCSTORE$ and $MTSSTORE$ will be updated as $\langle oid_u, -, - \rangle$ and $\langle oid_u, -, -, - \rangle$ respectively. Once revoked, any verification of ciphertexts signed by user u with identifier oid_u at the MS and MTS will fail and the MTS will not transform any new ciphertexts for user u.

Message Encryption $\mathsf{CT}_{ABE} \leftarrow \mathsf{EncryptMessage}(\mathsf{MPK}, \mathsf{M}, A_{\mathsf{M}}, oid_u, sk_u)$: When a user u wants to send a message M, the user u will specify the access structure A_{M} to encrypt the message. The user u generates a random key $k_{\mathsf{M}} \leftarrow \mathsf{Gen}^{SE}$ to encrypt the message M (i.e $\mathsf{CT}_{\mathsf{M}} = \mathsf{Enc}^{SE}(k_{\mathsf{M}}, \mathsf{M})$). The user u performs the ABE encryption on the random key k_{M} with access structure A_{M} to get the key ciphertext $\mathsf{CT}_{k_{\mathsf{M}}}^{ABE} = \mathsf{Encrypt}^{ABE}(\mathsf{MPK}, k_{\mathsf{M}}, A_{\mathsf{M}})$. The ciphertext $\mathsf{CT}_{\mathsf{M}}^{ABE} = (\mathsf{CT}_{\mathsf{M}}, \mathsf{CT}_{k_{\mathsf{M}}}^{ABE})$ forms the ciphertext for the message M. The ciphertext $\mathsf{CT}_{\mathsf{M}}^{ABE}$, a message identifier mid_{M}, user identifier oid_u and ciphertext timestamp ts_{M} forms the ciphertext $\mathsf{CT}_{ABE_{\mathsf{M}}} = (mid_{\mathsf{M}}, oid_u, A_{\mathsf{M}}, ts_{\mathsf{M}}, \mathsf{CT}_{\mathsf{M}}^{ABE})$ for the MS. Before sending the ciphertext $\mathsf{CT}_{ABE_{\mathsf{M}}}$, the user u signs the ciphertext $\mathsf{CT}_{ABE_{\mathsf{M}}}$ with the user signing key sk_u (i.e $\sigma_{\mathsf{CT}_{ABE}} \leftarrow \mathsf{Sign}^{DS}(sk_u, \mathsf{CT}_{ABE_{\mathsf{M}}})$) and sends the ciphertext $\mathsf{CT}_{ABE} = (\mathsf{CT}_{ABE_{\mathsf{M}}}, \sigma_{\mathsf{CT}_{ABE}})$ to MS.

MS and MTS will verify the ciphertext $\mathsf{CT}_{ABE} = (\mathsf{CT}_{ABE_{\mathsf{M}}}, \sigma_{\mathsf{CT}_{ABE}})$ using the user verifying key pk_u (i.e $\mathsf{Verify}^{DS}(pk_u, \mathsf{CT}_{ABE_{\mathsf{M}}}, \sigma_{\mathsf{CT}_{ABE}})$). Once the ciphertext is verified, the MTS determines recipients with attributes that satisfies the access structure A_{M} of the ciphertext CT_{ABE}. With the list of recipients $R_{A_{\mathsf{M}}}$, the MTS uses the transformation key TK_{r_i} of each recipient $r_i \in R_{A_{\mathsf{M}}}$ to transform the ciphertext CT_{ABE}.

Message Transformation $\mathsf{CT}_{\mathsf{TM}_i} \leftarrow \mathsf{TransformMessage}(\mathsf{CT}_{ABE}, oid_{r_i}, \mathsf{TK}_{r_i}, sk_{mts})$: The MTS parses the ciphertext $\mathsf{CT}_{ABE} = (\mathsf{CT}_{ABE_{\mathsf{M}}}, \sigma_{\mathsf{CT}_{ABE}})$, then parses ciphertext $\mathsf{CT}_{ABE_{\mathsf{M}}} = (mid_{\mathsf{M}}, oid_u, A_{\mathsf{M}}, ts_{\mathsf{M}}, \mathsf{CT}_{\mathsf{M}}^{ABE})$ and transforms the ciphertext $\mathsf{CT}_{\mathsf{M}}^{ABE}$ for a user u. The MTS transforms the key ciphertext $\mathsf{CT}_{k_{\mathsf{M}}}^{ABE}$ ($\mathsf{CT}_{\mathsf{M}}^{ABE} = (\mathsf{CT}_{\mathsf{M}}, \mathsf{CT}_{k_{\mathsf{M}}}^{ABE})$) to get the transformed key ciphertext $\mathsf{CT}_{out_i}^{ABE} = \mathsf{Transform}^{ABE}(\mathsf{TK}_{r_i}, \mathsf{CT}_{k_{\mathsf{M}}}^{ABE})$ for a user u. Once the key ciphertext $\mathsf{CT}_{k_{\mathsf{M}}}^{ABE}$ is transformed, MTS needs the MS to store the transformed ciphertext $\mathsf{CT}_{out_i}^{ABE}$. The ciphertext $\mathsf{CT}_{\mathsf{TM}_i}^{ABE} = (\mathsf{CT}_{\mathsf{M}}, \mathsf{CT}_{out_i}^{ABE})$ forms the transformed ciphertext for user u. A transformed message identifier mid_{TM_i}, sender user identifier oid_u, recipient user identifier oid_{r_i} and timestamp ts_{TM_i} form the transformed ciphertext $\mathsf{CT}_{T_i} = (mid_{\mathsf{TM}_i}, oid_u, oid_{r_i}, ts_{\mathsf{TM}_i}, \mathsf{CT}_{\mathsf{TM}_i}^{ABE})$ for user u that will be stored by MS. MTS signs the transformed ciphertext CT_{T_i} with the MTS signing key sk_{mts} to get the signature σ_{T_i} (i.e $\sigma_{T_i} \leftarrow \mathsf{Sign}^{DS}(sk_{mts}, \mathsf{CT}_{T_i})$). The MTS will then send the ciphertext $\mathsf{CT}_{\mathsf{TM}_i} = (\mathsf{CT}_{T_i}, \sigma_{T_i})$ to the MS. It outputs the transformed ciphertext $\mathsf{CT}_{\mathsf{TM}_i}$.

Message Decryption $\mathsf{M}/\perp \leftarrow \mathsf{DecryptMessage}(\mathsf{CT}_{\mathsf{TM}_i}, pk_{mts}, \mathsf{DK}_{r_i})$: When a user u receives the transformed ciphertext $\mathsf{CT}_{\mathsf{TM}_i} = (\mathsf{CT}_{T_i}, \sigma_{T_i})$, the user u verifies the transformed ciphertext $\mathsf{CT}_{\mathsf{TM}_i}$ with MTS verifying key pk_{mts} (i.e $\mathsf{Verify}^{DS}(pk_{mts}, \mathsf{CT}_{T_i}, \sigma_{T_i})$). Once the transformed ciphertext is verified, the user will decrypt the transformed key ciphertext $\mathsf{CT}^{ABE}_{out_i}$ using the decryption key DK_{r_i} to retrieve the random key $k_{\mathsf{M}}/\perp = \mathsf{Decrypt}^{ABE}(\mathsf{DK}_{r_i}, \mathsf{CT}^{ABE}_{out_i})$ or $\perp$ indicates error.

If the user u is able to retrieve the random key k_{M}. The user u performs the decryption to retrieve the message $\mathsf{M}/\perp = \mathsf{Dec}^{SE}(k_{\mathsf{M}}, \mathsf{CT}_{\mathsf{M}})$. If the decryption fails, an error $\perp$ will be returned to the user indicating an error.

4.1. Security Analysis

This section analyses the security of the system.

- **Confidentiality**
  Since all the ciphertexts are stored in the encrypted forms, malicious users whose attributes do not satisfy the access policy of the ciphertext cannot obtain the content of the underlying message. Therefore, our ABSM-OD system preserves the confidentiality of the data users.

- **Efficient Decryption**
  With the use of ABE-OD scheme, the MTS will help the privileged users with decryption and each user will only be required to perform the final decryption to retrieve the original message, hence reducing the computational cost requirements on the users.

- **Message Authenticity**
  Any modification of ciphertexts will be detected by the signature verification.

- **Traceability**
  When some ciphertexts or transformed ciphertexts are found with issues, the ciphertexts can be traced back to the source according to the ciphertext signature.

- **Revocation**
  Once a user is revoked from the system, KGC updates the list in the MTS. The ciphertext with a signature signed by a user that is not in the user list in the MTS will not be accepted by MTS. Also, no ciphertext will be transformed for revoked users as there is no transformation key in the MTS for revoked user. Users are effectively revoked from the system.

5. Implementation

We implemented the ABSM-OD as a Simple Chat System in Java. Tests were conducted for the main functions of the system. In the tests, we measure the recipients resolution time, encryption time, transformation time and decryption time with different number of attributes and data size. The recipients resolution time is the time taken to determine the number of recipients based on the access policy of a ciphertext. Other performance overheads like database, disk and network latency which are dependent on the deployment are not the focus of our tests.

We tested the performance of recipients resolution on a system with Intel i7 2.6GHz processor with 768 MB of RAM. Assuming that the system is supporting 10K users, we generated random messages that are encrypted with access policies of 10 attributes and operators (”AND”/”OR”). The results indicate that the system is able to resolve on average 1404 recipients in around 514ms.

We also tested the performance of cryptographic operations on a system with Intel i7 2.6GHz processor with 8 GB of RAM. We generated random messages of different sizes (1-64 MB) with access policies of 10 attributes and ”AND” operators. The messages were encrypted with an average time of around 556ms, transformed around 144ms and decrypted around 7ms. To test for possible worst case situation, we generated messages of different sizes (1-64 MB) with access policies of 50 attributes and ”AND” operators. The messages were encrypted with an average time of around 2.8s, transformed around 728ms and decrypted around 7ms. Table 1 summarized the preliminary results.

Table 1. Performance of Cryptographic Operations

No of Attributes	Message Size	Operators	Average Time		
			Encryption	Transformation	Decryption
10	1-64 MB	AND	556ms	144ms	7ms
50	1-64 MB	AND	2.8s	728ms	7ms

6. Related Work

Attribute-Based Encryption - ABE schemes can be broadly categorized into two categories namely, Key Policy ABE (KP-ABE) and Ciphertext- Policy ABE (CP-ABE). In KP-ABE schemes, ciphertexts are associated with attributes while decryption keys are associated with access structures. However, in CP-ABE schemes, the association is reversed where ciphertexts are associated with access structures while decryption keys are associated with attributes.

The concept of ABE was first proposed by Sahai and Waters [13] as a type of Fuzzy Identity Based Encryption (Fuzzy-IBE) scheme. Subsequently, Goyal et al. [8] introduced the notion of KP-ABE and CP-ABE and provided the construction for KP-ABE scheme. Soon after, Bethencourt et al. [3] provided the construction for CP-ABE. Initial work on ABE schemes focus on monotonic access structures, but soon, Ostrovsky et al. [10] proposed an ABE scheme with non-monotonic policies to represent negative constraints. To meet the need for complex access policy, several efforts focus on providing more fine-grained access control. For instance, Bobba et al. [5] extended CP-ABE by representing attributes as a recursive set structure and named the scheme Ciphertext-policy Attribute-Set-Based Encryption (CP-ASBE or ASBE) while Wang et al. [16] introduced Hierarchical Attribute-Based Encryption (HABE) by combining Hierarchical Identity-based Encryption (HIBE) with CP-ABE. Thereafter, Wan et al. [15] introduced Hierarchical Attribute-Set-Based Encryption (HASBE) by extending ASBE with users organized into a hierarchical structure to provide a more scalable, flexible and fine-grained control of ABE.

Revocable ABE - Attrapadung and Imai [1] proposed two types of revocation techniques, namely direct revocation and indirect revocation. Direct revocation is performed by sender specifying the revocation list while indirect revocation is enforced by the key authority providing key updates to non-revoked users. Attrapadung and Imai [2] also presented a hybrid revocable attribute encryption (HR-ABE) by combining the two techniques and data owner may select either technique. Boldyreva et al. [6] proposed an efficient revocable KP-ABE scheme by improving the key updates. Sahai et al. [12] proposed a revocable scheme by revoking stored data together with key updates. Yang et al. [18] proposed a revocable ABE scheme by denying user decryption capability via a cloud server.

Secure Messaging - Messaging protocols and solutions secure messages with a wide range of techniques [14]. In particular, ABE is an upcoming and promising technique that can achieve secure messaging. Bobba et al. [4] proposed Attribute-Based Messaging (ABM) by integrating ABE with Mail Transfer Agent (MTA). Weber et al. [17] proposed MundoMessage which integrated ABE and Location-Based Encryption (LBE) into emergency communication. In healthcare, Picazo-Sanchez et al. [11] incorporated ABE into messaging protocol for monitoring and managing the medical wireless body area networks (WBANs). Unger et al. [14] evaluated existing messaging systems and established that messaging systems face three key challenges: trust establishment, conversation privacy and transport privacy. They also proposed a framework to evaluate the properties of messaging systems.

7. Conclusion

In this paper, we proposed a system architecture for an Attribute-Based Secure Messaging System with Outsourced Decryption (ABSM-OD) that provides end-to-end message security in the cloud. ABSM-OD is built upon ABE-OD scheme [9] and achieves three key features namely fine-grained access control, outsourced decryption and effective user revocation. Our prototype implementation demonstrates the feasibility of the architecture with reasonable performance. From our preliminary results, the system is able to determine recipients from the access polices of ciphertexts within an acceptable amount of time. Also, the decryption of transformed ciphertexts are more manageable for mobile users. Hence, ABSM-OD can provide scalable and secure messaging within the public cloud.

Acknowledgments

This research work is supported by the Singapore National Research Foundation under the NCR Award Number NRF2014NCR-NCR001-012.

References

[1] N. Attrapadung and H. Imai. *Attribute-Based Encryption Supporting Direct/Indirect Revocation Modes*, pages 278–300. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[2] N. Attrapadung and H. Imai. *Conjunctive Broadcast and Attribute-Based Encryption*, pages 248–265. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[3] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, SP '07, pages 321–334, Washington, DC, USA, 2007. IEEE Computer Society.

[4] R. Bobba, O. Fatemieh, F. Khan, A. Khan, C. A. Gunter, H. Khurana, and M. Prabhakaran. Attribute-based messaging: Access control and confidentiality. *ACM Trans. Inf. Syst. Secur.*, 13(4):31:1–31:35, Dec. 2010.

[5] R. Bobba, H. Khurana, and M. Prabhakaran. Attribute-sets: A practically motivated enhancement to attribute-based encryption. In *Proceedings of the 14th European Conference on Research in Computer Security*, ESORICS'09, pages 587–604, Berlin, Heidelberg, 2009. Springer-Verlag.

[6] A. Boldyreva, V. Goyal, and V. Kumar. Identity-based encryption with efficient revocation. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, pages 417–426, New York, NY, USA, 2008. ACM.

[7] T. Frosch, C. Mainka, C. Bader, F. Bergsma, T. Holz, et al. How secure is textsecure? In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 457–472. IEEE, 2016.

[8] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98. ACM, 2006.

[9] M. Green, S. Hohenberger, and B. Waters. Outsourcing the decryption of abe ciphertexts. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, pages 34–34, Berkeley, CA, USA, 2011. USENIX Association.

[10] R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 195–203, New York, NY, USA, 2007. ACM.

[11] P. Picazo-Sanchez, J. E. Tapiador, P. Peris-Lopez, and G. Suarez-Tangil. Secure publish-subscribe protocols for heterogeneous medical wireless body area networks. *Sensors*, 14(12):22619, 2014.

[12] A. Sahai, H. Seyalioglu, and B. Waters. *Dynamic Credentials and Ciphertext Delegation for Attribute-Based Encryption*, pages 199–217. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[13] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'05, pages 457–473, Berlin, Heidelberg, 2005. Springer-Verlag.

[14] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith. Sok: Secure messaging. In *2015 IEEE Symposium on Security and Privacy*, pages 232–249, May 2015.

[15] Z. Wan, J. Liu, and R. H. Deng. Hasbe: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing. *IEEE Transactions on Information Forensics and Security*, 7(2):743–754, April 2012.

[16] G. Wang, Q. Liu, J. Wu, and M. Guo. Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers. *Comput. Secur.*, 30(5):320–331, July 2011.

[17] S. G. Weber, Y. Kalev, S. Ries, and M. Mühlhäuser. Mundomessage: Enabling trustworthy ubiquitous emergency communication. In *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*, ICUIMC '11, pages 29:1–29:10, New York, NY, USA, 2011. ACM.

[18] Y. Yang, X. Ding, H. Lu, Z. Wan, and J. Zhou. *Achieving Revocable Fine-Grained Cryptographic Access Control over Cloud Data*, pages 293–308. Springer International Publishing, Cham, 2015.

A Systems Approach to Cyber Security
A. Roychoudhury and Y. Liu (Eds.)

doi:10.3233/978-1-61499-744-3-97

Faster ECC over $\mathbb{F}_{2^{571}}$ (feat. PMULL)

Hwajung Seo
Institute for Infocomm Research (I2R), Singapore

Abstract. In this paper, we show efficient elliptic curve cryptography implementations over advanced ARMv8 processor. We improve the previous binary field multiplication over the processor with finely aligned multiplication and incomplete reduction techniques by taking advantages of advanced 64-bit polynomial multiplication (`PMULL`). This approach shows performance enhancements by a factor of 1.34 times than previous implementation of binary field multiplication. For the point addition and doubling, the special types of multiplication, squaring and addition operations are combined together and optimized, where one reduction operation is optimized in each case. The scalar multiplication is implemented in constant-time window method, which is secure against timing attacks. Finally the proposed implementations achieved 759,630/331,944 clock cycles for random/fixed scalar multiplications for B-571 curve over ARMv8, respectively.

Keywords. ARMv8, Elliptic Curve Cryptography, Binary Field Multiplication

1. Introduction

Elliptic Curve Cryptography (ECC) is the most popular Public Key Cryptography (PKC) in pre-quantum cryptography. However due to its high complexities of computations, the execution timing is serious problem for the practical applications. Particularly, the binary field multiplication is regarded as the most expensive operation in the elliptic curve cryptography. Many researchers have studied the high-speed implementation of binary field multiplication in order to improve the performance. The classical binary field multiplication performs the bitwise exclusive-or operation with the operands and the intermediate results when the target bit of operand is set to one [13,10,11]. The alternative approach takes advantages of the pre-computed Look-Up Table (LUT). The method constructs the part of results in advance and then the logical operations are replaced into the simple memory access operations [6,9]. Recently, the modern embedded processors support the advanced built-in polynomial multiplication. ARMv7 architecture supports `VMULL.P8` operation which computes eight 8-bit wise polynomial multiplications with single instruction and then outputs eight 16-bit results to the 128-bit NEON register. In [2], Câmara et al. shows that the efficient 64-bit polynomial multiplication with the `VMULL.P8` instruction. Since the `VMULL.P8` instruction only provides the outputs in vectorized formats, the author presents noble approaches to align the vectorized formats into sequential results. After then multiple levels of Karatsuba multiplications are applied to various binary field multiplications ranging from $\mathbb{F}_{2^{251}}$, $\mathbb{F}_{2^{283}}$ to $\mathbb{F}_{2^{571}}$. The advanced

ARMv8 architecture supports PMULL instruction which computes the 64-bit wise polynomial multiplication. This nice property improves the polynomial multiplication significantly over ARMv8. In CT-RSA'15, Gouvêa and López presented compact implementations of GCM based Authenticated Encryption (AE) with the built-in AES encryption and PMULL instruction [3]. Since the 128-bit polynomial multiplication only needs 4 times of PMULL instructions (64-bit polynomial multiplication), the basic multiplication approach shows better performance than asymptotically faster Karatsuba multiplication. After then the authors evaluate the built-in AES encryption, which improves the performance of AES–GCM by about 11 times than that of ARMv7, which does not support AES and polynomial multiplication with hardware accelerators. In [12], authors evaluated the PMULL based binary field multiplication techniques ranging from 192-bit to 576-bit for ECC. From 256-bit polynomial multiplication, Karatsuba multiplications show higher performance than traditional approaches. However, the paper does not explore the full implementations of ECC with proposed binary field multiplication and we found a room to improve the performance further from the work.

In this paper, we present efficient implementation techniques for B-571 on ARMv8. We improve the previous binary field multiplication by introducing finely aligned multiplication and incomplete reduction technique. The proposed technique improves the performance by a factor of 1.34 times than previous Seo et al.'s implementations [12]. For the point addition and doubling, we perform the combined reduction on special types of binary field multiplication, squaring and addition operations. The scalar multiplication is implemented in window method, which ensures constant timing and security against timing attacks. Finally, we set the speed record for B-571 on ARMv8, which performs the unknown/fixed scalar multiplications within 759,630/331,944 clock cycles, respectively.

The remainder of this paper is organized as follows. In Section 2, we recap the B-571 curve, target ARM processor and previous polynomial multiplication on ARMv8. In Section 3, we propose the efficient ECC implementations on ARMv8. In Section 4, we evaluate the performance of proposed methods. Finally, Section 5 concludes the paper.

2. Related Works

2.1. Elliptic curve over $\mathbb{F}_{2^{571}}$

The 571-bit elliptic curve standardized in [1] and the finite field $\mathbb{F}_{2^m}$ is defined by:

$$f(x) = x^{571} + x^{10} + x^5 + x^2 + 1$$

The curve $E : y^2 + xy = x^3 + ax^2 + b$ over $\mathbb{F}_{2^m}$ is defined by:

$a =$ 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000001

$b =$ `02F40E7E 2221F295 DE297117 B7F3D62F 5C6A97FF CB8CEFF1 CD6BA8CE 4A9A18AD 84FFABBD 8EFA5933 2BE7AD67 56A66E29 4AFD185A 78FF12AA 520E4DE7 39BACA0C 7FFEFF7F 2955727A`

and group order is defined by:

$n =$ `03FFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF E661CE18 FF559873 08059B18 6823851E C7DD9CA1 161DE93D 5174D66E 8382E9BB 2FE84E47`

2.2. ARM Processor

Advanced RISC Machine (ARM) is an Instruction Set Architecture (ISA) for high-performance embedded applications. ARM architecture has nice properties including low-power consumption and high code density. The most advanced ARMv8 processor supports both 32-bit (AArch32) and 64-bit (AArch64) architectures. Particularly the processor supports a single-instruction multiple-data (SIMD) instruction sets namely NEON engine. The processor has 32 64-bit registers (`X0-X31`) and 32 128-bit NEON registers (`V0-V31`). Particularly, 64-bit wise polynomial multiplication instructions (`PMULL` and `PMULL2`) are available. The `PMULL` instruction uses the lower 64-bit part in 128-bit register for the input, while the `PMULL2` instruction uses the higher 64-bit part in 128-bit register for the input [3].

2.3. Polynomial Multiplication on ARMv8

In [12], authors evaluated the `PMULL` instructions for the various polynomial multiplications ranging from 192-bit to 576-bit. Particularly, the authors perform the three terms of Karatsuba multiplication for 576-bit case, which reduces the number of 192-bit wise multiplication from 9 to 6 [7,14,5]. The author claims that basic approach for 192-bit case is more efficient than Karatsuba multiplication on the ARMv8 architecture, since additional number of addition operations are larger than optimized multiplication operations. The detailed program codes are drawn in Algorithm 1. The approach requires the 9 64-bit wise polynomial multiplications. The partial products ($A[0] \times B[1]$, $A[1] \times B[0]$, $A[1] \times B[2]$, $A[2] \times B[1]$) are computed and shifted by 64-bit. The shifted results are accumulated to the intermediate results for partial products ($A[0] \times B[0]$, $A[0] \times B[2]$, $A[1] \times B[1], A[2] \times B[0], A[2] \times B[2]$).

3. Proposed Method

3.1. Optimization for Finite Field Operation

The polynomial addition/subtraction can be performed with bit-wise exclusive-or instructions on both operands. For the 576-bit case, each operand is loaded to the 5 128-bit

Algorithm 1 192-bit Polynomial Multiplication in Program Codes [12]

```
Require: 192-bit operands $A[2 \sim 0]$ (v0, v1) and $B[2 \sim 0]$ (v5, v6).
Ensure: 384-bit result $C[5 \sim 0] \leftarrow A[2 \sim 0] \times B[2 \sim 0]$ (v10, v11, v12).
1: pmull v10.1q, v0.1d, v5.1d          {$A[0] \times B[0]$}
2: pmull v11.1q, v0.1d, v6.1d          {$A[0] \times B[2]$}
3: pmull2 v28.1q, v0.2d, v5.2d         {$A[1] \times B[1]$}
4: eor.16b v11, v11, v28
5: pmull v28.1q, v1.1d, v5.1d          {$A[2] \times B[0]$}
6: eor.16b v11, v11, v28
7: pmull v12.1q, v1.1d, v6.1d          {$A[2] \times B[2]$}
8: ext.16b v30, v0, v0, #8
9: pmull2 v29.1q, v30.2d, v5.2d        {$A[0] \times B[1]$}
10: pmull v28.1q, v30.1d, v5.1d        {$A[1] \times B[0]$}
11: eor.16b v29, v29, v28
12: pmull v30.1q, v30.1d, v6.1d        {$A[1] \times B[2]$}
13: ext.16b v28, v1, v1, #8
14: pmull2 v28.1q, v28.2d, v5.2d       {$A[2] \times B[1]$}
15: eor.16b v30, v30, v28
16: ext.16b v28, v31, v29, #8
17: ext.16b v29, v29, v30, #8
18: ext.16b v30, v30, v31, #8
19: eor.16b v10, v10, v28
20: eor.16b v11, v11, v29
21: eor.16b v12, v12, v30
```

NEON registers ($5 = \lceil 4.5 \rceil = \lceil \frac{576}{128} \rceil$) and 5 times of bit-wise exclusive-or operations are performed.

The binary field multiplication is the most expensive operation in the finite field operations. For 576-bit case, Seo et al. proposed the three-term of Karatsuba multiplication [12]. Each term performs the classical 192-bit wise polynomial multiplication (See Algorithm 1). The 192-bit multiplication always outputs the 384-bit results in 3 consecutive 128-bit registers. However, this alignment style requires additional 64-bit wise shift operations to get aligned intermediate results in three steps for 576-bit multiplication (See Step 4, 8, 10 in Algorithm 2). In order to hide these latencies, we used both previous and shifted 192-bit polynomial multiplication. In Algorithm 3, shifted version of multiplication is described. Unlike previous approach described in Algorithm 1, the partial products ($A[0] \times B[0]$, $A[0] \times B[2]$, $A[1] \times B[1]$, $A[2] \times B[0]$, $A[2] \times B[2]$) are shifted by 64-bit. The shifted results are accumulated to the intermediate results for partial products ($A[0] \times B[1]$, $A[1] \times B[0]$, $A[1] \times B[2]$, $A[2] \times B[1]$). The 64-bit shifted results are stored into 4 consecutive NEON registers (`v16, v17, v18, v19`) where the least significant 64-bit of `v16` and most significant 64-bit of `v19` are set to zero. The detailed 576-bit multiplication is described in Algorithm 2. The 576-bit polynomial multiplication requires 6 192-bit polynomial multiplications in Step 3, 4, 5, 8, 9 and 10. The results are required to be shifted by 192, 384 or 576-bit to the left before intermediate result are accumulated. The 384-bit shift case does not require additional shift operations on the 128-bit register. However, two cases (192 and 576-bit) requires 64-bit wise shift to the left to align the results (Step 4, 8, 10). In this case, we used the shifted 192-bit polynomial multiplication

Algorithm 2 Aligned Polynomial Multiplication for 576-bit

```
Require: 576-bit operands $A[8 \sim 0]$ and $B[8 \sim 0]$.
Ensure: 1152-bit result $C[17 \sim 0] \leftarrow A[8 \sim 0] \times B[8 \sim 0]$.
 1: $A \leftarrow \{A_H, A_M, A_L\} \leftarrow \{(A[8], A[7], A[6]), (A[5], A[4], A[3]), (A[2], A[1], A[0])\}$
 2: $B \leftarrow \{B_H, B_M, B_L\} \leftarrow \{(B[8], B[7], B[6]), (B[5], B[4], B[3]), (B[2], B[1], B[0])\}$
 3: $C_H \leftarrow (A_H \times_{192} B_H) \ll 384$                                  { Algorithm 1}
 4: $C_M \leftarrow (A_M \times_{192} B_M) \ll 192$                                  { Algorithm 3}
 5: $C_L \leftarrow A_L \times_{192} B_L$                                            { Algorithm 1}
 6: $T \leftarrow C_H \oplus C_M \oplus C_L$
 7: $C \leftarrow T \oplus (T \ll 192) \oplus (T \ll 384)$
 8: $C_H \leftarrow ((A_H \oplus A_M) \times_{192} (B_H \oplus B_M)) \ll 576$        { Algorithm 3}
 9: $C_M \leftarrow ((A_H \oplus A_L) \times_{192} (B_H \oplus B_L)) \ll 384$        { Algorithm 1}
10: $C_L \leftarrow ((A_M \oplus A_L) \times_{192} (B_M \oplus B_L)) \ll 192$        { Algorithm 3}
11: $C \leftarrow C_H \oplus C_M \oplus C_L \oplus C$
```

Algorithm 3 (Shifted) 192-bit Polynomial Multiplication in Program Codes

```
Require: 192-bit operands $A[2 \sim 0]$ (v1, v2) and $B[2 \sim 0]$ (v6, v7).
Ensure: 384-bit result $C[5 \sim 0] \leftarrow A[2 \sim 0] \times B[2 \sim 0]$ (v16, v17, v18, v19).
 1: pmull v17.1q, v1.1d, v6.1d          {$A[0] \times B[0]$}
 2: pmull v18.1q, v1.1d, v7.1d          {$A[0] \times B[2]$}
 3: pmull2 v28.1q, v1.2d, v6.2d         {$A[1] \times B[1]$}
 4: eor.16b v18, v18, v28
 5: pmull v28.1q, v2.1d, v6.1d          {$A[2] \times B[0]$}
 6: eor.16b v18, v18, v28
 7: pmull v19.1q, v2.1d, v7.1d          {$A[2] \times B[2]$}
 8: ext.16b v16, v31, v17, #8
 9: ext.16b v17, v17, v18, #8
10: ext.16b v18, v18, v19, #8
11: ext.16b v19, v19, v31, #8
12: ext.16b v30, v1, v1, #8
13: pmull2 v29.1q, v30.2d, v6.2d        {$A[0] \times B[1]$}
14: pmull v28.1q, v30.1d, v6.1d         {$A[1] \times B[0]$}
15: eor.16b v29, v29, v28
16: pmull v30.1q, v30.1d, v7.1d         {$A[1] \times B[2]$}
17: ext.16b v28, v2, v2, #8
18: pmull2 v28.1q, v28.2d, v6.2d        {$A[2] \times B[1]$}
19: eor.16b v30, v30, v28
20: eor.16b v17, v17, v29
21: eor.16b v18, v18, v30
```

described in Algorithm 3. For the other three cases (Step 3, 5, 9), we used the previous approach described in Algorithm 1. By using shifted approach, we can avoid three times of 64-bit wise shift operations in each multiplication. In instruction set level, 12 times of extraction instructions are optimized.

The polynomial squaring is a linear operation, since the result is obtained by inserting a 0 bit between consecutive bits of operand. By using the 64-bit polynomial multi-

Algorithm 4 576-bit Polynomial Squaring

```
Require: 576-bit Operand $A[8 \sim 0]$.
Ensure: 1152-bit Result $C[17 \sim 0] \leftarrow A[8 \sim 0] \times A[8 \sim 0]$.
1: for i = 0 to 8 by 1 do
2:     $\{C[2 \times i + 1] || C[2 \times i]\} \leftarrow A[i] \times A[i]$
3: end for
```

Algorithm 5 Fast Reduction over $\mathbb{F}_{2^{571}}$

```
Require: 576-bit (complete) or 1152-bit (incomplete) operands $A$, complete reduction.
Ensure: 571-bit (complete) or 576-bit (incomplete) result $C$.
1: if complete reduction then
2:     $r \leftarrow$ 0x425
3:     $A_L \leftarrow A \bmod 2^{571}$
4:     $A_H \leftarrow A \operatorname{div} 2^{571}$
5:     $T \leftarrow A_H \times r$
6:     $C \leftarrow A_L \oplus T$
7: else
8:     $r \leftarrow$ 0x84A0
9:     $A_L \leftarrow A \bmod 2^{576}$
10:    $A_H \leftarrow A \operatorname{div} 2^{576}$
11:    $T \leftarrow A_H \times r$
12:    $T \leftarrow A_L \oplus T$
13:    $T_L \leftarrow T \bmod 2^{576}$
14:    $T_H \leftarrow T \operatorname{div} 2^{576}$
15:    $T \leftarrow T_H \times r$
16:    $C \leftarrow T_L \oplus T$
17: end if
```

plication (PMULL) instruction, we can compute the 64-bit wise squaring with PMULL and PMULL2 instructions, since each instruction outputs only half results at a time. In Algorithm 4, the 576-bit wise squaring operation is drawn. The 576-bit operand requires 18 ($\lceil \frac{571}{64} \rceil \times 2$) times of PMULL instructions.

The m-bit polynomial multiplication/squaring operations produce the values of degree at most $2m$-bit, which must be reduced by modulo. When the modulo is smaller than operand size (64-bit) of PMULL instruction, we can perform the multiplication on higher parts ($> m$) by modulo. The modulo of binary field $\mathbb{F}_{2^{571}}$ is defined by $f(x) = x^{571} + x^{10} + x^5 + x^2 + 1$, which is only 11-bit modulo so we can use 64-bit wise PMULL instruction for polynomial multiplication. However, 571-bit modulo is not efficient over the 64-bit machine since this requires 5-bit wise shift operations to align the results. Alternatively, we choose the 64-bit machine friendly modulo ($f(x) = x^{576} + x^{15} + x^{10} + x^7 + x^5$) and incomplete reduction. This approach avoids the number of 5-bit wise shift operations and complete results are also obtained by performing the complete reduction before outputting the results. The detailed reduction process is available in Algorithm 5. If the complete reduction is selected, the modulo (r) is set to 0x425 representing the values ($x^{10} + x^5 + x^2 + 1$). In Step 3, the part of A which is lower than 571-bit is extracted to A_L. In Step 4, the part of A which is higher than 571-bit is extracted to A_H. In Step 5, the higher part (A_H) is multiplied by modulus (r). In Step 6, the results are added to the lower part (A_L). In case of incomplete reduction, the modulus (r) is set to 0x84A0 representing the values ($x^{15} + x^{10} + x^7 + x^5$). In Step 9, the part of A which is lower than 576-bit is extracted to A_L. In Step 10, the part of A which is higher than 576-bit is extracted to A_H. In Step 11, the higher part (A_H) is multiplied by modulus (r). In Step 12, the lower part (A_L) are added to the intermediate results T. In Step 13, the part of T which is lower than 576-bit is extracted to T_L. In Step 14, the part of T which is higher than 576-bit is extracted to T_H. In Step 15, the higher part (T_H) is multiplied by modulus (r). In Step 16, the lower part (T_L) are added to the intermediate results T.

For fast and secure inversion operation, we used the Itoh-Tsujii algorithm [4], which is an optimization of inversion through Fermat's little theorem ($f(x)^{-1} = f(x)^{2^m-2}$), ensuring the constant time computations. The algorithm uses a repeated field squaring and multiplication operations for $f(x)^{2^k}$, which follows a chains of multiplication and squaring sequences ($f_1 \rightarrow f_2 \rightarrow f_4 \rightarrow f_8 \rightarrow f_{16} \rightarrow f_{17} \rightarrow f_{34} \rightarrow f_{35} \rightarrow f_{70} \rightarrow f_{71} \rightarrow f_{142} \rightarrow f_{284} \rightarrow f_{285} \rightarrow f_{570}$). The inversion algorithm requires 13 multiplication and 570 squaring operations.

3.2. *Optimization for Scalar Multiplication*

In order to perform the scalar multiplication, the point addition and doubling operations are required, which consist of a number of finite field operations. Depending on specific coordinates, the number of finite field operations are varied each other. The point addition in López-Dahab/affine coordinates requires 8 multiplication (`M`), 5 squaring (`S`) and 1 a-multiplication (`a-M`). Alternative point addition in López-Dahab coordinates requires 13`M` and 5`S`. For the point doubling in López-Dahab coordinates requires 3`M`, 5`S`, 1`a-M` and 1 b-multiplication (`b-M`). Particularly, the variable (a) is set to 1 in the B-571 curve so the `a-M` operation is free. The binary field multiplication and squaring operations are performed by following the implementation techniques described in Section 3.1. A sequence of multiplication, squaring and addition operations are optimized again by combining the reduction operations . This sequence of field operations involve a type ($A \times B + C \times D$). The straight-forward implementation of type requires 2 multiplication, 2 reduction and 1 addition operations. One reduction operation can be optimized by performing the multiplication and addition operations in advance [8]. Similar a type ($A^2 + C \times D$) is also optimized from 1 squaring, 1 multiplication, 2 reduction and 1 addition operations to 1 squaring, 1 multiplication, 1 reduction and 1 addition operations. We employed the Negre and Robert techniques for the point addition in López-Dahab/affine coordinates and doubling in López-Dahab coordinates. For point addition in López-Dahab/affine coordinates described in Algorithm 6, Step 13 and 19 can be optimized through optimal ($A^2 + C \times D$) and ($A \times B + C \times D$) types. For point doubling in López-Dahab coordinates described in Algorithm 7, Step 12 can be optimized through optimal ($A \times B + C \times D$) type. We extended this technique to point addition in López-Dahab coordinates in Algorithm 8. The Step 17, 18 and 20 include the ($A \times B + C \times D$) type and this approach optimizes the 3 reduction operations in each point addition operation.

The scalar multiplication is implemented in window method. This algorithm always performs the point addition and doubling operations in each bit and our implementations of finite field arithmetic are also regular fashion, which ensure constant-time computation and security against Simple Power Analysis (SPA). For unknown point, we used point addition/doubling in López-Dahab coordinates with window methods and for fixed point we used point addition in López-Dahab/affine coordinates and doubling in López-Dahab coordinates with window methods.

Algorithm 6 Optimization for Point Addition in López-Dahab/affine coordinates [8]

```
Require: Point $P1$ $(X1, Y1, Z1)$ in López-Dahab coordinates and $P2$ $(X2, Y2, 1)$ in affine coordinates
Ensure: Point $P3$ $(X3, Y3, Z3)$ in López-Dahab coordinates
1: $t0 \leftarrow Z1^2$
2: $t1 \leftarrow Y2 \times t0$
3: $k0 \leftarrow Y1 + t1$
4: $t2 \leftarrow X2 \times Z1$
5: $k1 \leftarrow X1 + t2$
6: $k2 \leftarrow k1 \times Z1$
7: $Z3 \leftarrow k2^2$
8: $k4 \leftarrow X2 \times Z3$
9: $t3 \leftarrow k1^2$
10: $t4 \leftarrow a \times k2$
11: $t5 \leftarrow k0 + t3$
12: $t6 \leftarrow t5 + t4$
13: $X3 \leftarrow k0^2 + k2 \times t6$   $\{A^2 + C \times D\}$
14: $t7 \leftarrow k0 \times k2$
15: $t8 \leftarrow k4 + X3$
16: $t9 \leftarrow t7 + Z3$
17: $t10 \leftarrow Y2 + X2$
18: $t11 \leftarrow Z3^2$
19: $Y3 \leftarrow t10 \times t11 + t8 \times t9$   $\{A \times B + C \times D\}$
```

Algorithm 7 Optimization for Point Doubling in López-Dahab coordinates [8]

```
Require: Point P1 (X1, Y1, Z1) in López-Dahab coordinates
Ensure: Point P3 (X3, Y3, Z3) in López-Dahab coordinates
1: $k0 \leftarrow Z1^2$
2: $t0 \leftarrow k0^2$
3: $k1 \leftarrow b \times t0$
4: $k2 \leftarrow X1^2$
5: $Z3 \leftarrow A \times k2$
6: $t1 \leftarrow k2^2$
7: $X3 \leftarrow t1 + k1$
8: $t2 \leftarrow Y1^2$
9: $t3 \leftarrow a \times Z3$
10: $t4 \leftarrow t2 + t3$
11: $t5 \leftarrow t4 + k1$
12: $Y3 \leftarrow t5 \times X3 + Z3 \times k1$   $\{A \times B + C \times D\}$
```

Algorithm 8 Optimization for Point Addition in López-Dahab coordinates [8]

```
Require: Point $P1$ $(X1, Y1, Z1)$ and $P2$ $(X2, Y2, Z2)$ in López-Dahab coordinates
Ensure: Point $P3$ $(X3, Y3, Z3)$ in López-Dahab coordinates
1: $k0 \leftarrow X1 \times Z2$
2: $k1 \leftarrow X2 \times Z1$
3: $k2 \leftarrow k0^2$
4: $k3 \leftarrow k1^2$
5: $k4 \leftarrow k0 + k1$
6: $k5 \leftarrow k2 + k3$
7: $t0 \leftarrow Z2^2$
8: $k6 \leftarrow Y1 \times t0$
9: $t1 \leftarrow Z1^2$
10: $k7 \leftarrow Y2 \times t1$
11: $k8 \leftarrow k6 + k7$
12: $k9 \leftarrow k8 \times k4$
13: $t2 \leftarrow Z1 \times Z2$
14: $Z3 \leftarrow k5 \times t2$
15: $t3 \leftarrow k7 \times k3$
16: $t4 \leftarrow k2 \times k6$
17: $X3 \leftarrow k1 \times t4 + k0 \times t3$   $\{A \times B + C \times D\}$
18: $t5 \leftarrow k5 \times k6 + k0 \times k9$   $\{A \times B + C \times D\}$
19: $t6 \leftarrow k9 + Z3$
20: $Y3 \leftarrow t6 \times X3 + t5 \times k5$   $\{A \times B + C \times D\}$
```

Table 1. Comparison results of binary field multiplication for B-571 curve

Algorithm	Clock cycle
Seo et al. [12]	132
Proposed Method	99

Table 2. Performance evaluations of B-571 curve, where w is window size

Operation	Clock cycle
Binary Field Operation	
Multiplication	99
Squaring	24
Inversion	31,232
Group Operation	
Point addition (LD/affine)	1,107
Point addition (LD)	1,537
Point doubling (LD)	609
Scalar Multiplication	
Unknown point ($w = 4$)	759,630
Fixed point ($w = 4$)	331,944

4. Evaluation

We used Xcode (ver 6.3.2) as a development IDE and programmed over iPad Mini2 (iOS 8.4). The iPad Mini2 equipped Apple A7 with 64-bit architecture operated in the frequency of 1.3GHz. The program is written in C and assembly codes and complied with `-Ofast` optimization level. The timing are acquired through the clock cycles of real device.

In Table 1, the comparison results of binary field multiplication over B-571 curve are drawn. We only compared results with Seo et al. [12] since SUPERCOP benchmark tool does not support the iOS operating system which is required for our experiments and the work by Gouvêa and J. López is only provide the GCM operations [3]. The Seo et al. achieved the high performance with three-term of Karatsuba multiplication for 576-bit polynomial multiplication and fast reduction techniques. In our implementation, we further improved performance by a factor of 1.34 times with the finely aligned multiplications and incomplete reduction techniques.

In Table 2, we listed the whole results of B-571 implementations. Unfortunately, there is no paper about ECC implementations on ARMv8. We only provide our results. The squaring operation is linear computations, which requires small number of clock cycles. The inversion operation is implemented in Fermat's little theorem, which requires 570 squaring and 13 multiplication operations. For group operations, three different point operations are evaluated. The doubling in López-Dahab coordinates shows the lowest clock cycles. The point addition in López-Dahab coordinates shows the highest clock cycles. Finally, the scalar multiplication is efficiently implemented with window methods. In the fixed point, points can be pre-computed and the number of doubling operations are optimized. In this paper, we explore the medium window size ($w = 4$) but this can be easily extended to the long window size ($w > 4$) by sacrificing the RAM storages.

5. Conclusion

In this paper, we show efficient finite field and group operations for B-571 implementations over ARMv8. We optimized the binary field arithmetics by introducing the several optimization techniques. The group operations are also improved by reducing the number of reduction operations in point addition and doubling operations. Finally, we achieved the high speed implementation of B-571 implementation over ARMv8.

References

[1] Recommended elliptic curve domain parameters. *Standards for Efficient Cryptography Group, Certicom Corp*, 2000.

[2] D. Câmara, C. P. Gouvêa, J. López, and R. Dahab. Fast software polynomial multiplication on ARM processors using the NEON engine. In *International Conference on Availability, Reliability, and Security*, pages 137–154. Springer, 2013.

[3] C. P. Gouvêa and J. López. Implementing GCM on ARMv8. In *Cryptographers Track at the RSA Conference*, pages 167–180. Springer, 2015.

[4] T. Itoh and S. Tsujii. A fast algorithm for computing multiplicative inverses in GF(2m) using normal bases. *Information and computation*, 78(3):171–177, 1988.

[5] A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. In *Soviet physics doklady*, volume 7, page 595, 1963.

[6] J. López and R. Dahab. High-speed software multiplication in GF(2^m). In *International Conference on Cryptology in India*, pages 203–212. Springer, 2000.

[7] P. L. Montgomery. Five, six, and seven-term Karatsuba-like formulae. *IEEE Transactions on Computers*, 54(3):362–369, 2005.

[8] C. Negre and J.-M. Robert. Impact of optimized field operations AB, AC and AB+CD in scalar multiplication over binary elliptic curve. In *International Conference on Cryptology in Africa*, pages 279–296. Springer, 2013.

[9] L. B. Oliveira, D. F. Aranha, C. P. Gouvêa, M. Scott, D. F. Câmara, J. López, and R. Dahab. TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. *Computer Communications*, 34(3):485–493, 2011.

[10] H. Seo, Y. Lee, H. Kim, T. Park, and H. Kim. Binary and prime field multiplication for public key cryptography on embedded microprocessors. *Security and Communication Networks*, 7(4):774–787, 2014.

[11] H. Seo, Z. Liu, J. Choi, and H. Kim. Karatsuba–block-comb technique for elliptic curve cryptography over binary fields. *Security and Communication Networks*, 8(17):3121–3130, 2015.

[12] H. Seo, Z. Liu, Y. Nogami, J. Choi, and H. Kim. Binary field multiplication on ARMv8. *Security and Communication Networks*, 2016.

[13] M. Shirase, Y. Miyazaki, T. Takagi, and H. Dong-Guk. Efficient implementation of pairing-based cryptography on a sensor node. *IEICE transactions on information and systems*, 92(5):909–917, 2009.

[14] A. Weimerskirch and C. Paar. Generalizations of the Karatsuba algorithm for efficient implementations. *IACR Cryptology ePrint Archive*, 2006:224, 2006.

A Systems Approach to Cyber Security
A. Roychoudhury and Y. Liu (Eds.)

doi:10.3233/978-1-61499-744-3-107

Scholarly Digital Libraries as a Platform for Malware Distribution

Nir Nissim[1], Aviad Cohen[1], Jian Wu[2], Andrea Lanzi[3], Lior Rokach[1], Yuval Elovici[1] and Lee Giles[2]
[1]*The Malware Lab at the Cyber Security Research Center (CSRC), Ben-Gurion University*
[2]*The Pennsylvania State University*
[3]*University of Milano*

Abstract. Researchers from academic institutions and the corporate sector rely heavily on scholarly digital libraries for accessing journal articles and conference proceedings. Primarily downloaded in the form of PDF files, there is a risk that these documents may be compromised by attackers. PDF files have many capabilities that have been widely used for malicious operations. Attackers increasingly take advantage of innocent users who open PDF files with little or no concern, mistakenly considering these files safe and relatively non-threatening. Researchers also consider scholarly digital libraries reliable and home to a trusted corpus of papers and untainted by malicious files. For these reasons, scholarly digital libraries are an attractive target for cyber-attacks launched via PDF files. In this study, we present several vulnerabilities and practical distribution attack approaches tailored for scholarly digital libraries. To support our claim regarding the attractiveness of scholarly digital libraries as an attack platform, we evaluated more than two million scholarly papers in the CiteSeerX library that were collected over 8 years and found it to be contaminated with a surprisingly large number (0.3%-2%) of malicious scholarly PDF documents, the origin of which is 46 different countries spread worldwide. More than 55% of the malicious papers in CiteSeerX were crawled from IP's belonging to USA universities, followed by those belonging to Europe (33.6%). We show how existing scholarly digital libraries can be easily leveraged as a distribution platform both for a targeted attack and in a worldwide manner. On average, a certain malicious paper caused high impact damage as it was downloaded 167 times in 5 years by researchers from different countries worldwide. In general, the USA and Asia downloaded the most malicious scholarly papers, 40.15% and 27.9%, respectively. The top malicious scholarly document downloaded is a malicious version of a popular paper in the computer forensics domain, with 2213 downloads in a worldwide coverage of 108 different countries. Finally, we suggest several concrete solutions for mitigating such attacks, including simple deterministic solutions and also advanced machine learning-based frameworks.

Keywords Scholarly, Digital, Library, Paper, PDF, Malware, Malicious, Attack, Distribution.

1. Introduction

The number of scholarly documents (English-language) accessible on the Web is enormous, estimated at 114 million PDF documents in 2014 [9], of which over 27 million (~24%) can be freely accessed without payment or subscription [9]. These documents are freely accessible in part because researchers publish draft versions of their papers on their professional homepages (often within the domains of universities), before the final versions are published by the publishers. Researchers also publish their research on their homepages to increase exposure, reach researchers around the world, and gain citations and recognition for their work [10], [11]. In order to assist researchers, many scholarly digital libraries and search engines collect and index the author's version. Thus, the papers can be freely downloaded worldwide. This free collection of scholarly documents is a valuable resource for most researchers and academics who may not have a comprehensive subscription to all publishers' content.

Figure 1 presents a snapshot of search results for a searched paper using Google Scholar. At the bottom of the page, one can access all 15 versions of the paper, already indexed by Google Scholar, simply by clicking on the blue "All 15 versions" link; thereby, free and convenient versions, are literally at the user's fingertips, as seen in Figure 2.

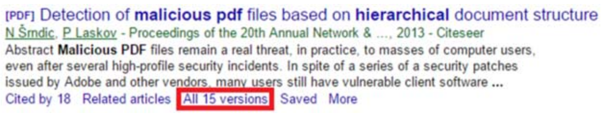
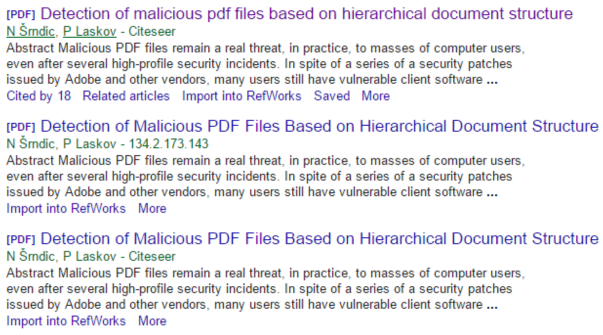

[PDF] Detection of **malicious pdf** files based on **hierarchical** document structure
N Šrndic, P Laskov - Proceedings of the 20th Annual Network & ..., 2013 - Citeseer
Abstract **Malicious PDF** files remain a real threat, in practice, to masses of computer users, even after several high-profile security incidents. In spite of a series of security patches issued by Adobe and other vendors, many users still have vulnerable client software ...
Cited by 18 Related articles All 15 versions Saved More

Figure 1. Google Scholar's search results for a given academic paper, including 14 additional versions of the paper.


[PDF] Detection of malicious pdf files based on hierarchical document structure
N Šrndic, P Laskov - Citeseer
Abstract Malicious PDF files remain a real threat, in practice, to masses of computer users, even after several high-profile security incidents. In spite of a series of security patches issued by Adobe and other vendors, many users still have vulnerable client software ...
Cited by 18 Related articles Import into RefWorks Saved More

[PDF] Detection of Malicious PDF Files Based on Hierarchical Document Structure
N Šrndic, P Laskov - 134.2.173.143
Abstract Malicious PDF files remain a real threat, in practice, to masses of computer users, even after several high-profile security incidents. In spite of a series of security patches issued by Adobe and other vendors, many users still have vulnerable client software ...
Import into RefWorks More

[PDF] Detection of Malicious PDF Files Based on Hierarchical Document Structure
N Šrndic, P Laskov - Citeseer
Abstract Malicious PDF files remain a real threat, in practice, to masses of computer users, even after several high-profile security incidents. In spite of a series of security patches issued by Adobe and other vendors, many users still have vulnerable client software ...
Import into RefWorks More

Figure 2. Some of the additional versions of the searched paper, including those available for free.

Researchers heavily use scholarly digital libraries to access and download scholarly documents. For example, according to a survey by EBLIDA[1], the total number of academic libraries in Europe is 5,974; however, this number is far from complete given that it is based on information provided by only 25 countries participating in the survey. Nevertheless, the number of registered users of these libraries is 39,328,294. As Europe represents only part of the world's research activity, the global use of scholarly digital libraries is much higher.

Universities are considered to be highly reputable institutions that primarily focus on research and the goal of which is to contribute new and valuable knowledge to the world. Therefore, they are considered a trusted content source without malicious intent. Correspondingly, the Websites of their academics and researchers (which reside on the institution's network domain) are also considered to contain only trusted content, free of

[1] http://www.eblida.org/activities/kic/academic-libraries-statistics.html

malicious PDF files. In a circular fashion, academic digital libraries tend to harvest these allegedly trusted sites without hesitation or fear and therefore do not even scan them to detect malicious content[2]. In addition, this reputation as sources of trusted scholarly documents makes digital libraries an attractive platform from which to take advantage of and distribute malicious PDF files; researchers' Webpages have become a target that can be used to launch attacks[3]. In addition, researchers, professors, and research students are naturally attractive candidates for attack, because, due to the nature of their work, they have access to confidential and sensitive information, such as nuclear knowledge, medical records [32][33], aviation, and educational records and materials (student data, exams). Moreover, researchers collaborate with governmental agencies and industry, which allows them access to national and confidential information from governments (such as computational criminology), national institutions, and companies (such as strategic information).

Recent studies have presented many methods of improving the detection of malicious PDF files [1], [2]. These studies focused on detection techniques based on analyzing the malicious PDF files when they have already been downloaded to the host machine. To the best of our knowledge, no study addressed the issue at the stage one step before downloading, a step at which it might be possible to prevent malicious PDF files from being mass-distributed through legitimate channels and exiting platforms, and thus, markedly improve the detection of malicious PDF files, including those found on popular, well-known, and extensively used sources of PDF files, such as scholarly digital libraries. These libraries can be intentionally used as a free and very successful platform for distributing PDF malware quickly and easily to a desired group of victims with access to valuable information. An academic paper arouses little suspicion, particularly if an attacker wants to distribute a new 0-day attack quickly in the shape of a benign PDF file. 0-day attack[4] utilizes new attack techniques or new vulnerabilities[5] that are difficult to detect, particularly by the antivirus tools commonly used by organizations such as universities and academic digital libraries for scanning PDF files. Thus, these libraries can easily be used as a new and convenient platform for distributing 0-day attacks. The contributions of our paper include:

1. A demonstration of the vulnerability of digital libraries and also an estimation of the extent of malicious use of scholarly digital libraries. Specifically, we perform a retro-perspective analysis of the papers that were collected by CiteseerX over a period of eight years. Using current antiviruses, we can assess which paper contained malware when it was indexed.
2. An evaluation of the impact damage of malicious documents published in a scholarly digital library.
3. Additional distribution attack approaches that can be used by attackers to leverage these digital libraries.

In addition to the above contributions we suggest methods for mitigating the problem we identified:

[2] According to CiteseerX team which are part of the authors in this paper.

[3] http://www.nytimes.com/2013/07/17/education/barrage-of-cyberattacks-challenges-campus-culture.html?pagewanted=all&_r=0.

[4] http://www.bullguard.com/bullguard-security-center/pc-security/computer-threats/what-are-zero-day-attacks.aspx

[5] http://www.pctools.com/security-news/zero-day-vulnerability/

- Compatibility check of PDF files so they can be correctly opened by users before their publication. (96.5% of malicious PDF files are incompatible).
- Re-check of re-uploaded PDF Files
- Periodic review of a library's files in the library when a new PDF malware/vulnerability-exploitation is identified.
- Machine learning-based methods for enhancing the detection of malicious PDF files.

2. Background

As indicated previously, the Web contains more than 114 million scholarly documents [9], and this number represents a significant attack power for adversaries who want to utilize the fact that scholarly digital libraries are considered trusted and that their content (PDF files) is used and downloaded by many users worldwide. In order to grasp the potential harm that can be done by malicious PDF files existing in a scholarly digital library, we briefly present targeted attacks through scholarly digital libraries using malicious PDF files. Then, we present the possible attacks that can be launched by a malicious PDF file mistakenly considered a benign scholarly document, and the techniques used to achieve this. Thereby, we aim to raise the awareness of scholarly digital libraries, as well as of innocent researchers and readers, of the power of a malicious PDF file, so that they will improve their security level.

2.1. Targeted Attacks via Scholarly Digital Libraries using Malicious PDF Files

Sophisticated attackers interested in sensitive and novel knowledge about a specific domain, such as nuclear energy, can launch a targeted attack by inserting an attractive, yet malicious, paper that addresses nuclear energy into digital libraries, engaging and tempting researchers to download the paper. It is noteworthy that the attacker does not need to be a co-author of the paper. Our investigation showed that most scholarly digital libraries (such as Google Scholar) crawl academic Websites and index the papers they find, disregarding any mismatches between the author's affiliation and the Website that stores the paper. Thus, an attacker can take a popular paper written by someone else, inject malicious code into it, and upload it to a Website. When the victim opens the malicious PDF file, a malicious code will be executed in the computer. This malicious code will allow the attacker to extract data from the victim's machine and send it to a remote server controlled by the attacker.

This attack is within the realm of reality, for the previously mentioned reasons, as well as because users consider non-executable files safer than executables, and thus are less suspicious of PDF files, especially when downloaded from popular and trusted scholarly sources. Unfortunately, non-executable files such as PDF files are as dangerous as executable files, since their readers can contain vulnerabilities that, when exploited, can allow an attacker to execute malicious actions on the victim's computer. F-Secure's 2008-2009 report[6] indicates that the most popular file types for targeted attacks in 2008-2009 were PDF and Microsoft Office files. Note that since that time, the number of targeted attacks on Adobe Reader has almost doubled. In the following section, we elaborate on several of the most common techniques and attacks involving the use of malicious PDF files.

[6] http://www.f-secure.com/weblog/archives/00001676.html

To demonstrate the damage that can be caused by malicious PDF files, we refer to a famous incident involving the Israeli Ministry of Defense (IMOD) that took place on January 15, 2014, which provides an example of a new type of targeted cyber-attack. According to various media reports[7] published on January 26, 2014, the Seculert[8] Company reported that it had identified an attack in which attackers sent email messages, allegedly from the IMOD, with a malicious PDF file attachment posing as an IMOD document. When opened, the PDF file installed a Trojan horse that enabled the attacker to take control of the computer.

2.2. Possible Attack Techniques using PDF Files

Before explaining how scholarly digital libraries can be easily used as a platform to leverage and distribute attacks worldwide, we now present some of the many ways PDF files can be used maliciously when created or manipulated by an attacker.

JavaScript code

PDF files may contain embedded JavaScript code or code retrieved from URIs [5], including 3D content, form validation, and calculations. Typically, a malicious JavaScript code in a PDF file attempts to exploit a vulnerability in the PDF viewer in order to divert the normal execution flow to the embedded malicious code. This is achieved by a heap spraying[9] attack. JavaScript also allows the download of an executable file that may contain malicious content. Alternatively, JavaScript code can access Websites, whether malicious or benign.

Code obfuscation and filters

Code obfuscation is used legitimately to prevent reverse engineering of proprietary applications. However, it can be also used by attackers to hide malicious content. Filters are used in PDFs to compress data for encoding and reduce file size and are frequently used by attackers to conceal malicious content. Available filters and their primary purposes are discussed by Baccas and Kittilsen [6], [7].

Embedded Files

A PDF file can contain other file types, such as HTML, JavaScript, SWF, XLSX, EXE, or even another PDF file, which can be used to embed malicious files that are frequently obfuscated. When special techniques are applied, the embedded file can be opened without alerting the user. Recently, Maiorca et al. [3] presented a novel evasion technique called "reverse mimicry," which was designed to evade state-of-the-art malicious PDF detectors based on their logical structure[10] [4]. Mimicry attacks inject malicious content into a benign PDF while maintaining its benign structure. This method can be automated

[7] http://www.israeldefense.co.il/?CategoryID=512&ArticleID=5766.
http://www.ynet.co.il/articles/0,7340,L-4481380,00.html.
http://www.israelhayom.co.il/article/152741

[8] http://www.seculert.com/

[9] **Heap Spraying** - A technique used in exploits to assist random code execution.

[10] PDF logical structure is a hierarchy of structural elements, each represented by a dictionary. See the PDF file structure section.

easily and does not require knowledge of the structural features used in the maliciousness detector.

Form submission and URI attacks

Hamon [8] presented practical techniques that can be used by attackers to execute malicious code from a PDF file. The author showed that security mechanisms, such as the Protected Mode of Adobe Reader X or the URL Security Zone Manager of Internet Explorer, can be easily disabled by changing the corresponding registry key. Moreover, a URI[11] address can be used (instead of a URL), directing the user to any type of file located remotely, including executables. It should be noted that Adobe Reader version X, released in 2011, included a new sandbox isolated environment, Protected Mode Adobe Reader (PMAR), that ensures that malicious code operations cannot affect the operating system. Nevertheless, most organizations (including universities) do not always keep up with the newest versions of PDF readers, and thus, are exposed to many of the well-known attacks.

3. Analyzing Vulnerabilities of Popular Scholarly Digital Libraries

Now, we briefly present the most popular libraries, their market share, and their uniqueness, and then explain what vulnerabilities exist within them. In addition, we present new vulnerabilities that we utilized. We first present three libraries in which we found a vulnerability, and then, we briefly present additional scholarly digital libraries that should be further checked for vulnerabilities, and finally, for the reader's convenience, we provide a summary table of the different scholarly digital libraries and the manner in which they work.

3.1. Google Scholar

Google Scholar[12] is a free public Web search engine for scholarly literature. It consists of nearly 100 million scholarly documents and is considered the largest scholarly digital library, encompassing 87% of these documents [9]. It indexes scholarly literature across publishing sources. Current articles are indexed and can be found when searched. A user clicking on an article that appears on the results page of Google Scholar is usually directed to the article’s Web page on the publisher’s official Website. In addition to articles on the publisher’s Website, other versions of the papers, from other places on the Web are also indexed (e.g., papers from a researcher’s Web page on an academic institution’s Website).

In order to demonstrate contamination of a digital library such as Google Scholar, we used the Web page of a researcher at a known university (we do not give details for privacy reasons). The articles on the researcher’s Web page were indexed by Google Scholar previously and can be accessed by clicking the “All X versions” link under the relevant article in Google Scholar, as shown in Figure 1. With no connection to the researchers’ names appearing in Figure 1, after we had obtained another researcher’s

[11] **URI** – "a compact string of characters for identifying an abstract or physical resource," RFC2396. It is an extension of URL used for identifying any Web resource (not limited to Web pages).

[12] https://scholar.google.co.il/

permission, we downloaded the most popular paper (a PDF file) from his Web page and injected a malicious JavaScript into it using a PDF editing program called *PDFFill*[13] (such that the malicious JavaScript code is launched when the new article's file is opened). Then, we replaced the benign paper with this new malicious version of the paper on the researcher's website. Now, the malicious paper is available for downloading through Google Scholar using the original indexing information that was neither changed nor updated toward the replacement of the paper behind the published URL. The vulnerability in Google Scholar lies in the indexing mechanism, which checks only the title and author's name and pays no attention to whether a new file was uploaded with the same title and author's name.

As far as we could determine, Google Scholar does not verify that the uploaded paper is related to the researcher's homepage. Thus, a malicious PDF file that carries the same title and authors of a popular paper can easily be created and placed on other Web pages unconnected to a researcher's home page within a university. These malicious papers can be easily promoted with an acceptable payment to Google for a promoted link. Thereby, the attacker uses several elements to launch his attack. First, he takes advantage of the popularity of a particular paper, second he uses the fact that Google Scholar is a trusted source of information, and third he exploits a vulnerability in the Google Scholar indexing mechanism. Consequently, the attacker achieves his attack goals by redirecting the download traffic to his malicious version.

3.2. CiteseerX

CiteSeerX[14] is a growing scientific literature digital library and search engine that focuses primarily on literature in the areas of computer and information science. It is unique in that it collects papers solely from researchers' homepages from the domains of universities and physically stores the papers themselves, in addition to linking to them. The result is that the library contains over four million academic papers in PDF format, and its total size is estimated at about 3.8 terabyte.

According to the way in which CiteseerX collects academic documents, we identified several methods by which a malicious PDF paper could be indexed by a popular digital library. A malicious paper could be uploaded to a researcher's Website directly. This can happen unintentionally if the paper was infected by a malware resident on the computer before it was placed on the Website. Alternatively, the paper could be contaminated using a free, malicious PDF creator that injects malicious code into the edited papers. Another likely scenario is that the researcher's page could be hacked, with the attacker replacing a benign paper with a malicious one. In each of these examples, a malicious paper finds its way to the researcher's homepage within an academic institution's trusted domain, making it available for uploading by CiteseerX as well as to the general public worldwide.

3.3. Social Network Based Scholarly Digital Libraries

Research-Gate[15] (founded in 2008) is a social networking site for scientists and researchers, enabling them to share papers, communicate, and find collaborators. Today,

[13] http://www.pdfill.com/
[14] http://csxstatic.ist.psu.edu/about
[15] http://www.researchgate.net/

it has more than six million members. Research-Gate is also considered an academic digital library as its members can upload and share papers with other members. Academia.edu (launched in September 2008) is a platform for academics for sharing research papers, monitoring their impact, and following researchers in a particular field. A total of 17,896,413 academics have signed up to Academia.edu, adding 5,089,710 papers and 1,450,657 research interests. Academia.edu attracts over 15.7 million unique visitors a month[16]. Research-Gate and Academia.edu are examples of scholarly academic digital libraries affiliated with social networks for researchers whose purpose it is to share data, papers, and knowledge with other researchers.

We created a fictitious profile of a famous researcher through Academia.edu, a process during which we were asked many questions about the researcher and were even asked to upload some of his papers. We uploaded several of his well-known and published papers in order to boost the profile's credibility and gain the trust of colleagues. After several weeks, when the profile was active and papers had been downloaded from the profile, we were able to easily upload a malicious version of the same papers in order to perform an attack. The uploading of an existing malicious PDF file (a non 0-day malicious PDF file) that should have been recognized by an antivirus tool was not stopped by any security mechanisms of the library. Thus, we also show here that social relationships and trust can be sufficient for leveraging a social network-based library for the distribution of a malicious PDF based attack.

3.4. Additional Existing Scholarly Digital Libraries

The following libraries are additional existing scholarly digital libraries that we have not yet checked for vulnerabilities; however, we assume that vulnerabilities exist and should have been further investigated.

Microsoft Academic Search[17] is a free public Web search engine for academic papers and literature, developed by Microsoft Research for the purpose of algorithm research on object-level vertical search, data mining, entity linking, and data visualization. Microsoft Academic Search consists of almost 50 million scholarly documents and is considered one of the top alternatives to Google-Scholar [9].

Web of Science[18] is an online subscription-based scientific citation indexing service maintained by Thomson Reuters that provides comprehensive citation search. It consists of nearly 50 million scholarly documents and is considered, together with MAS, one of the largest academic digital libraries after Google-Scholar [9]. One should note that Web of Science does not index the PDF files, as Google-Scholar does.

PubMed[19] is a free search engine that primarily accesses the MEDLINE database of references and abstracts on life sciences and biomedical topics. The United States National Library of Medicine at the National Institutes of Health maintains the database as part of the Entrez system of information retrieval. PubMed comprises over 24 million citations of biomedical literature from MEDLINE, life science journals, and online books. Citations may include links to full-text content from PubMed Central and publishers' Websites.

arXiv is an automated electronic repository and distribution server for research articles, consisting of electronic preprints of scientific papers in the fields of mathematics,

[16] https://www.academia.edu/about
[17] http://academic.research.microsoft.com/
[18] https://apps.webofknowledge.com/
[19] http://www.ncbi.nlm.nih.gov/pubmed

physics, astronomy, computer science, quantitative biology, statistics, and finance, which can be accessed online. Almost all scientific papers within arXiv are self-archived, meaning that they were uploaded by the users themselves.

Table 1 summarizes the details of the interesting aspects mentioned in this section. The largest libraries, Google-Scholar, MAS, and Web of Science, do not rely on papers uploaded by users as they crawl papers from the publishers as well and do not store them. Note that there are several closed group\ libraries within the Darknet, such as Libgen, Sci-hub and Booksc, and we assume that specifically in these not wide-open libraries the probability and percentage of malicious papers is higher than in the known and wide-open libraries. This assumption should be scrutinized in future research.

Scholarly digital libraries	Upload by User (Preprint)	Crawling Publisher	Crawling Authors Homepage (Preprint)	Indexing the PDF content	Store the PDF	Link to the original PDF	Number of Scholarly documents (In Millions)
Google Scholar	No	Yes	Yes	Yes	No	Yes	99.3
Microsoft Academic	No	Yes	Yes	Yes	No	Yes	50
Web of Science	No	Yes	No	No	No	Yes	50
CiteseerX	Yes	No	Yes	Yes	Yes	Yes	4.2
PubMed	No	Yes	No	Yes	No	Yes	24
Research Gate	Yes	No	No	Yes	Yes	No	Unknown
Academia.edu	Yes	No	No	Yes	Yes	No	5
arXiv	Yes	No	No	Yes	Yes	No	1
http://libgen.org/scimag (Darknet)	Yes	Yes	No	No	Yes	No	36
http://sci-hub.org/ (DarkNet)	No	Yes	No	No	Yes	No	Unknown
http://booksc.org/ (DarkNet)	No	Yes	No	No	Yes	No	18

Table 1. Summary of Scholarly digital libraries' details regarding to their crawling, indexing and redirecting approaches to the scholarly documents.

4. Methods

On this section we present the Dataset scanning tools and technical details that allowed us to provide the results and insights of this study.

4.1. Dataset

As part of this collaborative study with the CiteseerX team, we scanned and analyzed the CiteseerX digital library as our dataset. Our goal was to determine whether this platform had already been used, either intentionally by an attacker or unintentionally by an innocent researcher, to distribute malicious PDF files, and in so doing, to measure the extent of harm that can be caused by such a scenario. When we began scanning, the CiteseerX library contained 4,044,118 academic papers in PDF format that were collected up to the end of 2014, from more than 188 different countries over most of the continents, written by 1.3 million disambiguated authors from 4963 different universities.

4.2. Scanning Tool for Malicious files

We used the VirusTotal[20] service to scan the entire CiteseerX library for malicious PDF files. VirusTotal, a subsidiary of Google, is a free online service that provides comprehensive analysis of files and Websites (URLs) by a set of ~57 antivirus engines and Website scanners. VirusTotal allows a user to submit suspicious files for analysis. After the analysis, VirusTotal provides a report that specifies the identification of suspicious files for each of the antivirus engines. When a file is about to be scanned, VirusTotal calculates its hash to determine whether if it was previously scanned. If so, the stored report is provided to the user; otherwise, the file is then uploaded and scanned, and a report is generated when the process is complete. VirusTotal also provides a rich and public API[21] for the submission of files and URL addresses and retrieval of the analysis reports. The public API can be used through several programming languages that assist with automating the submission and report retrieval procedures. Note that we considered a PDF file as a malicious, only if at least 5 different anti-viruses detect it as a malicious file. In addition we emphasize that rather than presenting a novel technique of malicious PDF files detection, the goal of this study is revealing a simple yet very dangerous way by which the scholarly digital libraries can be utilized as a platform for malware distribution.

4.3. Scanning Technical Details

Since only a small percentage of CiteseerX's papers had been previously scanned, we uploaded all of its content, file by file, to VirusTotal, in order to scan the whole library. Three interrelated problems with this approach were encountered: 1) the enormous size of the library; 2) the length of time it would take to upload the entire library to VirusTotal; and 3) VirusTotal’s submission limit for regular users of four per minute. A quick calculation showed that the scanning process would take approximately 700 days. To cope with the issue of data size, we took a different approach and used VirusTotal’s option to analyze URL addresses of files (URI[22]). When a URI address is submitted to VirusTotal for analysis, it downloads the file that stands behind the address and analyzes it too. However, there is no guarantee that this operation is actually done. The submission of the URI addresses of CiteseerX’s articles to VirusTotal for analysis (versus submitting the actual PDF files) facilitated the process and shortened the time it took to upload the entire library (~3.8 TB). To circumvent the limitation of four files per minute, we requested special privileges (a private API key) from VirusTotal that allowed us to perform many more submissions per minute. We used VirusTotal’s private API to scan the entire CiteseerX library consisting of 4,044,118 academic papers in PDF format. Initially, we submitted the URI addresses of the PDF files iteratively for analysis. Then, we submitted a request for the analysis reports. The scan of the CiteseerX scholarly digital library was completed in five months.

[20] https://www.virustotal.com/
[21] https://www.virustotal.com/en/documentation/public-api/
[22] http://searchsoa.techtarget.com/definition/URI

5. Scanning results

On this section we present the results and their analysis regarding to the scanning process of the PDF files within CiteseerX library. We provide analysis both in the aspects of crawling and downloading the malicious papers, on the basis of worldwide breakdown.

5.1. Crawled Malicious Papers

Of the 4,044,118 URI addresses of PDF files that were submitted for analysis from the CiteseerX library, only 2,586,820 were actually scanned (the process that was previously described). Of these files, 753 (~0.3%) were found and classified as malicious by VirusTotal's antivirus engines. Figure 3 present the breakdown of the threats identified.

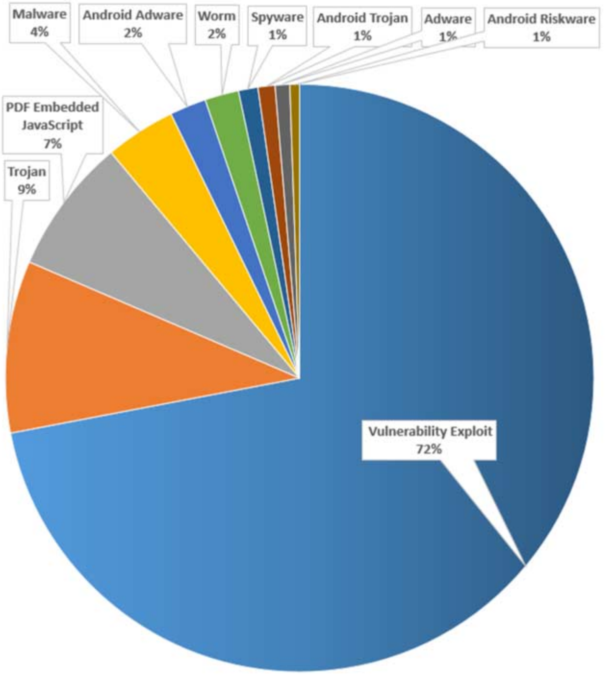


Figure 3. Breakdown of the threats identified among the 753 malicious PDF files found by VirusTotal on the CiteseerX library.

The threats' categories were provided by the identifying antivirus engine. As can be seen in Figure 3, 72% of the malicious files were identified based on vulnerability exploitation[23]. Usually, vulnerability in the PDF file format is exploited utilizing an embedded JavaScript code[24]. 9.5% were classified as a Trojan, a malicious program that when executed performs covert actions that have not been permitted by the user. 7.5% of the malicious files contained JavaScript code that was recognized as malicious.

[23] http://searchsecurity.techtarget.com/definition/exploit

[24] http://blogs.technet.com/b/mmpc/archive/2013/04/29/the-rise-in-the-exploitation-of-old-pdf-vulnerabilities.aspx

JavaScript code can be identified as malicious although it does not exploit any vulnerability and is considered malicious when the code signature is known to represent a malicious code. 3.9% of the malicious files were classified as malware, which means that malicious software (e.g., Exe, PDF, etc.) was found embedded in them. 3.4% of the malicious files contained a threat (Adware[25], Trojan, or Riskware[26]) targeting the Android operating system widely used on mobile devices. 1.9% of the malicious files contained a computer worm[27], which is a malicious program that can propagate by autonomously copying itself from one machine to another. A small percentage of files (1.1%) were classified as Spyware[28], which is a malicious computer program aimed at collecting personal information from the victim's computer. Although it does not damage the victim's computer, it can cause damage to the victim by stealing sensitive information. An Adware is a program that aims to support advertising and operates without the user's permission. An additional 5,775 files were identified as malicious by the Fortinet antivirus, because they contained a suspicious threat called "HTML/Redirector.BK!tr". These files might be malicious since they may direct the user to malicious destinations such as Websites, IP-addresses, and servers. A deeper analysis is required to reach a final decision; however, when the percentage of malicious PDF files in the CiteseerX library rises from 0.3% to 2%, this strengthens even more the phenomenon we are presenting in this paper.

Figure 4 presents the distribution of the malicious scholarly documents according to the geographical location from which they were crawled by CiteseerX scholarly digital libraries. More than 55% of the malicious papers in CiteSeerX were crawled from IP's belonging to USA universities, whereas about 33% were crawled from IP's belonging to European universities.

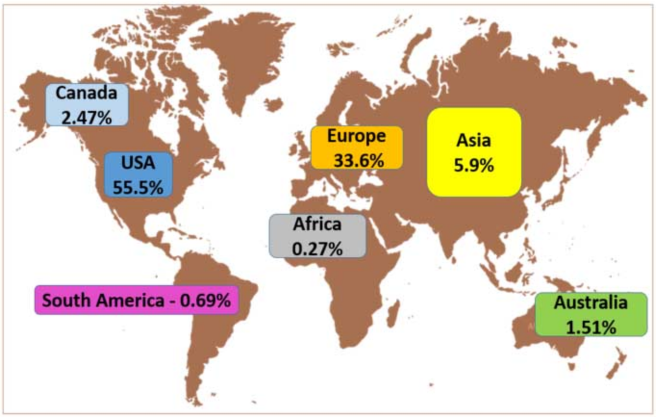


Figure 4. Distribution of the malicious scholarly documents according to the geographical location from which they were crawled by CiteseerX scholarly digital library.

In Table 2, we can see the top 11 European countries in terms of the percentage of malicious scholarly documents crawled from their IP's. Germany was the origin of 10.7% of the malicious papers in CiteSeerX out of the total world's malicious papers,

[25] http://www.pctools.com/security-news/what-is-adware-and-spyware/
[26] http://usa.kaspersky.com/internet-security-center/threats/riskware
[27] http://www.pctools.com/security-news/what-is-a-computer-worm/
[28] http://www.microsoft.com/security/pc-security/spyware-whatis.aspx

and is the origin of more than 31% of the malicious papers in CiteSeerX out of Europe's malicious papers share. It was followed by United Kingdom (6.04%), Holland (2.74%), and France (2.61%). Each of the other European countries not presented here were the origin of less than 0.41% of malicious scholarly documents out of the total world's malicious papers in CiteSeerX.

Country	Percentage
Germany	10.70%
United Kingdom	6.04%
Holland	2.74%
France	2.61%
Austria	2.33%
Luxembourg	2.06%
Sweden	0.82%
Switzerland	0.82%
Denmark	0.69%
Italy	0.55%
Turkey	0.55%

Table 2. Breakdown of the distribution of the malicious scholarly documents according to universities in countries within Europe from which they were crawled by CiteseerX out of total world's malicious papers.

Asia includes several countries, e.g., China, Russia and Korea, which on the one hand are known to have a large population of researchers and on the other were found to be the origin of many malwares. We were surprised to find that only 5.9% of the malicious papers were crawled from IP's belonging to an Asian institution.

In Table 3, we can see some interesting statics regarding Asian countries. Israel is quite a small country with a population constituting 0.1% of the world's population and naturally thus has also a small research community as compared to other countries in the world. Nevertheless, Israel is the origin of 1.51% of the malicious papers in CiteSeerX out the total world's malicious papers, whereas China, that has 20% of the worlds' population is the origin of only 0.55% of the malicious papers in CiteSeerX out the total world's malicious papers. Note that we did not find any malicious paper crawled from Russia or Korea which are the origin of many malicious Android applications found in applications' markets [31].

Country	Percentage
Israel	1.51%
Japan	1.23%
India	0.69%
Republic of Korea	0.69%
China	0.55%
Singapore	0.55%
Taiwan	0.27%
Hong Kong	0.14%
Iran	0.14%
Pakistan	0.14%

Table 3. Breakdown distribution of the malicious scholarly documents according to universities in countries within Asia from which they were crawled by CiteseerX scholarly digital libraries out of the total world's malicious papers.

5.2. Downloaded Malicious Papers

We now present the impact and power attack of malicious papers published in a scholarly digital library. Using CiteeSeerX's database and its Website historic log files, we extracted and aggregated the information regarding the download data of the malicious papers we found. We faced a big-data scale problem due to the enormous amount of data we needed to extract and process, and therefore, we extracted the downloading information for only the top 31 malicious papers identified by a larger number of antivirus engines out of the total 723 that were found. We also focused on download statistics for the five preceding years and therefore we can provide conclusions regarding updated download trends. In addition, we also used GNU Parallel[29] to boost the speed and reduce the very long running time. These data comprised 5197 successful downloads of malicious papers (during 2009-2014) that resulted from only 31 malicious papers crawled by CiteeSeerX's, meaning that scholarly digital libraries have an average 'damage coefficient' of 167 in the last 5 years. The average number of different countries that downloaded malicious papers was 16 over most of the continents (apart from Antarctica), which constitutes a very wide coverage of the worlds' research population within universities and other institutions. Table 4 presents information regarding the top 20 most downloaded malicious papers during the last 5 years. The most downloaded malicious paper is on the topic of Computer Forensics and apparently was a malicious version of a very popular paper; it was downloaded 2213 times in 108 different countries on all continents (apart from Antarctica). The popular topics among malicious papers were related to computers, such as cyber security and computer sciences.

Paper's Topic	Origin Country	Total Number of Downloads	Number of Countries
Computer Forensics	USA	2213	108
Network Security	France	860	77
Computer Hardware	Germany	633	59
Computer Networks	Germany	480	52
Social Networks	USA	235	51
Learning	Germany	123	20
Mathematics	Germany	90	12
Computer Science	USA	79	11
Software Engineering	BVI	77	4
Mathematics	USA	57	7
Computer Science	USA	57	4
Sociology	Brazil	46	10
Economics	USA	42	4
Computer Science	China	35	7
Computer Science	France	23	7
Computer Science	Holland	20	3
Astronomy	USA	20	3
Medical	USA	17	5
Physics	USA	13	4
Meteorology	Canada	13	3

Table 4. Top 20 most downloaded malicious scholarly documents during the last 5 years, their origin country, and the number of countries in which they were downloaded.

[29] http://www.gnu.org/software/parallel/

Figure 5 presents the distribution of the malicious scholarly documents according to the geographical location from which they were downloaded from CiteseerX scholarly digital libraries. More than 41% of the malicious papers in CiteSeerX were downloaded from IP's belonging to USA, whereas about 28% were downloaded from IP's belonging to Asia.

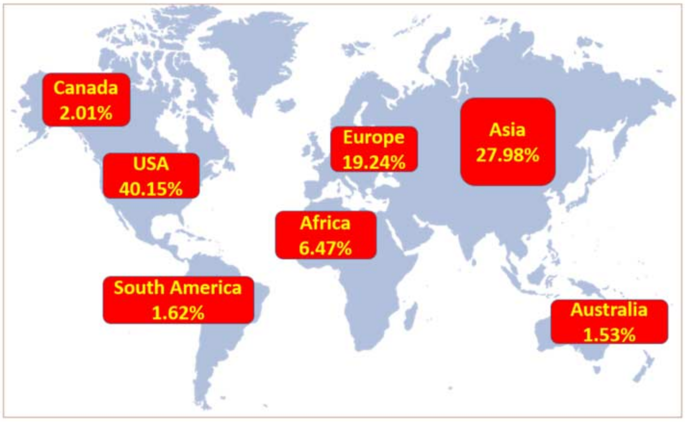


Figure 5. Distribution of the malicious scholarly documents according to the geographical location from which they were downloaded from CiteseerX scholarly digital library.

Figure 4 shows that the USA was the origin of more than 55% of the malicious papers in CiteseerX, while Table 5 shows that the USA was also the most popular destination, where more than 40% of the malicious papers were downloaded, followed by India (9.52%), China (5.04%), and the UK (3.77%). As can be seen, using a scholarly digital library as a platform, an attacker can easily distribute a worldwide attack through a malicious scholarly document.

Country	Percent of Downloads
United States	40.15%
India	9.52%
China	5.04%
United Kingdom	3.77%
Germany	2.87%
Philippines	2.77%
Spain	2.16%
Canada	2.01%
Iran	1.67%
Malaysia	1.23%
Italy	1.21%
France	1.15%
Australia	1.14%
Egypt	1.10%
Ethiopia	0.98%
Russia	0.89%
Korea	0.87%

Table 5. Top countries downloaded most of the malicious scholarly documents from CiteseerX during the last five years.

6. Directions for Security Enhancements

Several steps can be taken to mitigate and improve the detection of malicious PDF files within scholarly digital libraries. We will elaborate on several, beginning with the simplest and easiest to apply.

6.1. Compatibility Check of PDF Files

We found that many of the malicious files are not compatible with the PDF file format specifications according to the Adobe PDF Reference[30] and cannot in fact be opened by the PDF reader and viewed by the user. In cases involving malicious PDF files, the malicious operations will be executed anyway. We suggest using these observations to flag files that can initially be blocked from publication by the digital libraries. In order to empirically support our claim, we collected and created a dataset of malicious and benign PDF files. We acquired a total of 50,908 PDF files, including 45,763 malicious and 5,145 benign files, from 4 sources, as presented in Table 6 below. Note that this collection is not related to the CiteseerX collection that we analyzed on the previous sections. The benign files were reported to be virus-free by Kaspersky antivirus software. The malicious PDF files contain several types of malware families, such as viruses, Trojans, and backdoors. We also included obfuscated PDF files.

The analysis of our large dataset of 50,908 files by the parser (PdfFileAnalyzer[31]) shows that most of the malicious files (96.5%) are not compatible with the PDF file format specifications according to the Adobe PDF Reference. When the user tries to open an incompatible file (malicious or benign), the PDF reader is not able to open it and provides an error message. If it is a malicious PDF file, the malicious operation is executed; if it is a benign file, nothing occurs. However, in both cases the file remains unopened and cannot be viewed by an innocent user. Thus, it is clear that there is no reason to deliver an incompatible file to the user, and this observation should be taken into account in academic digital libraries, which can easily identify such files and mark them as suspicious, or even block them from being published before they are ever opened by an innocent user.

The incompatibility observed was located at the end of the file between the "startxref" and "%%EOF" lines. This line should contain a number serving as a reference (offset) to where the last cross reference table section is located in the file. In cases of incompatibility, the number that appears is incorrect. Table 6 includes the number of compatible files (bracketed) in each of our collected datasets. Note that while incompatible benign files were not present in our dataset, this does not necessarily mean that there were no such files. It might, however, suggest the very low probability of incompatibility among benign files and it provides support of our observation mentioned above.

[30]http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/pdf_reference_1-7.pdf
[31] http://www.codeproject.com/Articles/450254/PDF-File-Analyzer-With-Csharp-Parsing-Classes-Vers

Dataset Source	Year	Malicious Files	Benign Files
VirusTotal Repository	2012-2014	17,596 (1,017)	-
Srndic and Laskov [4]	2012	27,757 (437)	-
Contagio Project	2010	410 (175)	-
Internet and BGU Univ. (random selection)	2013-2014	0	5,145
Total		**45,763 (1,629)**	**5,145**

Table 6. Our collected dataset categorized as malicious, benign and the rate of incompatibles PDF files among the categories.

6.2. Update Check of Re-uploaded PDF Files

One of the vulnerabilities we found in academic digital libraries (particularly Google Scholar) relies on the fact that once a new paper is initially uploaded and indexed, it is then assumed to be scanned to verify that it is virus-free. However, in cases in which the clean paper file behind the indexed link was later replaced by a malicious version, the file was not rescanned and is now the paper's version available as a malicious file through these libraries. We suggest applying a simple check of the hash function behind each indexed file after it is first uploaded. The original hash function is compared to a daily hash function of each indexed file; thus a mismatch between the daily hash of the file and the original version acquired on the initial upload serves as an indication that the file should be further scanned to verify that it is virus free.

6.3. New PDF Malware Backward Check

While the vulnerabilities of new PDF files are identified from time to time by virus experts, the duration of the discovery period might be quite long. The new vulnerability is meanwhile being used and distributed in additional PDF files. Considering a 0-day malware contains such new vulnerabilities, it will probably evade the widely used antivirus tools. Therefore, as new vulnerabilities are discovered and antivirus tools are updated accordingly, we suggest a periodic re-check to provide a comprehensive review of a process that could easily be automated for all the files in the scholarly digital library.

6.4. Machine Learning Algorithms for Unknown Malware Detection

To date, antivirus packages are not sufficiently effective at intercepting malicious PDF files, even in the case of highly prominent PDF threats (Tzermias et al. [13]). On the other hand, according to studies such as [13], [4], [5], [14], [15], [16], [17], [18], [19], [20], machine learning (ML) methods can effectively distinguish between malicious and benign PDF files.

In this section, we explain comprehensively and in depth which and how existing high performance detection methods based on machine learning should be applied by scholarly digital libraries. These solutions should be applied offline after a new scholarly PDF document is found and before it is published and indexed. We propose using a machine learning-based detection model that includes a hybrid detection approach that conducts both static and dynamic analysis, as suggested in [13], [3], [15], and [19]. Thus, the chance of an attack evading the detection mechanism is significantly reduced, because most attacks can be determined by dynamic analysis. Still, several techniques may evade detection, including those that perform the malicious actions of the PDF file only when specific conditions are met (e.g., time, date, IP, and specific user intervention). In these instances, dynamic analysis will be ineffective since it will not encounter and detect the malicious behavior through its analysis. Static analysis scrutinizes the file's genes, content, and structure, which are usually constant; consequently, static analysis will not be affected by these techniques and will therefore be more effective than dynamic analysis. Because of these advantages of static analysis, we suggest an initial static analysis stage for unknown PDF files. A file that is not identified as malicious after this initial stage and is also not detected by antivirus tools would merit further dynamic analysis.

For the static analysis phase, the key to precise and sensitive detection is preliminary knowledge of the primary attack and evasion techniques that could be used by a PDF file, as described in Section 2-b. The first mission is therefore to find and extract the indicators that assist and support the determination of these attacks. A prerequisite for a comprehensive analysis of a given PDF file is the development of a sophisticated and robust parser that is able to extract all the relevant information from the analyzed file (including corrupted PDF files, embedded EXE, PDF, and SWF files).

According to Vatamanu et al.'s study [14] in which the largest PDF file corpus was used, about 93% of the one million malicious PDF files (out of a corpus of 2.2 million) contained JavaScript, whereas only 5% of the benign PDF files contained JavaScript code. Therefore, as a mitigation strategy for malicious JavaScript code, all the JavaScript code should be extracted using a robust parser (including an unrelated object of JavaScript code as presented in [3]). The JavaScript code should be analyzed using two different direct representations that provide high TPR, the lexical analysis of JavaScript code [5], and tokenization of the embedded Java Script [14]. Direct representation means analyzing the code itself, while indirect representation means analyzing meta-features related to the entire content of the file. The JavaScript will also be dynamically analyzed during the dynamic analysis phase. We also suggest conducting an indirect static analysis, which analyzes the general descriptive content in the PDF file rather than directly analyzing the JavaScript code. This can be achieved by an approach that utilizes the meta-features of the content and structure of the PDF file, such as structural paths [4], summarized meta-features [16], and frequency of keywords [17], which also provided satisfactory results. The advantage of using meta-features such as structural paths [4] is that they are not affected by code obfuscation. It was shown to be a very effective method to discriminate malicious PDFs from benign PDFs, even in malicious files created two months after the classification model was created.

As a solution to embedded malicious files (reverse mimicry attacks [3]), the parser should also indicate whenever this scenario (a file embedded inside the PDF) exists in the suspicious PDF file. Generally speaking, there are few benign reasons to embed a file inside a PDF file. In addition, the parser should recursively extract every embedded file inside the PDF and analyze it using the static analysis methods suggested above. One of the reverse mimicry attacks [3] that embeds malicious EXE files in the PDF and auto-executes it when the PDF file is opened is based on a well-known legitimate feature that has been blocked in Adobe Reader X (version 10). Many organizations, however, do not update their installed software, and thus, are exposed to EXE running (such as in Adobe Reader MS Office). Regardless, when another feature or vulnerability has been found that allows the operation of running EXE files embedded in PDF files, it can be detected with a variety of advanced techniques aimed at the detection of malicious executables using static and dynamic analysis.

All the extracted features mentioned in this section can be leveraged by an ensemble of classifiers such that each classifier will be induced from different sets of features. Menahem et al. [12] showed that applying an ensemble of classifiers using different features can significantly improve detection capabilities.

The attacks that were presented by Hamon et al. [8] dynamically load malicious code from a remote source as well as URI resolving (executing external malicious file). These attacks usually rely on clicking on a link; however, it is possible to open the link when the PDF file is opened, and therefore the PDF file becomes the link. Consequently, as indicated by Hamon et al. [8], the /OpenAction command is considered dangerous and can be detected by simple static analysis. Restricted use of this command will help prevent this kind of attack.

In the dynamic analysis phase, it is more effective to rely on hooking the Adobe Reader or using hardware virtualization to execute the JavaScript code embedded in the PDF file rather than to run it in an emulator, as presented in [19] and [20]. The malicious JavaScript code inside the PDF, however, can recognize that it is being executed in an emulated environment, and therefore, it might refrain from performing its malicious behavior. This will, however, probably provide a solution for malicious obfuscated JavaScript code that was not detected by the static analysis.

As far as we could identify, no product or academic solution actually analyzes the URLs inside the links in a PDF file. A link, having been clicked, can refer the user to a malicious Website that, when loaded, initiates an attack on the user's computer. An attacker can place a malicious link inside a benign file and persuade the user to click it. Dynamic analysis methods will not be able to detect this kind of attack, since user intervention is needed to click on the link. However, static analysis methods can easily extract and analyze the links that may be malicious. Thus, we recommend the addition to the detection model of a module that checks the links inside the PDF file for maliciousness, as this module can integrate many of the academic solutions designed for analyzing links (URLs) or Websites for maliciousness [21][25-30].

Full dynamic analysis of PDF files is a costly approach. For instance, Checkpoint Threat Emulation32 and SourceFire FireAMP[33] execute the entire PDF file in an isolated environment (sandbox) and examine the effect of the behavior and actions on the system during runtime. Nevertheless, this detection approach provides a comprehensive indication of the file's purposes and is robust against many evasion techniques, such as code obfuscation and URI resolving. Therefore, we suggest the integration of a full dynamic analysis module that might detect malicious behavior or determine the intention of PDF files in cases where the static or dynamic analyses (based on analysis of specific components of the PDF file) are unable to provide the comprehensive inspection provided by full dynamic analysis.

We also suggest running each suspicious PDF file through several versions of Adobe Reader (or any PDF reader) in order to compare its behavior. Some malicious PDF files will behave differently depending on the version of Adobe Reader used, because vulnerabilities are treated differently from one version to another. The differing behavior might provide an indication of a file’s maliciousness.

[32] https://threatemulation.checkpoint.com/teb/

[33] http://www.sourcefire.com/security-technologies/advanced-malware-protection/fireamp

Moreover, one should remember that many organizations currently rely on outdated versions of PDF readers due to financial constraints and lack of proper administrative controls. The fact that many organizations do not update their installed software (including their PDF readers) exposes their computers and users to many known exploitations and bugs associated with PDF readers, such as the JBIG2Decode algorithm and util.printf Java function, as was discussed by Stevens [24]. New readers take these exploits into account; however, the exploits and bugs remain relevant in older versions of software. As a rule of thumb, we therefore recommend that organizations strive to equip themselves with the latest version of PDF readers as a standard security policy.
We also suggest applying an active learning framework for enhancing the detection of malicious PDF files that was recently presented by Nissim et al. [2], [36], which addresses an important issue that none of the above mentioned papers considered: the detection model's lack of updatability. It is not adequate to construct and calibrate a preliminary detection model based on sophisticated feature extraction techniques, but rather the model should be constantly updated in light of the daily creation of new malicious PDF files. While machine learning has been successfully used to induce malicious PDF detection models, all methods utilizing this approach focus on passive learning. Alternatively, we suggest focusing on active learning [23] and the use of the active learning methods that have been specially designed to enrich the detection model with new malicious PDF files over the course of several days, thus ensuring that the detection model is up to date. This notion was successfully used to enhance the detection of variety of malware including executable malwares in the Microsoft Windows OS [22], malicious documents of MS office word [34], Android malware [35], and is expected to enhance the detection of malicious PDF files as well.

7. Discussion and Conclusion

This study revealed the phenomenon of the contamination of scholarly digital libraries with malicious PDF documents and showed how these libraries can be easily used for launching and distributing targeted cyber-attacks aimed at a specific group of researchers, universities, institutions, and countries. As far as we know, there are no reliable reports of the accurate percentage of malicious PDF files on the Web, and therefore, we cannot determine whether scholarly digital libraries are more or less contaminated than the Web itself. In addition, as we found these malicious documents on CiteSeerX, we will have to remove these papers from it and also update other scholarly digital libraries regarding these malicious papers in order to prevent the attacks they are carrying from being further distributed. This process of removal should be done through cooperation with the authors of these papers, as the authors might discover the existence of resident malware in their computers that caused the infection of their paper, in the case that their paper was not contaminated intentionally.

In this study, we evaluated more than two million scholarly papers in the CiteSeerX library and found it to be contaminated with a surprisingly large number (0.3%-2%) of malicious PDF files belonging to a variety of different virus families, 72% of which exploit vulnerabilities in PDF readers. These malicious documents were uploaded from 46 different countries covering most of the continents. The USA's universities were found to be the origin of more than 55% of the malicious papers in CiteSeerX. The USA also downloaded more than 41% of these malicious scholarly papers during the last five years. On average, a malicious paper was downloaded 167 times in 5 years by researchers from many different countries worldwide. The most popular malicious scholarly document is a malicious version of a famous paper in the computer forensics domain, crawled from the USA, and downloaded 2213 times in 108 different countries. Therefore, as we indicated, several vulnerabilities exist in scholarly digital libraries, and an attacker needs only to have a malicious version of a popular paper on an attractive topic (e.g., cyber-security) in a scholarly digital library to utilize the high damage coefficient we found and thus cover most of countries in the world. We also suggested several solutions for mitigating such attacks, including simple deterministic solutions and also advanced machine learning-based frameworks that should both be integrated in scholarly digital libraries.

In future work, we suggest that the other digital libraries for which we presented vulnerabilities be scanned further, and also that additional scholarly digital libraries be investigated for vulnerabilities, such as MAS, Web of Science, and Pub-Med. We also suggest investigating the rate of contamination of digital libraries within the Darknet, such as Libgen, Sci-hub, and Booksc, which we presented as well.

References

[1] Nir Nissim, Aviad Cohen, Chanan Glezer, Yuval Elovici, Detection of malicious PDF files and directions for enhancements: A state-of-the art survey, Computers & Security, Volume 48, February 2015, Pages 246-266, ISSN 0167-4048, http://dx.doi.org/10.1016/j.cose.2014.10.014.

[2] Nir Nissim, Aviad Cohen, Robert Moskovitch, Oren Barad, Mattan Edry, Assaf Shabatai and Yuval Elovici, " ALPD: Active Learning framework for Enhancing the Detection of Malicious PDF Files aimed at Organizations.", JISIC (2014)

[3] D. Maiorca, I. Corona and G. Giacinto. Looking at the bag is not enough to find the bomb: An evasion of structural methods for malicious PDF files detection. Presented at Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security. 2013, .

[4] N. Šrndic and P. Laskov. Detection of malicious pdf files based on hierarchical document structure. Presented at Proceedings of the 20th Annual Network & Distributed System Security Symposium. 2013.

[5] P. Laskov and N. Šrndić. Static detection of malicious JavaScript-bearing PDF documents. Presented at Proceedings of the 27th Annual Computer Security Applications Conference. 2011, .

[6] Baccas, Paul. "Finding rules for heuristic detection of malicious pdfs: With analysis of embedded exploit code." Virus Bulletin Conference. 2010.

[7] Kittilsen, Jarle. "Detecting malicious PDF documents." (2011).

[8] Hamon, Valentin. "Malicious URI resolving in PDF documents." Journal of Computer Virology and Hacking Techniques 9.2 (2013): 65-76.

[9] Khabsa, Madian, and C. Lee Giles. "The number of scholarly documents on the public web." PLOS one 9.5 (2014): e93949.

[10] Gargouri, Y., et al., Self-Selected or Mandated, Open Access Increases Citation Impact for Higher Quality Research, PLOS ONE, 5(10): e13636, October 18, 2010 GS Biblio

[11] Lawrence, S., Free online availability substantially increases a paper's impact, Nature, 31 May 2001 GS Biblio

[12] E. Menahem, A. Shabtai, L. Rokach and Y. Elovici. Improving malware detection by applying multi-inducer ensemble. Comput. Stat. Data Anal. 53(4), pp. 1483-1494. 2009.

[13] Z. Tzermias, G. Sykiotakis, M. Polychronakis and E. P. Markatos. Combining static and dynamic analysis for the detection of malicious documents. Presented at Proceedings of the Fourth European Workshop on System Security. 2011.

[14] C. Vatamanu, D. Gavriluţ and R. Benchea. A practical approach on clustering malicious PDF documents. Journal in Computer Virology 8(4), pp. 151-163. 2012.

[15] F. Schmitt, J. Gassen and E. Gerhards-Padilla. PDF scrutinizer: Detecting JavaScript-based attacks in PDF documents. Presented at Privacy, Security and Trust (PST), 2012 Tenth Annual International Conference On. 2012.

[16] C. Smutz and A. Stavrou. Malicious PDF detection using metadata and structural features. Presented at Proceedings of the 28th Annual Computer Security Applications Conference. 2012.

[17] D. Maiorca, G. Giacinto and I. Corona. "A pattern recognition system for malicious pdf files detection," in Machine Learning and Data Mining in Pattern RecognitionAnonymous 2012.

[18] H. Pareek, P. Eswari, N. S. C. Babu and C. Bangalore. Entropy and n-gram analysis of malicious PDF documents. Int. J. Eng. 2(2), 2013.

[19] X. Lu, J. Zhuge, R. Wang, Y. Cao and Y. Chen. De-obfuscation and detection of malicious PDF files with high accuracy. Presented at System Sciences (HICSS), 2013 46th Hawaii International Conference On. 2013.

[20] K. Z. Snow, S. Krishnan, F. Monrose and N. Provos. SHELLOS: Enabling fast detection and forensic analysis of code injection attacks. Presented at USENIX Security Symposium. 2011.

[21] J. Ma, L. K. Saul, S. Savage and G. M. Voelker. Learning to detect malicious URLs. ACM Transactions on Intelligent Systems and Technology (TIST) 2(3), pp. 30. 2011.

[22] Nir Nissim, Robert Moskovitch, Lior Rokach, Yuval Elovici, Novel Active Learning Methods for Enhanced PC Malware Detection in Windows OS, Expert Systems with Applications, Available online 19 March 2014, ISSN 0957-4174, http://dx.doi.org/10.1016/j.eswa.2014.02.053.

[23] Settles, Burr. "Active learning literature survey." University of Wisconsin, Madison 52 (2010): 55-66.

[24] Stevens, D., "Malicious PDF Documents Explained," Security & Privacy, IEEE, vol.9, no.1, pp.80,82, Jan.-Feb. 2011. doi: 10.1109/MSP.2011.14

[25] Xuewen, Zhu, Wan Xinochuan, and Ye Hua. "Detection of malicious URLs in a web page." U.S. Patent No. 8,505,094. 6 Aug. 2013.

[26] B. Eshete. Effective analysis, characterization, and detection of malicious web pages. Presented at Proceedings of the 22nd International Conference on World Wide Web Companion. 2013.

[27] H. Zhou, J. Sun and H. Chen. Malicious websites detection and search engine protection. Journal of Advances in Computer Network 1(3), 2013.

[28] K. Su, K. Wu, H. Lee and T. Wei. Suspicious URL filtering based on logistic regression with multi-view analysis. Presented at Information Security (Asia JCIS), 2013 Eighth Asia Joint Conference On. 2013.

[29] S. Chitra, K. Jayanthan, S. Preetha and R. U. Shankar. Predicate based algorithm for malicious web page detection using genetic fuzzy systems and support vector machine. International Journal of Computer Applications 40(10), pp. 13-19. 2012.

[30] D. Ranganayakulu and C. Chellappan. Detecting malicious URLs in E-mail–An implementation. AASRI Procedia 4pp. 125-131. 2013.

[31] Axelle Apvrille, Tim Strazzere: Reducing the window of opportunity for Android malware Gotta catch 'em all. Journal in Computer Virology 8(1-2): 61-71 (2012).

[32] Nissim, N., Boland, M. R., Moskovitch, R., Tatonetti, N. P., Elovici, Y., Shahar, Y., & Hripcsak, G. (2015). An Active Learning Framework for Efficient Condition Severity Classification. In Artificial Intelligence in Medicine (pp. 13-24). Springer International Publishing. (AIME-15).

[33] Nir Nissim, Mary Regina Boland, Nicholas P. Tatonetti, Yuval Elovici, George Hripcsak, Yuval Shahar, Robert Moskovitch, Improving condition severity classification with an efficient active learning based framework, Journal of Biomedical Informatics, Volume 61, June 2016, Pages 44-54, ISSN 1532-0464.

[34] Nir Nissim, Aviad Cohen, Yuval Elovici. ALDOCX: Detection of Unknown Malicious Microsoft Office Documents using Designated Active Learning Methods Based on New Structural Feature Extraction Methodology. IEEE Transactions on Information Forensics and Security, 15.1 (2017): 40-55.

[35] Nissim, N., Moskovitch, R., BarAd, O., Rokach, L., & Elovici, Y. (2016). ALDROID: efficient update of Android anti-virus software using designated active learning methods. Knowledge and Information Systems (2016), 1-39.

[36] Nissim, N., Cohen, A., Moskovitch, R., Shabtai, A., Edri, M., BarAd, O., & Elovici, Y. (2016). Keeping pace with the creation of new malicious PDF files using an active-learning based detection framework. *Security Informatics*,*5*(1), 1-20.

A Systems Approach to Cyber Security
A. Roychoudhury and Y. Liu (Eds.)

doi:10.3233/978-1-61499-744-3-129

Low-Dimensional Bigram Analysis for Mobile Data Fragment Classification

Chien Eao Lee [1], Lilei Zheng, Ying Zhang and Vrizlynn L. L. Thing
Cyber Security & Intelligence Department,
Institute for Infocomm Research, Singapore

Abstract. File carving is the process which aims to recover files from storage media without the file system meta-data. The ability to perform such recovery is particularly important in this digital era when it involves forensic investigation. Due to the inevitable occurrence of file fragmentation in storage system, fragment classification is an important step in the file recovery process. Following the increase of storage capacity and usage of mobile phones, large amount of personal data tends to be stored on such devices, which is of great interest for forensic analysis during investigations. In this paper, we present an approach in classifying the most commonly found fragment types on mobile phones, which include JPG, MP3, MP4, MOV and SQLite. Departing from the conventional approaches that utilize analysis derived from unigram statistics, we employ bigram statistics in our approach in order to capture the frequency of local byte order which retains meaningful and exploitable pattern in the fragments. While being able to capture more information, the bigram statistics also contain a large amount of redundant data which greatly increases the computational workload. Therefore, we perform dimensionality reduction through Principal Component Analysis (PCA) in order to extract only the most significant dimensions for classification purpose of the targeted file types. Using the resulting features, an average classification accuracy of 96.19% is achieved, comparing to 88.40% while using the unigram statistics alone through Support Vector Machine (SVM).

Keywords. Fragment classification, mobile data, bigram, principal component analysis, support vector machine

1. Introduction

The continuous advancement of electronics components and its increasing affordability has seen the trend of migration from conventional hard disk drives (HDD) to solid states drives (SSD) in recent years. The many attributes of SSD, including its high speed, small physical size and non-mechanical nature make it extremely suitable for mobile phone implementation. Besides, the expanding storage capacity to support applications and user data has tremendously improved the ability and efficiency of mobile phone in performing many tasks.

A recent survey conducted in [22] indicates that the common usage of mobile phones include activities such as emailing, web surfing, photo taking, social networking, direc-

[1]Corresponding Author: Chien Eao Lee, Cyber Security & Intelligence Department, Institute for Infocomm Research, Singapore; E-mail: leece@i2r.a-star.edu.sg

tion mapping, audio and video recording and downloading. Considering this, some of the most commonly found file types on mobile phones would consequently include images (JPG), audio and video clips (MP3, MP4, MOV), as well as the application database (SQLite). As a highly utilized device in daily life, mobile phone captures a broad range of user data. Therefore, the ability to recover lost, hidden, deleted or corrupted data from mobile phones has become important, especially for the purpose of forensic analysis to obtain compelling digital evidence.

Data carving is the process to extract data from unallocated file system, i.e., when data cannot be identified or recovered due to the absence of meta-data [19]. Common recovery methods involve searching for the headers and footers of the files and then merging all the blocks in between, based on the assumption that the fragments of the files are stored contiguously in memory. File types are then determined by the magic number found in the header fragments. However, studies show that fragmentation often occurs on mobile storage system due to the wear-leveling algorithm in flash-based storage [11], where file fragment has to be identified based on its byte content.

For more than a decade, one of the focuses in file carving area is fragment classification, which serves as an important preliminary step for subsequent recovery processes. Fragment classification is carried out in order to identify the file type a fragment belongs to. Approaches in tackling fragment classification task can be categorized into signature-based, statistical, machine learning, context-based and other approaches such as gray-scale visualization method [20]. In most of these approaches, the statistical characteristics of the bytes in a fragment are utilized to establish its type by performing comparisons across the characteristics models developed for known file types. Many of these characteristics are derived from the unigram statistics of the fragments, where each byte is considered as an independent entity and the order of the bytes is ignored. In order to capture the byte order, N-gram analysis has been used [3,5,8,10,15]. However, the increment of N causes the feature dimension to increase exponentially, where most of them are redundant. Therefore in these works, the computational workload of N-gram statistics are often expensive.

In this paper, we present a low-dimensional bigram approach to classify fragment types that are commonly found on mobile phones, namely JPG, MP3, MP4, MOV and SQLite. We apply Principal Component Analysis (PCA) to the high-dimensional bigram statistics to obtain reduced-dimension bigram features that are significant for classification. Classification is then performed through Support Vector Machine (SVM). Using the classification accuracy through unigram statistics alone as performance baseline, we demonstrate that our approach is highly accurate.

In the next section, Section 2, a brief overview of related work is provided. Section 3 describes our method and Section 4 details our experimental setup. We present our evaluation results and observations in Section 5 and conclude in Section 6.

2. Related Work

With the aim to overcome the limitations of traditional file carving tools which use file extension and magic numbers for file type identification, McDaniel and Heydari [17] proposed the usage of "fileprints", which are generated for different file types using Byte Frequency Analysis (BFA), Byte Frequency Cross-correlation (BFC) and File

Header/Tralier (FHT). BFA algorithm uses the frequency of occurrence of each byte values (0 to 255) in the file to establish the characteristics of the file type. BFC and FHT are used to strengthen the file type identification method, where BFC utilize the relationship between the frequencies of the byte values and FHT perform identification based on the byte patterns that can be found at the beginning and end of a file. Based on the 30 types of "fileprints" generated, the authors performed experiments on 120 complete test files. The reported classification accuracies were 27.50% for BFA, 45.83% for BFC and 95.83% for FHT algorithms.

Noticing the difficulties to establish one single model that is able to accurately represent all files of the same type, Li et al. [15] proposed centroid models, which are generated using Byte Frequency Distribution (BFD), i.e., unigram analysis as the signature. K-means clustering was applied to develop each model for 8 different file types. In order to evaluate the performance of this approach, the authors performed classification using the first 20, 200, 500, 1000 bytes of the file and the complete file. Experimental results indicated that while the classification accuracy were at least 98.3% for the 20-bytes fragments, the accuracy for the classification of the entire file dropped considerably. Besides, the classification for fragments that exclude the header information at the beginning of the files was not evaluated. However, this evaluation is important because fragments could be separated from header due to file fragmentation.

Identifying the limitations in the previous works [15,17], which exploited the header data in the classification process, Karresand and Shahmehri [13] proposed the centroids of mean and standard deviation of the BFD, which uses only the data fragment content as a method of classification. Data fragments are classified based on the comparison of distance and threshold from the centroids. In [12], the authors extended their method by introducing centroid construction based on the distribution of the Rate-of-Change (RoC) metric. RoC is measurement of the difference between two consecutive bytes in terms of ASCII values. From their test results, the RoC method performed exceptionally well in identifying JPG fragments, obtaining an accuracy of 99.2%. Such result was anticipated due to the exploitable pattern of the 0xFF00 stuffing byte in JPG fragments. For other fragment types that did not have such specific patterns, the RoC method showed no convinced performance gain.

Veenman [24] proposed the use of cluster content features in their classification approach, which included the combination of byte values histogram, Shannon entropy and the Kolmogorov complexity measurement. A dataset of 450MB which consists of 11 file types was collected through the internet for experiments. Using a training set of 35,000 clusters and a testing set of 70,000 cluster, the classification approach achieved an accuracy of 99% for html and 98% for JPG. Classification results for the remaining types ranged from 18% to 78%, with the overall average accuracy of 45%. Based on very similar ideas to Veenmans [24], Calhoun and Coles [4] employed additional statistics and the combination of the statistics. Their experiments achieved an average accuracy of 88.3% by employing Fisher linear discriminant.

Axelsson [1,2] employed the k-Nearest-Neighbors (kNN) classification algorithm, using the calculation of Normalized Compression Distance (NCD) between the blocks of known and unknown type. Evaluation on the method was performed using the research data made publicly available by Garfinkel [9]. Using this approach, the authors achieved an accuracy between 32.86% and 36.43%, with k ranged from 1 to 10.

Using the byte values frequencies or unigram distribution as feature vectors, Li et al. [14] performed fragment classification using SVM targeting high entropy files. Four file types obtained from private data set were considered in the multi-class classification using fragments of 4,096 bytes. Evaluations were performed using four different kernels, namely linear kernel, polynomial kernel, Radial Basis Function (RBF) kernel and sigmoid kernel. Their classification achieved an average accuracy of 81.5% using the linear kernel.

Gopal et al. [10] focused on file type identification based on several scenarios of file corruption, such as file fragmentation. Using data set provided by Garfinkel [9], the classification results using SVM and *k*NN were compared against those obtained through commercial tools such as Libmagic, TrID, Outside-In and DROID. Experimenting using *N*-gram features, with *N* varying from 1 to 3, the authors showed that *k*NN and SVM approaches produced better results comparing to the commercially available tools. Also observed in their experiments was that unigram and bigram produced better results for *k*NN and SVM respectively. Apart from the indication that fragment classification is performed better when bigram of fragments were used for training rather than bigram of the full file, no further fragment classification performance was studied as the focus of the authors were on file classification.

Fitzgerald [8] explored the Natural Language Processing (NLP) methods for fragment classification, using unigram and full bigram as well as other statistics like entropy as the features through SVM. The authors also evaluated the methods using the data corpus provided by Garfinkel [9]. For each of the 24 file types, 4,000 fragments were selected in the evaluation where an average accuracy of 49.1% was achieved. Among the 24 file types, the classifier performed well on the low entropy fragment types. The classification accuracy for high entropy fragment types were rather moderate.

Beebe et al. [3] performed comparative evaluations on the RBF and linear kernel of SVM using a different combination of feature vectors, which include unigram, full bigram and other non-*N*-gram features. A total of 30 file types and 8 data types obtained from the Garfinkel corpus [9] and other sources were used in the evaluations, where an accuracy of 73.4% was obtained using linear kernel with concatenation of unigram and bigram as feature vector.

As opposed to most classification strategies that were performed using statistical measure and machine learning approach, Roussev [21] stated the importance of considering the primitive types and compound data format when classifying data fragments. Also emphasized was the necessity of developing specialized classification approach for different targeted file types, rather than depending on generalized algorithms.

3. Low-Dimensional Bigram Analysis

3.1. Fragment Generation

Modern day mobile phones utilize non-volatile flash-based memories for data storage. The architecture of flash file system consists of logical units, blocks, pages and sectors [7]. Each page is formed by main areas of 512 bytes each for data storage and the corresponding spare areas of 16 bytes for memory management process, such as Error Correction Code (ECC), wear-leveling, and other software overhead functions [18]. Each

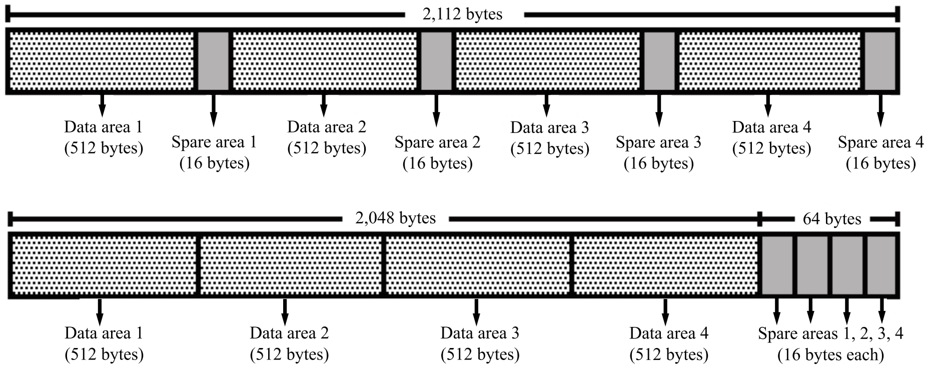


Figure 1. Data area and spare area layout in a page.

```
0C B3 9E 67 0C 1F 53 F7 23 1F 24 67 0C E5 1E E0
```

Figure 2. 16-byte sequence in a fragment.

of the page has possible layouts as shown in Figure 1, where the spare area is either adjacent to the corresponding data area or located separately from the data area at the end of the page [18].

As a content-based classification approach, we ignore the system data in the spare area and perform evaluation based on the data content in the 512-byte fragments.

3.2. Bigram Statistics

Given a sequence of elements, N-gram is the consecutive N elements in the sequence. Utilizing the corresponding N-gram distribution, N-gram model can be understood as a probabilistic model that predicts the occurrence of an item in a sequence based on the elements that precede it [5]. In a sequence S, prediction of the i^{th} item s_i is based on $s_{i-(N-1)}, \cdots, s_{i-1}$. N-gram model is most knowingly applied in statistical natural language processing, and has since been extended to applications in many other areas [5].

In fragment classification, we consider each byte in the fragment as an element in the sequence. Common characteristics used for fragment classification, such as byte frequency statistics, entropy and centroid-based models are often derived from unigram statistics ($N = 1$), where each element in the sequence is treated as an independent item.

Figure 2 shows an example of a 16-byte sub-sequence that can be obtained in a fragment. Unigram statistics can be represented by a feature vector consists of 256 dimensions (byte 0x00 until 0xFF) where each dimension records the frequency of occurrence of its corresponding byte. For example, unigram statistics of the 16-byte sub-sequence in Figure 2 indicates frequency of 3 occurrences for byte 0x0C, 2 occurrences for byte 0x67 and 0x1F, 1 occurrence for the remaining 9 bytes, and 0 occurrence for other bytes that are not shown in the sub-sequence. While the frequency of each 256 possible bytes is captured through unigram analysis, one important information that is omitted is the data structure of contextual bytes. Application of N-gram where $N > 1$ enables us to ob-

$\boxed{\text{0C B3}}$ 9E 67 0C 1F 53 F7 23 1F 24 67 0C E5 1E E0
0C $\boxed{\text{B3 9E}}$ 67 0C 1F 53 F7 23 1F 24 67 0C E5 1E E0
0C B3 $\boxed{\text{9E 67}}$ 0C 1F 53 F7 23 1F 24 67 0C E5 1E E0
0C B3 9E $\boxed{\text{67 0C}}$ 1F 53 F7 23 1F 24 67 0C E5 1E E0
...
...
0C B3 9E 67 0C 1F 53 F7 23 1F 24 $\boxed{\text{67 0C}}$ E5 1E E0
...
...
0C B3 9E 67 0C 1F 53 F7 23 1F 24 67 0C $\boxed{\text{E5 1E}}$ E0
0C B3 9E 67 0C 1F 53 F7 23 1F 24 67 0C E5 $\boxed{\text{1E E0}}$

Figure 3. Bigram analysis of the 16-byte sequence in a fragment.

tain the information regarding the local byte patterns up to N bytes. N-gram distribution can be considered as a sliding window process where we slide a window size of N bytes across the byte sequence of the fragment in the steps of one byte.

Figure 3 shows the bigram ($N = 2$) analysis of the 16-byte sub-sequence where the order of 2 consecutive bytes is considered. Through the 15 bigram features obtained, we are able to capture the byte order existing in the sequence, such as the frequency of the byte 0x670C is 2, as shown in the bold rectangle in Figure 3.

The ability to retain the byte order information is important when there is contextual byte pattern in a fragment type. However, the number of N needs to be optimized as the inclusion of extra bytes in a gram entity might result in the meaningful shorter byte patterns be converted into un-exploitable information. Moreover, increasing N would exponentially increase the feature dimensions, which would become intractable and costing unnecessary high computational overhead. For example, trigram ($N = 3$) has more than 16 million of dimensions (i.e., 256^3), which becomes generally impractical to process. Considering these, we focus our approach using bigram statistics.

For bigram analysis, the resulting dimensions is 65,536 calculated as 256^2. Using the fragment size of 512 bytes, we obtain 511 bigram counts from each fragment. In our research, we intend to capture the principal components of the bigram statistics while keeping the computational workload efficient. In order to do that, we explore the feasibility of utilizing low-dimensional bigram statistics for fragment classification targeting common file types on mobile phones. The restriction on the file types enables us to examine the effect of our approach for data carving through a more specialized and practical way in mobile data environment.

3.3. Low-Dimensional Bigram Using PCA

The high-dimensional features that we obtain through the bigram analysis are meant to capture meaningful underlying byte pattern. Nevertheless, a substantial part of the sparsely represented features does not carry any exploitable pattern that can be usefully interpreted. These extra unnecessary features exist as redundancy.

PCA is an unsupervised linear transformation method adopted frequently in dimensionality reduction procedures. PCA performs dimensionality reduction through orthogonal transformation based on the theory that a set of data can usually be represented using its corresponding principal components through a lower number of dimensions. PCA is a well-known method for finding patterns in high-dimensional data [23]. Using PCA, the original data set is transformed and projected on a new coordinate system based on the order of the greatest variance.

Provided with an original data set of $X = \{x_m \in R^d | m = 1, \ldots, M\}$ where M is the number of samples and d is the number of features or dimensions, we intend to compute a transformation matrix, W that can be applied to obtain the first k ($k < d$) principal components of X. This can be achieved by using the eigenvalues and eigenvectors obtained through Singular Value Decomposition (SVD) of the original data set X,

$$X = UDV^T, \tag{1}$$

U and V are the orthogonal matrices, whereas D is the diagonal matrix where its diagonal elements are the singular values of X. The columns of U are the eigenvectors of XX^T and the columns of V are the eigenvectors of X^TX. The diagonal values of D are the corresponding eigenvalues of XX^T and X^TX. With the eigenvectors and eigenvalues computed, U_k comprises of the first k columns of U, and D_k is the k-by-k matrix obtained starting from the top-left element in D. The transformation matrix, W is computed as

$$W = D_k U_k, \tag{2}$$

where $W \in R^{d \times k}$. For any original vector x, the reduced-dimension vector y is computed as $y = Wx$.

Through this approach, we first compute a transformation matrix, W using the independent model development data set, X_{model}. Then, we obtain the low-dimensional bigram of our training sample, x_t by performing $y_t = Wx_t$, where y_t is the resulting bigram of k dimensions. The dimension k can be adjusted accordingly, which will be discussed in the experiment section of this paper. Similar computation is performed on the testing sample.

3.4. *Classification through SVM*

SVM has been widely employed as a supervised learning algorithm for data classification purpose. The general process involves developing a classification model from a labelled data set, more commonly known as training set. The developed classification model is then used to predict the label of a testing set. The performance of the SVM is determined by the label prediction accuracy. Using the low-dimensional bigram statistics as feature vector, we perform multi-class classification on our targeted fragment types through SVM.

4. Experimental Setup

4.1. *Data Set*

We focus our attention on the classification of 5 fragment types that are commonly found on mobile phones, which include JPG, MP3, MP4, MOV and SQLite. Sample files that form our data set for the first 4 types are collected from the corpus provided by Garfinkel [9], whereas the SQLite data set is obtained from other sources, including several mobile devices. A total of 1,856MB data were collected. From the collected files, we construct data sets of approximately 400MB each for JPG, MP3, MP4 and MOV as well as 256MB for SQLite. For all the collected data, 25% data of each type are used as model development set for PCA and the remaining 50% and 25% are used as training and testing set respectively.

We segment each of the files in the data set into fragments of 512 bytes. The first fragment of each file is ignored as it often contains exploitable file type related header that could bias the results in our intended study. The last fragment of each file is also discarded as it often does not have sufficient 512 bytes of data in length [8].

4.2. PCA Model Development

From the fragments that are generated from the data set assigned for PCA model development, 2,000 fragments are randomly selected from each of the five file types. Through PCA, the bigram statistics of these fragments are used to develop a model to obtain a low-dimensional bigram. Notice that we use a separate data set that is not part of the training and testing data set to generate the PCA model in order to ensure that the developed model is not biased to either of the training set or testing set during the dimension reduction process. Also from a computational point of consideration, the independent model can be applied for all evaluation processes without the need to be re-developed each time the training set is changed.

4.3. Feature Selection and SVM Classification

The feature vectors that are used in our evaluations include the unigram statistics and low-dimensional bigram statistics. We obtain the classification accuracy through unigram statistics to establish a performance comparison baseline [14]. The low-dimensional bigram statistics used in our evaluations are computed through PCA using the independent PCA model. We conduct the classifier training and testing through SVM using the LIBSVM package [6].

5. Results and Discussions

5.1. Unigram vs. Low-Dimensional Bigram

We first perform a comparison of classification accuracy between using unigram and low-dimensional bigram as feature vectors.

Table 1 shows the classification accuracy for 10,000 test fragments (2,000 fragments per type) selected randomly from the test data set. For training the SVM model, we use another 10,000 fragments from the training data set. In this experiment, the bigram statistics of the training and testing data is reduced from 65,536 to 256 dimensions through the PCA model developed in order to perform a direct comparison with the 256-dimension unigram statistics of the data, through the same number of dimensions. From the results, we observed that 1) On average, low-dimensional bigram reports higher accuracy than

Table 1. Classification accuracy (%) using unigram and low-dimensional bigram of 256 dimensions through RBF and linear kernels.

	RBF kernel		Linear kernel	
Type	Unigram	Bigram (256-d)	Unigram	Bigram (256-d)
JPG	84.00	93.95	80.60	93.15
MP3	87.85	98.45	82.35	94.45
MP4	80.00	93.95	71.95	86.70
MOV	90.15	94.60	89.60	90.35
SQLite	100.00	100.00	95.95	92.70
Overall	**88.40**	**96.19**	84.09	91.47

unigram using either RBF or linear kernel. 2) Compared to that on unigram, the accuracy for most file types are improved using low-dimensional bigram.

To further investigate the classification performance on each file type, Table 2 shows the confusion matrix for the results obtained from bigram through RBF kernel. The confusion matrix presents the instances in terms of percentage where one fragment type is identified as another. The leftmost column is the ground truth of the fragment type and each numbered entry indicates how many percentage of this type is identified as the type indicated in the top row. Higher confusion is observed between MP4 and MOV as both are container format used to store multimedia content.

Table 2. Confusion matrix using low-dimensional bigram of 256 dimensions and RBF kernel (%)

	JPG	MP3	MP4	MOV	SQLite
JPG	**93.95**	0.35	0.90	3.40	1.40
MP3	0.05	**98.45**	1.10	0.30	0.10
MP4	1.70	1.35	**93.95**	2.15	0.85
MOV	0.35	1.35	2.20	**94.60**	1.50
SQLite	0.00	0.00	0.00	0.00	**100.00**

5.2. Dimension of Bigram

Since a major contribution of our work is to reduce the original bigram statistics to a low-dimensional representation, a key factor that determines the result is the reduced dimensions. Table 3 shows the classification performance by varying the number of dimensions in the resulting bigram.

Figure 4 illustrates the corresponding plot of the data in shown in Table 3. From the plots, we observed the following:

1. Overall accuracy for bigram dimensions of 256, 500, 1,000, 1,500, and 2,000 are observed to have marginal differences, with highest accuracy of 96.42% obtained through 1,500 dimensions. Therefore, We proceed with further experiments using the bigram of 1,500 dimensions.
2. The classification performs the best for SQLite among the five fragment types. This is because of the structure of the SQLite data which are mainly text-based as

Table 3. Classification accuracy (%) using varying number of dimensions in bigram

	Bigram dimensions (using RBF kernel)					
Type	256	500	1000	1500	2000	5000
JPG	93.95	94.70	94.60	95.25	95.15	95.35
MP3	98.45	97.95	97.70	98.00	98.10	98.40
MP4	93.95	94.75	94.90	94.85	94.45	91.55
MOV	94.60	94.75	94.45	94.30	95.45	93.10
SQLite	100.00	99.80	99.70	99.70	99.70	99.20
Overall	96.19	96.39	96.27	**96.42**	96.37	95.52

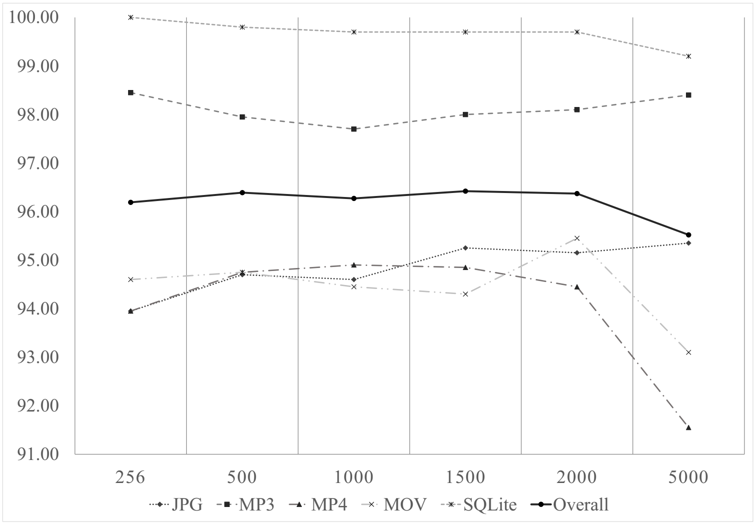


Figure 4. Classification accuracy with respect to number of dimensions of the low-dimensional bigram. X-axis: Dimension of the bigram. Y-axis: Classification accuracy (%).

compared to others, which are compressed multimedia fragment types. The information in the data structure can be presented through the average entropy [16] for the fragment types, as depicted in Figure 5. From this figure, we can see that among the five types, SQLite fragment is the only type that has significant lower entropy. This makes it highly distinguishable from others.

3. For each of the fragment types, increasing the dimension of the bigram can generate favorable or adverse impact on the accuracy, as the inclusion of more dimensions may function as useful information or “noise”. Such result suggests that a more specialized decision on the number of dimensions can be applied based on a targeted fragment type, e.g., higher accuracy can be achieved for SQLite fragments when we use lower dimension of bigram, whereas higher dimension is advantageous in identifying JPG fragments.

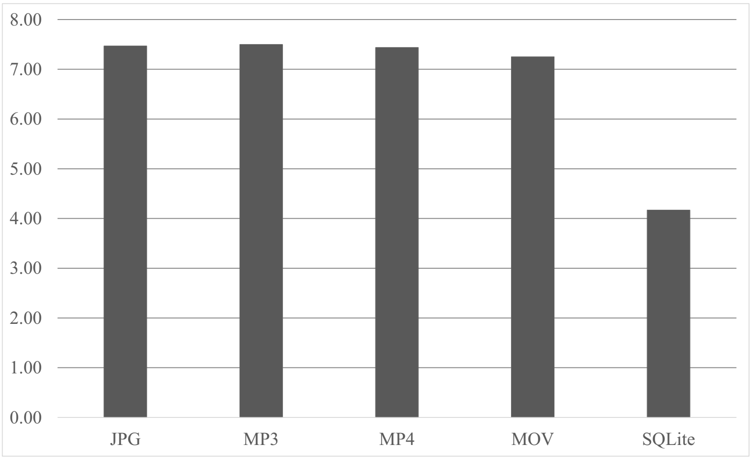


Figure 5. Average entropy of each fragment type.

5.3. Training Size

Using the bigram of 1,500 dimensions, we perform experiments to show the impact of the amount of training fragments on the classification accuracy in our approach.

Table 4. Classification accuracy (%) using varying training size

	Training fragments per type (using RBF kernel)			
Type	2000	3000	4000	5000
JPG	95.25	95.50	95.65	95.70
MP3	98.00	98.35	98.20	98.00
MP4	94.85	95.60	95.95	96.15
MOV	94.30	94.65	94.70	94.95
SQLite	99.70	99.70	99.70	99.70
Overall	96.42	96.76	96.84	**96.90**

Table 5. Training time (sec) using varying training size

Fragments per type	2000	3000	4000	5000
Training time	**97.10**	191.60	308.08	404.03

Figure 6 depicts the plot of the data in shown in Table 4, which shows the impact on the classification accuracy when the number of training fragments changes. Consistent trend can be observed where the classification accuracy of most fragment types increase or stay stabilized when the number of training fragment increases. In our evaluations, the increase in training size produced slightly higher overall accuracy. The corresponding increase in SVM training time for each increment of 1,000 training fragments per type is shown in Table 5. Based on these implications, trade-off between the accuracy and computational workload can be considered accordingly.

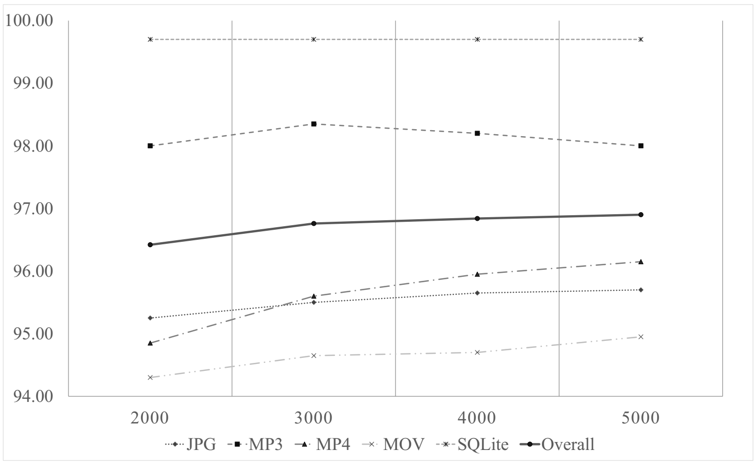


Figure 6. Classification accuracy with respect to training size. X-axis: Number of training fragments per type. Y-axis: Classification accuracy (%).

6. Conclusions and Future Work

In this paper, we studied the utilization of low-dimensional bigram statistics for fragment type classification, focusing on the types that are commonly found on mobile phones. Many fragment classification methods perform classification based on characteristics that are derived through unigram statistics, which fails to capture the order of the bytes. This causes the exploitable contextual byte pattern in the fragments to be disregarded.

The usage of N-gram where $N > 1$ enables us to retain local byte pattern. We presented our classification approach based on the bigram statistics of the fragments. Considering that a high proportion of features in the bigram statistics are redundant, we performed dimensionality reduction through PCA. This enabled us to greatly reduce the feature vectors and the computational workload during the classification process, while preserving the principal components.

The evaluations on our approach shows that we can achieve higher classification accuracy among our targeted fragment types when comparing to the classification results through unigram. Our evaluation results also indicated that the number of dimensions used in low-dimensional bigram has varying impact on different fragment types. This suggests that the number of dimensions of the bigram can be adjusted accordingly, depending on the targeted type.

With the feasibility of such approach, we will look into the inclusion of more fragment types in order to evaluate the performance of our method under a more general digital environment. Further enhancement can also be done by optimizing the parameters of the SVM. Other classification algorithms and feature selection methods or features combination will also be explored and analyzed. Besides, we will also further examine the implementation of our approach into the ultimate file carving and recovery process.

Acknowledgement

This material is based on research work supported by the Singapore National Research Foundation under NCR Award No. NRF2014NCR-NCR001-034.

References

[1] S. Axelsson. The normalised compression distance as a file fragment classifier. *Digital Investigation*, 7:S24–S31, 2010.

[2] S. Axelsson. Using normalized compression distance for classifying file fragments. In *Proceedings of International Conference on Availability, Reliability, and Security*, pages 641–646. IEEE, 2010.

[3] N. L. Beebe, L. A. Maddox, L. Liu, and M. Sun. Sceadan: using concatenated n-gram vectors for improved file and data type classification. *IEEE Transactions on Information Forensics and Security*, 8(9):1519–1530, 2013.

[4] W. C. Calhoun and D. Coles. Predicting the types of file fragments. *Digital Investigation*, 5:S14–S20, 2008.

[5] D. Cao, J. Luo, M. Yin, and H. Yang. Feature selection based file type identification algorithm. In *IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS)*, volume 3, pages 58–62. IEEE, 2010.

[6] C. Chang and C. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[7] S. Fiorillo. Theory and practice of flash memory mobile forensics. In *Proceedings of the 7th Australian Digital Forensics Conference*, page 67, 2009.

[8] S. Fitzgerald, G. Mathews, C. Morris, and O. Zhulyn. Using nlp techniques for file fragment classification. *Digital Investigation*, 9:S44–S49, 2012.

[9] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt. Bringing science to digital forensics with standardized forensic corpora. *Digital Investigation*, 6:S2–S11, 2009.

[10] S. Gopal, Y. Yang, K. Salomatin, and J. Carbonell. Statistical learning for file-type identification. In *10th Internation Conference on Machine Learning and Applications and Workshops*, volume 1, pages 68–73. IEEE, 2011.

[11] C. Ji, L. Chang, L. Shi, C. Wu, Q. Li, and C. J. Xue. An empirical study of file-system fragmentation in mobile storage systems. In *8th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 16)*, 2016.

[12] M. Karresand and N. Shahmehri. File type identification of data fragments by their binary structure. In *Proceedings of the IEEE Information Assurance Workshop*, pages 140–147. IEEE, 2006.

[13] M. Karresand and N. Shahmehri. Oscarfile type identification of binary data in disk clusters and ram pages. In *Proceedings of IFIP International Information Security Conference: Security and Privacy in Dynamic Environments*, pages 413–424. Springer, 2006.

[14] Q. Li, A. Ong, P. Suganthan, and V. Thing. A novel support vector machine approach to high entropy data fragment classification. *Proceedings of the South African Information Security Multi-Conference*, pages 236–247, 2010.

[15] W. Li, K. Wang, S. J. Stolfo, and B. Herzog. Fileprints: Identifying file types by n-gram analysis. In *Proceedings of the 6th Systems, Man and Cybernetics: Information Assurance Workshop*, pages 64–71. IEEE, 2005.

[16] D. MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[17] M. McDaniel and M. H. Heydari. Content based file type detection algorithms. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, pages 10–pp. IEEE, 2003.

[18] Micron. An introduction to nand flash and how to design it in to your next product. *Technical Report TN-29-19*, 2010.

[19] A. Pal and N. Memon. The evolution of file carving. *IEEE Signal Processing Magazine*, 26(2):59–71, 2009.

[20] R. Poisel, M. Rybnicek, and S. Tjoa. Taxonomy of data fragment classification techniques. In *International Conference on Digital Forensics and Cyber Crime*, pages 67–85. Springer, 2013.

[21] V. Roussev and S. L. Garfinkel. File fragment classification-the case for specialized approaches. In *Proceedings of the 4th International IEEE Workshop on Systematic Approaches to Digital Forensic Engineering*, pages 3–14. IEEE, 2009.

[22] A. Smith et al. Us smartphone use in 2015. *Pew Research Center*, 1, 2015.

[23] L. I. Smith. A tutorial on principal components analysis. *Cornell University, USA*, 51:52, 2002.

[24] C. J. Veenman. Statistical disk cluster classification for file carving. In *Pcoceedings of the 3rd International Symposium on Information Assurance and Security*, pages 393–398. IEEE, 2007.

A Systems Approach to Cyber Security
A. Roychoudhury and Y. Liu (Eds.)
© 2017 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/978-1-61499-744-3-143

Hardware Obfuscation Using Different Obfuscation Cell Structures for PLDs

G. SUMATHI[1,2], L. SRIVANI[1], D. THIRUGNANA MURTHY[1,2], ANISH KUMAR[1,2] and K. MADHUSOODANAN[1]
[1]Indira Gandhi Centre for Atomic Research
[2]Homi Bhabha National Institute
Kalpakkam, Tamilnadu-603102, India
E-mail: sumathig@igcar.gov.in

Abstract. In recent years, hardware obfuscation is one of the prominent anti-tamper solutions that are highly used against various hardware security threats such as piracy, cloning, reverse engineering, chip overbuilding, and hardware Trojans. Logic obfuscation is implemented either in the design description (for soft/firm/hard codes) or structure (for the chip) of electronic hardware to intentionally conceal its functionality. Most of the obfuscation schemes enable the circuit operation in two distinct modes such as obfuscated and normal modes. The mode control, mostly implemented by finite state machine, is performed by the application of a specific sequence of input vectors on initialization, called an 'initialization key.' Without the initialization key, it is difficult to comprehend the intended functional behavior of the circuit; hence, circuit tampering or malicious insertion will have a high probability of either becoming functionally benign or easily detectable by conventional logic testing. However, most of the existing obfuscation techniques have used similar obfuscation cell structures throughout the design which in turn leaves a hint to the adversary about circuit obfuscation during reverse engineering. In this paper, we aimed to mitigate this limitation by applying design obfuscation using different obfuscation cells. We performed the hardware obfuscation mechanism for field programmable gate array devices using standard ISCAS'89 benchmark circuits in Actel's ProAsic3 device by Libero SoC v10.1 (free version). We measured the performance overhead using the design parameters such as area, delay, and power.

Keywords. ASIC, Cloning, CPLD, FPGA, Hardware Trojan, Obfuscation, Overbuilding, Piracy, PLD, Reverse Engineering

1. Introduction

With the rapid development of embedded systems and the Internet of things, the programmable logic devices (PLD) and application specific integration circuits (ASIC) are playing a more and more important role in the electronic industry. Over the past few decades, PLDs such as complex PLDs (CPLD) and field programmable gate array (FPGA) devices are extensively used as the basic building modules in most digital systems due to their robust features such as high density, field re-programmability, and faster time-to-market. Besides, usage of PLDs in a design reduces discrete integrated circuits (IC) population and the associated interconnections on the printed circuit board. As a result, the reliability of PLD-based system increases. However, when features such as unit cost, speed, power are considered, ASICs are most suitable devices. They also address the problem of fast obsolescence associated with PLDs.

In general, the design of any digital systems includes outsourced intellectual property (IP) cores, commercial electronic design automation (EDA) tools, and offshore fabrication services. IP cores are the reusable unit of logic, cell, or chip layout that can be in any of following three forms such as, 1) Soft IP, i.e. synthesizable register transfer level (RTL) descriptions; 2) Firm IP, i.e. gate-level designs directly implementable in hardware; and 3) Hard IP, i.e. GDS-II design database. Due to this high third party involvement, the hardware design or IP cores in digital circuits are highly vulnerable to various design security threats such as 1) An attacker may steal and claim ownership of the design, resulting in "piracy attack." 2) An untrusted IP user may perform "duplication or cloning attack" whereas IC foundry may perform "IC overbuilding attack." 3) An attacker may "reverse engineer (RE)" the functionality of an IP/IC. 4) Rogue elements may insert malicious circuits, also known as "hardware Trojans (HT)." For example, RE of any safety critical designs may cause leakage of critical parameters/encrypted keys and insertion of HTs. As a result, it may deny/destroy the system during critical operations and cause serious consequences. As per [1, 2], the semiconductor industry loses \$4 billion annually due to such attacks. Also, it was estimated that the cost of counterfeiting and piracy for G20 nations was U.S. \$450–650 billion in 2008 and U.S. \$1.2–1.7 trillion in 2015 [3]. As these attacks will not only have a negative impact on brand reputation and research & development efforts but also might have the serious impact on systems and operations. As a result, it is at the most required to incorporate the necessary defensive solutions against such IP/IC attacks [4]. Especially, it is very important to ensure the design and data security of safety critical systems such as space, defense, and nuclear as the compromise of which will lead to the disastrous event [5].

Logic obfuscation approach is one such defensive solution that is used to increase the RE complexity. This, in turn, decreases the possibility of piracy, cloning, overbuilding, and successful HT attack. Hardware obfuscation techniques are implemented either in the design description (for soft, firm, hard IPs) or structure (for ICs) of electronic hardware to intentionally conceal its functionality. Basically, it is analogous to software obfuscation techniques which help in protecting against malicious modifications by hiding the functional behavior of a program. Most of the obfuscation schemes enable the circuit operation in two distinct modes such as obfuscated and functional modes. The mode control is performed by the application of a specific sequence of input vectors on initialization, called an 'initialization key.' Without the initialization key, an adversary fails to comprehend the intended functional behavior of the circuit; hence, circuit tampering or malicious insertion by an adversary will have a high probability of either becoming functionally benign or easily detectable by conventional logic testing. Over the last decade, various obfuscation techniques for ASICs/PLDs have been published [6-14]. However, most of the existing obfuscation techniques have used similar obfuscation cell (OC) to conceal the functionality. The repeated OC structure increases the possibility to understand the circuit obfuscation during RE. To mitigate this limitation, this paper discussed the hardware obfuscation technique using different OC structures. We performed the hardware obfuscation mechanism to PLDs using benchmark circuits.

The rest of this paper is organized as follows: Section II discusses the obfuscation, and its classification, requirement of effective obfuscation techniques and existing work on active obfuscation methods; Section III details the implementation of structural modification based netlist obfuscation technique using different OCs; Section IV presents the performance overhead measurements; finally, Section V concludes the paper.

2. Background

2.1. Logic Obfuscation and Classification

In general, logic obfuscation technique modifies either hardware description language (HDL) code or structure such that it is very difficult to read/understand during RE and hence increase the cost and complexity of RE attacks. This, in turn, ensures a certain level of design security by preventing against stealing of original design by analyzing and rebuilding during RE. Also, it avoids the insertion of successful and hard-to-detect HTs. To achieve better RE complexity, Desai [13] discussed the requirements to perform effective hardware obfuscation as follows. 1) Shall be hard to differentiate the obfuscated hardware from the core logic; 2) Shall change the behavior of the circuit dynamically in the obfuscated mode; 3) Original specifications of the chip should not be modified; 4) Shall preserve the same number of inputs and outputs as the original design with minimum design overhead. Besides, as timing is one of the critical design parameters in safety critical applications, obfuscated design should accomplish the system timing requirement.

With respect to the changes in original functionality, hardware obfuscation techniques are widely classified as passive and active obfuscation as shown in Fig. 1. By definition, passive obfuscation techniques modify the circuit description, but it does not affect the functionality. For example, it employs either string substitution by variable renaming or comment removal or structural change by loop unrolling or register renaming of HDL codes [15-17] or obscuring branch functions (e.g. for, while) [18]. In contrast, active obfuscation schemes directly alter the circuit functionality by

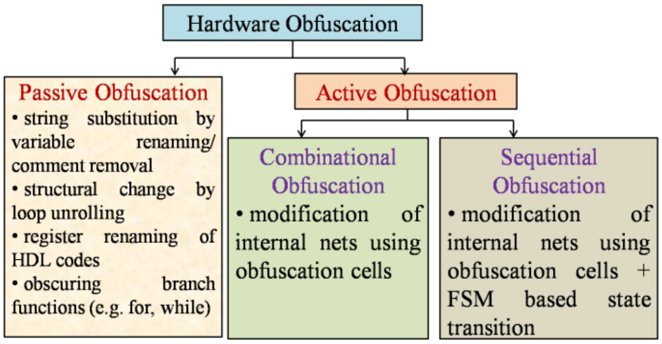

Hardware Obfuscation
Passive Obfuscation
• string substitution by variable renaming/ comment removal
• structural change by loop unrolling
• register renaming of HDL codes
• obscuring branch functions (e.g. for, while)
Active Obfuscation
Combinational Obfuscation
• modification of internal nets using obfuscation cells
Sequential Obfuscation
• modification of internal nets using obfuscation cells + FSM based state transition

Figure 1. Taxonomy of hardware obfuscation technique.

inserting additional logics in it. The active obfuscation techniques can, in turn, be broadly categorized as combinational and sequential obfuscations [19]. In combinational logic obfuscation, the selected internal wires are modified using various OC structures such as XOR/XNOR gates, MUX, etc. The general criterion to select internal wires is that it shall be the non-critical path and one of the inputs is connected to a 1-bit key input. Upon applying the valid key, the obfuscated design will exhibit the correct functionality. However, to build a secure key storage is highly challenging since attackers may control them in a hostile environment and thus, carry out physical attacks [20]. To solve this issue, sequential obfuscation methods are developed. In sequential logic obfuscation, additional obfuscated states are introduced in the finite state machine (FSM) of the original design in addition with the OCs. More often, it is implemented as key-based techniques, and the OCs are driven by FSM outputs, rather than externally stored key values. It enables circuit operation in two distinct modes such as obfuscated and functional mode. Normal functionality is enabled by successful application of the secret key, and the mode switching or state transition function (STF) is defined by (1).

$$\text{STF} = \text{combination (present state, present inputs)} \tag{1}$$

Hence, a particular sequence of input vectors on initialization is required to enter into functional mode. Otherwise, the circuit remains in obfuscated mode and generates incorrect values at the OCs. As a result, the circuit does not perform the actual operation. Since active obfuscation modifies the functionality of the design, it keeps the obfuscated code/circuit as a black block in a design that is not possible by passive methods. This paper concentrates mainly on active obfuscation schemes.

2.2. Need of Obfuscation Techniques for PLDs

In general, logic obfuscation techniques are extensively used to achieve the IP cores/IC protection against piracy, cloning, RE, chip over-building, etc. However, these techniques can also be applied to PLD-based digital designs for most of simulation/structural RE-based security threats. Most of the PLDs design starts with HDL coding as the front-end process. Following by few back-end processes such as synthesis, mapping, fitting/place and route (PAR), and bit stream generation are performed using the vendor specific EDA tools. To ensure design correctness, the HDL code undergoes simulation at pre-synthesis, post synthesis and post fitting/PAR. The netlist is generated in the synthesis stage whereas bitstream is generated after fitting/PAR. Finally, the bitstream is used to program the target device and deployed in the field. With the assumption that front-end processes are completely trusted, major PLD threats are induced by third party EDA tools [21, 22], manipulations at the field [23] and by natural/man-made phenomena such as radiation effects [24]. Most of the static random access memory based FPGAs are volatile. Hence, bitstreams are stored in a separate programmable read only memory chip connected to the FPGA. During every power up, the bitstream configures the FPGA. An adversary may utilize this opportunity to execute the side-channel attacks [25], further extract the bitstream to perform cloning/RE/tampering [26, 27]. In the case of CPLDs or non-volatile memory based FPGAs, theft of programmed boards can be performed to RE it [28]. The basic aim of such attacks can vary from analyzing and rebuilding of competitor’s technology to destroying of critical systems by inserting HTs [29]. To protect the IP cores and to prevent fraud, e.g., by cloning an FPGA or manipulating its content, modern FPGAs offer a bitstream encryption feature. However, a successful attack against the bitstream encryption engine was demonstrated in [30]. To successfully carry out such attacks, an adversary first performs RE of the complete design. Hence, it is necessary to embed a technique that either thwarts RE or increases RE complexity. Obfuscation is one such solution that increases RE complexity and acts as anti- tamper /anti- Trojan techniques to improve hardware security.

2.3. Existing Work on Active Obfuscation Methods

The logic obfuscation technique is one of the most popular IP/IC protection techniques. In 2008, Roy et al. [7, 30] explained the classical combinational logic obfuscation method that conceals the IC designs by inserting the XOR/XNOR gates on selected non-critical wires. One of the inputs to the key gates is the functional input, and the other control input is connected to the common key register. Upon applying the correct key, the obfuscated design will exhibit the intended functionality. In order to avoid the key extraction by image processing-based RE [31], the authors proposed to replace the XOR gate with the XNOR gate and the inverter and, similarly, replace XNOR gates with XOR gates and inverters. However, this approach incurs high area and power overheads due to the logic redesign. To increase RE complexity of obfuscated gates, Rajendran et al. [32] developed an algorithm to insert XOR/XNOR gates at non-resolvable and corruptible gates for a stronger obfuscation; so that, the encrypted circuit is not vulnerable to the fault-analysis attack. Later, Colombier et al. [12] presented the IP/IC protection mechanisms such as logic encryption, logic obfuscation, logic masking, and logic locking using few combinational circuits. Besides, graph analysis-based novel technique was proposed to select the optimal nodes to be modified to achieve effective logic locking of the combinational netlist. In addition to the usage of XOR/XNOR gates as the combinational key gates, MUXs are extensively used. For example, Rajendran et al. [10] proposed two algorithms that insert XOR/XNOR and MUX gates at locations which maximize the hamming distance between correct and incorrect outputs. In 2015, Zhang [11] explained the circuit obfuscation technique using two inputs MUX and physically unclonable functions (PUF) as the lock and key mechanisms. Here, the obfuscated net and its complement are connected to MUX inputs, and PUF key is given to the selection line of MUX. The functionality of OCs cannot be identified unless correct obfuscation keys are given. The chips that are authorized by the designer can only guarantee the correct functionalities. Hence, this obfuscation framework can prevent IC from RE, piracy, and overbuild. Afterward, Wang et al. [33] recommended a scheme to replace the selected logic gates with specially designed MUXs. Here, each input line of the MUX is connected to V_{dd} and V_{ss} by two camouflage connectors, but only one is programmed to be a connection, the other one is programmed to be isolation. Thereby, any 2^m-by-1 MUX can be configured with 2. 2^m camouflage connectors to perform 2^{2^m} possible m-input 1-output boolean functions.

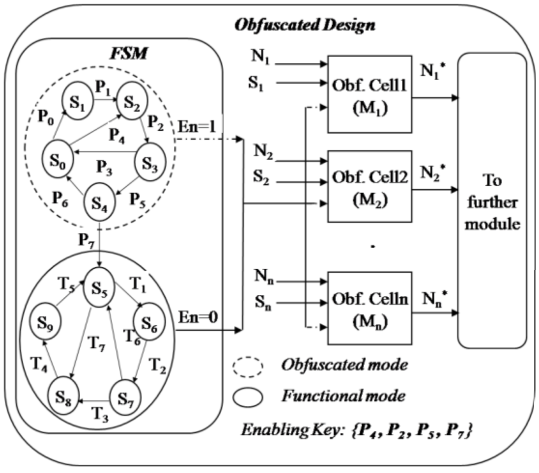
In 2009, Chakraborty et al. [6] explained the structural modification based obfuscation technique and it is the first work to put forth the active sequential obfuscation scheme by means of hardware protection and authentication. The obfuscated design has two modes of operation such as obfuscated and normal mode. The mode switching is performed using FSM. The control outputs from FSM need to be “0” for the proper functioning of the circuit. They are stitched with HF internal nets using appropriate obfuscation structures e.g. XOR gates/combination logics. During power up, a sequence of input vectors as authentication sequence must be applied to drive the circuit to functional mode. Otherwise, the circuit stays at obfuscated mode and does not perform actual operations. The authors extended the work to realize obfuscation at RTL [34] by changing the control/data flow of original circuit. Finally, the design procedure goes through synthesis and optimization and hence protecting hardware blends well into the rest of the logic. In most of the sequential obfuscation techniques, FSM enters into functional mode only when a correct initialization sequence is applied. In 2013, Desai [13] proposed an obfuscation technique where the functional mode is always entered. However, the critical operations (or states) are derived using a variable called "code-word" i.e. the code word is integrated into the STF of FSM. Therefore, interlocked control word generation during state transition ensures the correct functionality of the circuit. Except the FSM-based unlocking schemes, also termed as the sequential obfuscation, random number/signature generator circuits such as PUF can be used as the control circuitry [35, 36]. The physical characteristics of PUFs are unique. Therefore, it generates a device-specific unique challenge-response pairs that are physically unclonable. Wendt et al. [35] proposed PUF and PLD based obfuscation method where PUF and FPGA modules replace the critical portion of original logic. The PUF module implements the original functionality of the replaced circuit and FPGA device generates the corresponding challenges to implement its original functionality. Therefore, the designer holds the control on obfuscated PUF and FPGA logics and hides its functionality.

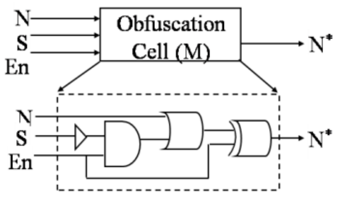
Apart from the application of key-based obfuscation techniques in IP protection, they can also be employed to achieve security against HTs [37-39]. As the circuit modification introduced by obfuscation hides the rareness of the internal circuit nodes, it is very challenging for an adversary to insert hard-to-detect Trojans. Chakraborty et al. [37] exploited this strategy to evade the HT action on the original functionality by making them activate only in the obfuscated mode. In [38], the authors employed the state obfuscation as an HT countermeasure as it tightly couples the obfuscation states with true states, providing more paths from the functional states to the obfuscation states. That is, without the correct key, the adversary cannot successfully tamper the critical control unit without being detected. Finally, the illegal states (i.e. obfuscation states) and illegal state transitions induced by a wrong key are examined to detect the occurrence of HTs.

3. Methodology

3.1. Implementation of Active Hardware Obfuscation Scheme using Different Obfuscation Cell Structures

We applied the structural modification based netlist obfuscation technique using different OCs for PLD-based digital designs. This method aims to achieve a high percentage of simulation/structural mismatch during RE. A combination of lock and key structure accomplish the requirement. The different OCs and FSM-based initialization keys act as the lock and key mechanisms as shown in Fig. 2a. There are two modes of operation in FSM such as obfuscated and functional modes. The control signal "En" derived from FSM is required to be "1" for the obfuscated mode and "0" for the functional mode. It is stitched with HF internal nets (N_1, N_2,..N_n) using different OC structures (M_1, M_2,..M_n) and the obfuscated nets are N_1^*, N_2^*,..N_n^*. The circuit stays in obfuscated mode upon global reset (i.e. initial state). From initial state, a set of inputs/initialization key sequence (e.g. P_4, P_2, P_5, P_7 in Fig. 2a) must be applied to drive the circuit to functional mode. This enabling key sequence acts as authentication





(a) (b)

Figure 2. (a). Structural modification based obfuscation technique, (b). Obfuscation cell structure.

sequence allows the circuit to enter into functional mode; otherwise, circuit stays at obfuscated mode and does not perform the required functionality. Once user authentication is performed, the circuit remains in function mode and ensures correct functionality. Usually, the net with larger fan-out logic cone (i.e. the set of logic bounded by registers, inputs/outputs, or black boxes) will affect the input logic of a comparatively larger number of nets which, in turn, controls more outputs. Hence, HF nets N_1, N_2…N_n are chosen to perform obfuscation of design. Also, large input logic cone of a net is indicative of its higher logic depth. Any change in such a net is likely to alter a large number of primary outputs. Therefore, specific internal signals S_1, S_2…S_n is included as one of the input to few OC structures to increase its input logic cone. The conditions to select internal nets "S" are as follows [6]. 1) It should have a very large fan-in cone, which in turn would substantially expand the logic cone of the obfuscated net. 2) It should not be in the fan-out cone of the obfuscated net. 3) It should not have any net in its fan-in cone that is in the fan-out cone of the obfuscated net. The

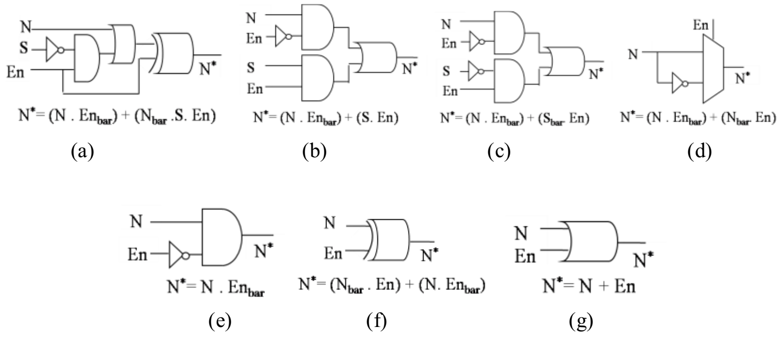

N
S
En
N*
$N^* = (N \cdot En_{bar}) + (N_{bar} \cdot S \cdot En)$
(a)
$N^* = (N \cdot En_{bar}) + (S \cdot En)$
(b)
$N^* = (N \cdot En_{bar}) + (S_{bar} \cdot En)$
(c)
$N^* = (N \cdot En_{bar}) + (N_{bar} \cdot En)$
(d)
$N^* = N \cdot En_{bar}$
(e)
$N^* = (N_{bar} \cdot En) + (N \cdot En_{bar})$
(f)
$N^* = N + En$
(g)

Figure 3. Different obfuscation cell structures, (a). OC_1, (b). OC_2, (c). OC_3, (d). OC_4, (e). OC_5, (f). OC_6, (g). OC_7.

conditions (2) and (3) are essential to prevent any combinational loop in the obfuscated netlist. Otherwise, specific input signals can be OR-ed to generate the internal nets 'S.'

It is well understood that the usage of similar OC as the locking mechanism leaves a hint to the adversary about circuit obfuscation during RE. Hence, we employed different OC structures for logic obfuscation. As shown in Fig. 2b, the value of an obfuscated net (N^*) is a function of (N, En, S). In the proposed method, the obfuscation logic is implemented such that N^*=N in functional mode (when En='0'), and N^*= ('0', '1', N_{bar}, S, S_{bar}) in obfuscated mode (when En='1'). Using these design criteria, five different OC structures are designed as shown in Fig. 3b, Fig. 3c, Fig. 3e, Fig. 3f, and Fig. 3g. It is very clear that an OC can be designed using simple combinational gates such as AND, OR, XOR, etc. or combinational logics. Also, the OCs explained in [6] and [11] are also used as shown in Fig. 3a, Fig. 3d respectively.

Algorithm 1 : Stitching obfuscation cells OC_j at obfuscated nets N_i

```
    Input: N : Set of obfuscated nets $N_i$,
           OC : Set of obfuscation cells $OC_j$
    begin
1.      j :=1;
2.      for each $N_i$ do
3.          Stitch $OC_j$ at $N_i$;
4.          j := j + 1;
5.            if j = = 8 then
6.                j := 1;
7.  end
    Output : Obfuscation cells $OC_j$ are stitched at obfuscated
             nets $N_i$
```

Totally, seven different OC structures are employed to hide the value of high fan-out obfuscated nets. It is already explained that OC with internal signal "S" has a high fan-in logic cone and any change in such a net is likely to alter a large number of primary outputs. The ranking of OCs are performed based on the usage of the signal "S" in OC structures such as OC_1 has the highest rank R_1, OC_2 with R_2, and so on. Finally, OC_7 has the least ranking R_7. During obfuscation, the procedure to select suitable OCs with the highest ranking is explained in Algorithm-1. The input arguments are the set of obfuscated nets (N_i), and the set of OCs (OC_j). As the set of OCs is arranged with the same order as that of their ranking, the parameter "rank of OCs (R_j)" is same as OC_j; hence, it is not used in the algorithm. From the set of seven different OCs, the one with the highest ranking is chosen at each iteration of OC stitching process. For example, it is performed such as OC_1 is stitched at N_1, OC_2 at N_2, and so on up to OC_7 at N_7. Again, the iteration starts from OC_1, OC_2, and so on for the remaining obfuscated nets N_{i-7}.

To increase simulation/structural RE complexity, we performed two scenarios of obfuscation such as 1) For better structural mismatch during RE, we inserted OCs at different numbers of HF nets with minimum initialization sequence length. As per the designer's area constraint, the total number of nets to be obfuscated is chosen. 2) For better functional simulation mismatch during RE, FSM with a different number of obfuscated states and initialization sequence is included in the design with a minimum

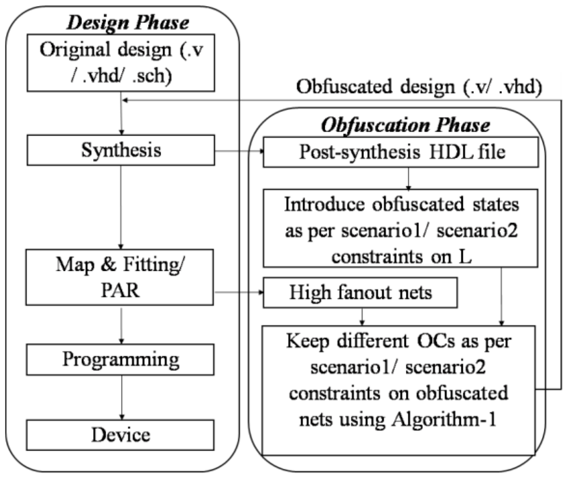

Design Phase
Original design (.v / .vhd/ .sch)
Synthesis
Map & Fitting/ PAR
Programming
Device
Obfuscated design (.v/ .vhd)
Obfuscation Phase
Post-synthesis HDL file
Introduce obfuscated states as per scenario1/ scenario2 constraints on L
High fanout nets
Keep different OCs as per scenario1/ scenario2 constraints on obfuscated nets using Algorithm-1

Figure 4. Flow diagram of structural modification based netlist obfuscation using different OCs.

number of OCs. The initialization sequence length "L" is decided with respect to the system clock cycle (i.e. delay constraint).

The detailed PLD design cycle explained in Section 2.2 is shown in Fig. 4 as the design phase. The obfuscation phase shows the complete flow diagram of structural modification based netlist obfuscation technique. The post-synthesis HDL netlist (.v/. vhd file) generated through back annotation is chosen as the input file for obfuscation process. Using the detailed fitting/PAR report generated by EDA tools, the list of HF nets are extracted, and they are used as obfuscated nets. As per the designer constraints on the total number of OCs (scenario1) and obfuscated states (scenario2), the required number of obfuscated nets and L values are chosen. Finally, the different OCs showed in Fig. 3 are stitched at selected HF nets "N" using the control signal "En," and internal net "S" (optional). The obfuscated netlist is re-synthesized and undergoes remaining back-end processes so that both original and obfuscation logics are merged to increase structural RE complexity.

4. Results and Discussions

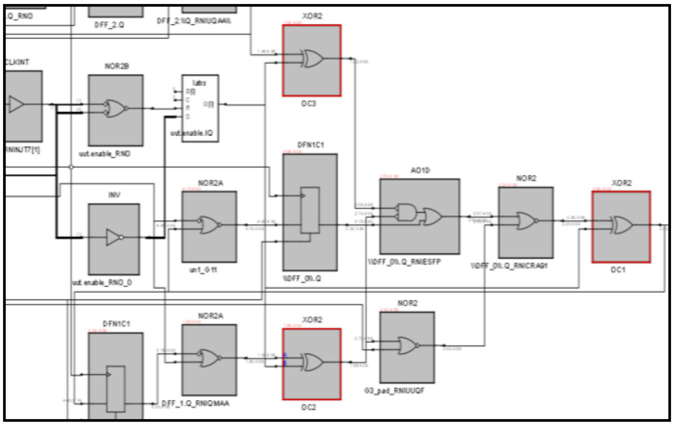
In order to verify the effectiveness of logic obfuscation technique on PLD-based digital designs, we applied the lock and key-based obfuscation technique on various ISCAS'89 sequential benchmark circuits. While structural modification based netlist obfuscation is applicable to both CPLDs and FPGAs, in this section we present the simulation results of FPGAs. The software implementation of this method is performed on Actel ProAsic3 device using Libero SoC v10.1 (free version). The Libero SoC v10.1 has multi-EDA tools support for various back-end processes such as SynplifyPro for synthesis, Modelsim for simulation, Designer for PAR, and FlashPro for device programming. The verilog codes of ISCAS circuits are converted to post-synthesis HDL netlist such as VHDL or Verilog files using SynplifyPro as described in the flow diagram in Fig. 4. The implementation details of the benchmark circuits with respect to I/O and core cells are listed in Table 1.

Table 1. Resource utilization of ISCAS'89 circuits

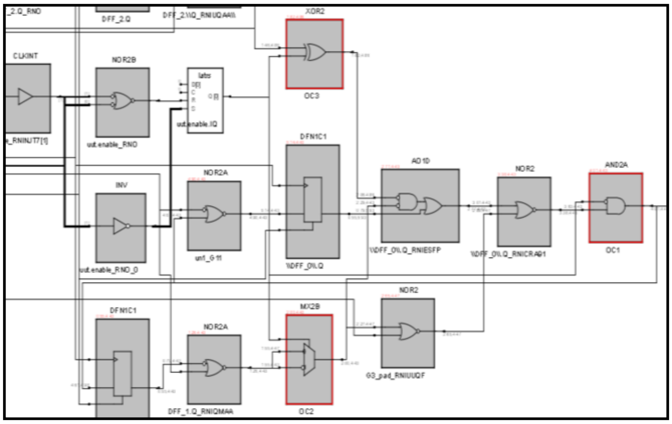
Circuits	**Resource Utilization (cells)**	
	IO	**Core**
S27	7	10
S298	11	67
S344	22	68
S382	11	90
S386	16	76
S420	21	85
S510	28	146
S641	60	108
S820	39	165
S838	37	182
S953	41	272

The proposed structural modification based netlist obfuscation method consists of two major implementations such as 1) FSM-based key mechanism, and 2) OC-based lock mechanism. In general, the key mechanism is implemented as separate FSM logic with varying obfuscation states, i.e. varying L values. The lock mechanism is implemented at the HF internal nets of target circuit using different OC structures shown in Fig. 3. To achieve better structural RE complexity, the HF nets that are directly connected to I/Os are not considered for circuit obfuscation as they are easily decodable nets during RE. We performed two scenarios of obfuscation such as 1) For better structural mismatch during RE, insertion of OCs at different numbers of HF nets with minimum L value. As per the designer's area constraint, the total number of nets to be obfuscated is chosen. For example, we inserted different OCs at 5%, 10%, 15%, and 20% HF nets of total eligible HF nets with L equal to 2. 2) For better functional simulation mismatch during RE, FSM with different L values is included in the design with a minimum number of OCs. The initialization sequence length is decided with respect to the system clock cycle (i.e. delay constraint). For example, we included FSM with L varying from 2 to 7 with 5% obfuscated nets. However, as the obfuscated net percentage and L value increases, the resource utilization, and system clock cycle increases respectively. Based on the designer's requirement either on simulation or structural RE complexity, obfuscation methods described in scenario1 or scenario2 can be applied. The technology view of the obfuscated s27 benchmark circuit using similar and different OCs is shown in Fig. 5a, and Fig. 5b respectively. It is well evident that the usage of similar OC structures (highlighted in red color in Fig. 5a) throughout the design leaves an indication to the adversary about circuit obfuscation. However, employing different OCs (highlighted in red color in Fig. 5b) evade this opportunity to the adversary by increasing the RE complexity.

As logic obfuscation is a key technique incorporated for the hardware security using additional logics, we measured the following three important design parameters. 1. Area- total number of technology mapped resources utilized to implement obfuscation, measured in terms of core and input/output cells; 2. Delay- critical path delay of the circuit, measured in nanoseconds (ns); and 3. Power- total power consumption of logic and it is a sum of static and dynamic power consumptions, measured in milliwatts (mW). The power delay product "PDP" value gives the switching energy value, and it is measured by (2) in picojoules (pJ). Using the measured area "A" and PDP values, we calculated the total design overhead (i.e. area, delay, and power) as APD values by (3).



(a)



(b)

Figure 5. (a). Obfuscated s27 circuit using similar OCs, (b). Obfuscated s27 circuit using different OCs.

$$\text{PDP (pJ)} = \text{Power (mW) X Delay (ns)} \tag{2}$$

$$\text{APD} = \text{Area (A)} + \text{PDP} \tag{3}$$

The calculated APD values for scenario1 and scenario2 are listed in Table 2 and Table 3. Using APD of original "APD_{org}" and obfuscated circuits "APD_{obf}," the percentage increase in APD parameter is measured as the performance overhead "PO" metric as described in (4).

$$\%\text{PO} = \frac{\boldsymbol{APD_{obf} - APD_{org}}}{\boldsymbol{APD_{org}}} \text{X100} \tag{4}$$

The results are listed in Table 2 and Table 3 and shown in Fig. 6a and Fig. 6b respectively. From results, it is clearly evident that the PO calculations i.e. the area/delay/power measurements are smaller than the imposed constraints in most of the scenarios. However, PO is gradually increasing as the percentage of obfuscated nets, and L values are increased. The result also shows that for small target circuit such as s27 alone the PO becomes almost double. While comparing the results of logic obfuscation using similar OCs [14] and different OCs, the average PO values for scenario 1 using different OCs is lesser than obfuscation using similar OCs.

Table 2. PO calculation with varying obfuscated nets (scenario1)

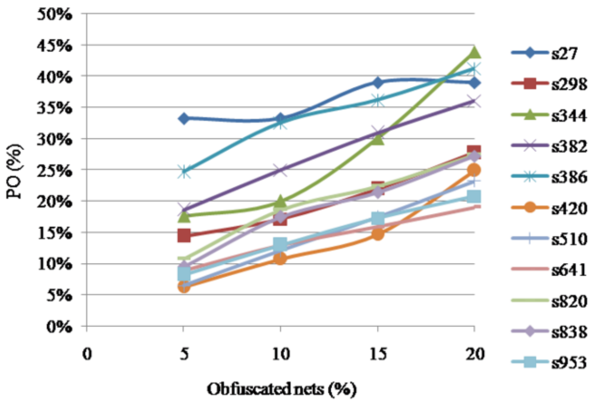
Circuits	**APD$_{org}$**	**5%, L=2**		**10%, L=2**		**15%, L=2**		**20%, L=2**	
		APD$_{obf}$	**PO (%)**	**APD$_{obf}$**	**PO (%)**	**APD$_{obf}$**	**PO (%)**	**APD$_{obf}$**	**PO (%)**
S27	42.60	56.77	33.27	56.77	33.27	59.22	39.00	59.22	39.00
S298	113.46	129.80	14.40	132.86	17.10	138.35	21.94	144.95	27.75
S344	140.02	164.63	17.58	168.03	20.00	182.10	30.06	201.55	43.95
S382	129.09	153.12	18.62	161.31	24.96	169.01	30.92	175.61	36.04
S386	126.69	157.94	24.67	167.91	32.54	172.52	36.18	178.98	41.28
S420	149.25	158.55	6.23	165.31	10.76	171.18	14.69	186.44	24.92
S510	222.97	237.39	6.47	249.88	12.07	261.71	17.37	274.80	23.24
S641	236.65	257.80	8.94	267.15	12.88	274.10	15.82	281.62	19.00
S820	259.58	287.45	10.74	307.87	18.60	317.66	22.38	330.75	27.42
S838	262.91	287.75	9.45	308.75	17.43	319.16	21.40	334.43	27.20
S953	376.46	407.25	8.18	425.38	12.99	441.45	17.26	454.69	20.78
Avg	187.24	208.95	11.59	219.20	19.33	227.86	24.27	238.46	30.05

Table 3. PO calculation with varying initialization sequence lengths (scenario2)

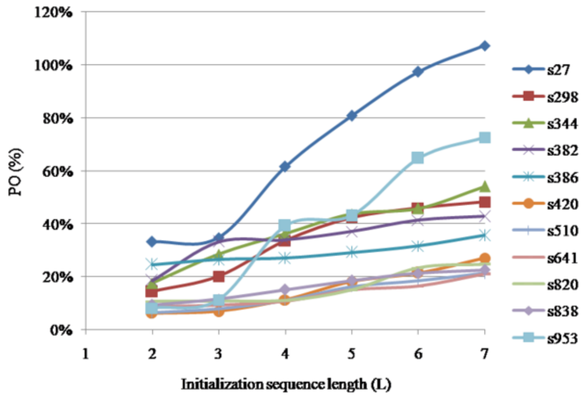
Circuits	**APD$_{org}$**	**L=3, 5%**		**L=5, 5%**		**L=6, 5%**		**L=7, 5%**	
		APD$_{obf}$	**PO (%)**	**APD$_{obf}$**	**PO (%)**	**APD$_{obf}$**	**PO (%)**	**APD$_{obf}$**	**PO (%)**
S27	42.60	57.46	34.87	77.02	80.80	84.13	97.48	88.30	107.27
S298	113.46	136.39	20.20	161.54	42.38	165.65	45.99	168.20	48.24
S344	140.02	180.02	28.57	201.29	43.76	204.27	45.89	215.86	54.17
S382	129.09	171.90	33.16	176.94	37.07	182.47	41.35	184.31	42.78
S386	126.69	160.32	26.55	163.71	29.23	166.81	31.67	171.80	35.61
S420	149.25	159.70	7.00	176.49	18.25	181.33	21.49	189.73	27.12
S510	222.97	240.61	7.91	259.39	16.33	264.35	18.56	270.64	21.38
S641	236.65	258.84	9.38	272.36	15.09	275.76	16.53	286.36	21.00
S820	259.58	287.89	10.91	298.75	15.09	320.35	23.41	323.58	24.66
S838	262.91	293.36	11.58	311.82	18.60	319.15	21.39	322.36	22.61
S953	376.46	418.92	11.28	538.54	43.05	620.38	64.79	650.07	72.68
Avg	187.24	215.04	18.31	239.81	32.69	253.15	38.96	261.02	43.41

5. Conclusion and Future Work

Reverse engineering of any safety critical designs may cause leakage of critical design parameters/encrypted keys and insertion of hardware Trojans to deny/destroy the critical systems. To ensure the design and data security of such systems, logic obfuscation techniques are widely adopted. Though various hardware obfuscation techniques are proposed during last few decades, most of the techniques used similar OC structures. This, in turn, enables an adversary to understand the logic obfuscation during the RE process. Hence, the structural modification based netlist obfuscation technique using different OC structures is proposed. In addition with logic obfuscation, usage of different OCs increases the complexity of RE. As per the need of simulation/structural RE complexity as well as the area and delay constraints, obfuscation with varying obfuscated nets and initialization sequence length can be



(a)



(b)

Figure 6. (a). PO calculation with varying obfuscated nets, (b). PO calculation with varying initialization sequence lengths (L).

applied. Results reveal that the proposed approach appears to be a quite useful technique for PLD designs, and design overhead is well below the design constraints. However, the metric to define the complexity of RE is required to be derived. To address the same, the future research shall concentrate on the development of a novel obfuscation metric to quantify the percentage of circuit hiding is achieved. Also, incorporating the randomization in the selection of OCs shall improve the RE complexity. In addition, an automation and integration of the obfuscation process with the regular design flow will be highly helpful for the system designers to obfuscate and preserve their designs.

Acknowledgment

G. Sumathi gratefully acknowledges the grant of research fellowship from Indira Gandhi Center for Atomic Research, Department of Atomic Energy, India.

References

[1] "Managing the risks of counterfeiting in the information technology" [Online]. Available: http://www.agmaglobal.org/press_events/press_docs/Counterfeit_WhitePaper_Final.pdf, accessed Dec. 14, 2016.

[2] "Innovation is at risk as semiconductor equipment and materials industry loses up to $4 billion annually due to IP infringement" [Online]. Available: http://semi.org/en/innovation-risk-losses-4-billion-annually-due-ip-infringement, accessed Dec. 14, 2016.

[3] "Estimating the global economic and social impacts of counterfeiting and piracy," [Online]. Available: http://www.illicittrademonitor. com/reports/article/estimating-the-global-economic-and-social-impacts-of-counterfeiting-and-piracy/, accessed Dec. 14, 2016.

[4] B. Colombier, and L. Bossuet, Survey of hardware protection of design data for integrated circuits and intellectual properties, *IET Comput. Digital Techn.* **8** (2014), 274-287.

[5] Department of Defense, Defense Science Board (DSB) "Study on High Performance Microchip Supply" [Online]. Available: http://www. aoq.osd.mil/dsb/reports/ADA435563.pdf, accessed Dec. 14, 2016.

[6] R.S. Chakraborty and S. Bhunia, HARPOON: An obfuscation-based SoC design methodology for hardware protection, *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **28** (2009), 1493-1502.

[7] J.A. Roy, F. Koushanfar, and I. L. Markov, Ending piracy of integrated circuits, *Computer* **43** (2010), 30-38.

[8] A. Baumgarten, A. Tyagi, and J. Zambreno, Preventing IC piracy using reconfigurable logic barriers, *IEEE Des. Test Comput.* **27** (2010), 66-75.

[9] W.P. Griffin, A. Raghunathan, and K. Roy, CLIP: Circuit level IC protection through direct injection of process variations, *IEEE Trans. Very Large Scale Integr. Syst.* **20** (2012), 791-803.

[10] J. Rajendran, et al. Fault analysis-based logic encryption, *IEEE Trans.Comput.* 64 (2015), 410-424.

[11] J. Zhang, A Practical Logic Obfuscation Technique for Hardware Security, *IEEE Trans. VLSI Syst.* **24** (2015), 1193-1197.

[12] B. Colombier, L. Bossuet, and D. Hély, From secured logic to IP protection, *Microprocessors and Microsystems*, In Press, Corrected Proof- Note to users, Available online 26 February 2016.

[13] A.R. Desai, et al. Interlocking Obfuscation for Anti-Tamper Hardware, *Proc. 8th Annual Workshop Cyb. Sec. & Inf. Intelligence Research* (2013).

[14] G. Sumathi, L. Srivani, D. Thirugnana Murthy, Anish Kumar, K. Madhusoodanan, and S.A.V. Satya Murty, Structural Modification based Netlist Obfuscation Technique for PLDs, *Proc. IEEE Int. Conf. Wireless Communications, Signal Processing and Networking* (2016), 1418-1423.

[15] M. Brzozowski, and V.N. Yarmolik, Obfuscation as intellectual rights protection in VHDL language, *Proc. 6th Int. Conf. Comput. Inf. Syst.* (2007), 337-340.

[16] U. Meyer-Base, E. Castillo, and G. Botello, Intellectual property protection (IPP) using obfuscation in C, VHDL, and Verilog coding, *Proc. SPIE Independent Component Analyses, Wavelets, Neural Networks, Biosystems, and Nanoengineering IX* **8058** (2011).

[17] V.V. Sergeichik, A.A. Ivaniuk, and Chang, Obfuscation and watermarking of FPGA designs based on constant value generators, *Proc. 14th Int. Symp. Integrated Circuits* (2014), 608-611.

[18] M. Kainth, L. Krishnan, C. Narayana, S.G. Virupaksha, and R. Tessier, Hardware-assisted code obfuscation for FPGA soft microprocessors, *Proc. Design Auto. Test Eur.* (2015), 127-132.

[19] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, Security analysis of integrated circuit camouflaging, *Proc. ACM SIGSAC conf. Computer & communications security* (2013), 709-720.

[20] V. Immler, M. Hennig, L. Kürzinger, and G. Sigl, Practical Aspects of Quantization and Tamper-Sensitivity for Physically Obfuscated Keys, *Proc. Third Workshop on Cryptography and Security in Computing Systems* (2016), 13-18.

[21] A. Roy, F. Koushanfar, and I. L. Markov, Extended abstract: Circuit CAD tools as a security threat, *Proc. IEEE Workshop on Hardware Oriented Security Trust* (2008), 65-66.

[22] G. Qu, and Y. Yuan, Design things for the internet of things—an EDA perspective, *Proc. IEEE/ACM Int. Conf. Computer Aided Design* (2014), 411-416.

[23] "N.S.A. Devises Radio Pathway Into Computers" [Online]. Available: http://www.nytimes.com/2014/01/15/us/nsa-effort-pries-open-computers-not-connected-to-internet.html?_r=0, accessed Dec. 14, 2016.

[24] "Time To Refocus on The EMP Threat" [Online]. Available: http://www.defensenews.com/story/defense/commentary/2015/08/18/time-refocus-emp-threat/31915021/, accessed Dec. 14, 2016.

[25] A. Moradi, D. Oswald, C. Paar, and P. Swierczynski, Side-channel attacks on the bitstream encryption mechanism of Altera Stratix II, *Proc. ACM/SIGDA Int. Symp. FPGAs* (2013), 91-100.

[26] J.B. Note, and E. Rannaud, From the bitstream to the netlist, *Proc. ACM/SIGDA Int. Symp. FPGAs* (2008), 264-264.

[27] F. Benz, A. Seffrin, and S. A. Huss, Bil: A tool-chain for bitstream reverse-engineering, *Proc. 22nd Int. Conf. Field Program. Logic Appl.* (2012), 735-738.
[28] "Design Security in Non-volatile Flash and Antifuse FPGAs" [Online]. Available: http://www.microsemi.com/document-portal/doc_view/131564-designsecurity-wp, accessed Dec. 14, 2016.
[29] "Stopping Hardware Trojans in Their Tracks" [Online]. Available: http://spectrum.ieee.org/semiconductors/design/stopping-hardware-trojans-in-their-tracks, accessed Dec. 14, 2016.
[30] J.A. Roy, F. Koushanfar, and I.L. Markov, EPIC: Ending piracy of integrated circuits, *Proc. Des. Autom. Test Eur.* (2008), 1069-1074.
[31] R. Torrance and D. James, The state-of-the-art in semiconductor reverse engineering, *Proc. 48th ACM/EDAC/IEEE Design Autom. Conf.* (2011), 333–338.
[32] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, Security Analysis of Logic Obfuscation, *Proc. IEEE/ACM Design Automation Conf.* (2012), 83-89.
[33] X. Wang et al., Secure and Low-Overhead Circuit Obfuscation Technique with Multiplexers, *Proc. 26th Great Lakes Symposium on VLSI* (2016), 133-136.
[34] R. S. Chakraborty and S. Bhunia, RTL hardware IP protection using key-based control and data flow obfuscation, *Proc. Int. Conf. on VLSI Design* (2010), 405-410.
[35] J.B. Wendt, and M. Potkonjak, Hardware obfuscation using PUF-based logic, *Proc. IEEE/ACM Int. Conf. Computer-Aided Design* (2014), 270-277.
[36] D. Li et al., Hardware IP Protection through Gate-Level Obfuscation, *Proc. 14th Int. Conf. Computer-Aided Design and Computer Graphics* (2015), 186-193.
[37] R. Chakraborty, and S. Bhunia, Security against Hardware Trojan through a Novel Application of Design Obfuscation, *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design* (2009), 113-116.
[38] J. Frey, and Q. Yu, Exploiting state obfuscation to detect hardware trojans in NoC network interfaces, *Proc. IEEE 58th Int. Midwest Symp. Circuits and Systems* (2015), 1-4.
[39] A. Nejat, D. Hely, and V. Beroulle, Facilitating side channel analysis by obfuscation for Hardware Trojan detection, *Proc. 10th Int. Design & Test Symp.* (2015), 129-134.

Subject Index

Author Index