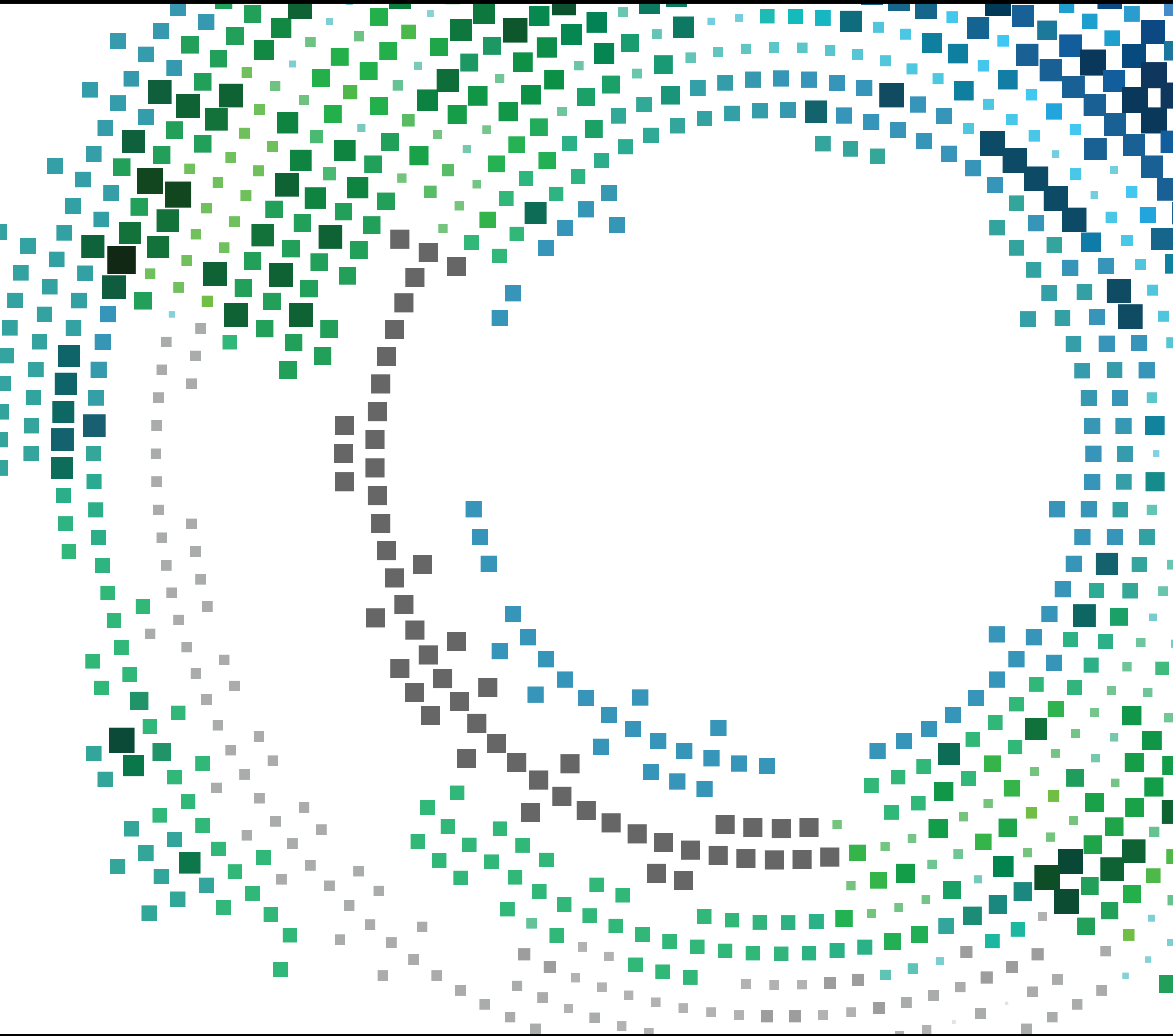
Mobile Information Systems

Advances in Mobile Security Technologies

Guest Editors: Jeong Hyun Yi, Aziz Mohaisen, Sean Yang, and Ching-Hsien Hsu



Advances in Mobile Security Technologies

Mobile Information Systems

Advances in Mobile Security Technologies

Guest Editors: Jeong Hyun Yi, Aziz Mohaisen, Sean Yang, and Ching-Hsien Hsu

Contents

Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 9501918, 2 pages
http://dx.doi.org/10.1155/2016/9501918

Editorial

Advances in Mobile Security Technologies

Jeong Hyun Yi,[1] Aziz Mohaisen,[2] Sean Yang,[3] and Ching-Hsien Hsu[4]

[1] *School of Software, Soongsil University, Seoul 06978, Republic of Korea*
[2] *Department of Computer Science and Engineering, The State University of New York, University at Buffalo, Buffalo, NY 14260, USA*
[3] *Information Technology, Georgia Gwinnett College, Lawrenceville, GA 30043, USA*
[4] *Department of Computer Science and Information Engineering, Chung Hua University, Hsinchu 30012, Taiwan*

Correspondence should be addressed to Jeong Hyun Yi; jhyi@ssu.ac.kr

Received 6 June 2016; Accepted 6 June 2016

Mobile devices such as smartphones and Internet tablets have achieved computing and networking capabilities comparable to traditional personal computers. An explosion of mobile devices being used for work has also become a source of pain for adopting users and organizations. For example, the widespread presence of information-stealing applications raises substantial security and privacy concerns. Recently, there has been an explosion of the Internet of things which will make us increasingly rely on intelligent, interconnected devices in every aspect of our lives. Examples include smart systems, smart vehicles, and wearable devices such as smart watches and digital glasses. In such always connected environment, mobile devices still will be an essential gateway from the personal point of view and cross-linking of things offers new possibilities to influence business intelligence and to exchange various data items. This may also lead to a variety of new potential risks concerning both security and privacy, which must be considered. Thus, it is tremendously essential to develop security technologies for mobile system, in particular, dealing with mobile devices.

Hence, the main motivation for this special issue is to bring together researchers, practitioners, policy makers, and hardware and software developers of mobile systems to explore the latest understanding and advances in the security and privacy for mobile devices, applications, and systems.

We received 36 submissions covering all areas of mobile security. Every submission received at least three reviews. Based on the paper topics, review feedback from the reviewers, and follow-up discussions among guest editors, we selected seven papers for this special issue.

The paper titled "The Effective Ransomware Prevention Technique Using Process Monitoring on Android Platform" by S. Song et al. proposes an effective method to prevent the attacks of modified ransomware on Android platform. The proposed technique specifies and intensively monitors processes and specific file directories using statistical methods based on processor usage, memory usage, and I/O rates so that the process with abnormal behaviors can be detected. The proposed technique can detect ransomware even if you do not save its patterns. Its speed of detection is very fast because it can be implemented in Android source code instead of mobile application.

The paper titled "Learning-Based Detection of Harmful Data in Mobile Devices" by S.-W. Jang and G.-Y. Kim proposes a method to assess the harmfulness of input images automatically based on an artificial neural network. The proposed method first detects human face areas based on the MCT features from the input images. Next, based on color characteristics, this study identifies human skin color areas along with the candidate areas of nipples, one of the human body parts representing harmfulness. Finally, the method removes nonnipple areas among the detected candidate areas using the artificial neural network.

The paper entitled "Enhancing the Security of Personal Identification Numbers with Three-Dimensional Displays" by M.-K. Lee et al. provides a novel solution based on three dimensions, particularly suitable for glasses-free three-dimensional (3D) displays found in many smartphones and handheld game consoles. A user at the "3D spot" may log in easily, while nearby shoulder-surfers gain no advantage. A detailed experimental usability analysis is performed to demonstrate the effectiveness of the proposed scheme in comparison to the existing methods.

The paper "Anomaly Detection for Internet of Vehicles: A Trust Management Scheme with Affinity Propagation" by S. Yang et al. introduces a trust-based anomaly detection scheme for IVs, where some malicious or incapable vehicles are existing on roads. The proposed scheme works by allowing IVs to detect abnormal vehicles, communicate with each other, and finally converge to some trustworthy cluster heads (CHs). Periodically, the CHs take responsibility for intracluster trust management. Moreover, the scheme is enhanced with a distributed supervising mechanism and a central reputation arbitrator to assure robustness and fairness in detecting process.

The paper titled "Security Analysis and Improvement of Fingerprint Authentication for Smartphones" by Y.-H. Jo et al. identifies a few vulnerabilities in one of the currently deployed smartphones equipped with fingerprint verification service by analyzing the service application. Y.-H. Jo et al. demonstrate actual attacks via two proof-of-concept codes that exploit these vulnerabilities. By the first attack, a malicious application can obtain the fingerprint image of the owner of the victimized smartphone through message-based interprocess communication with the service application. In the second attack, an attacker can extract fingerprint features by decoding a file containing them in encrypted form.

The paper entitled "Function-Oriented Mobile Malware Analysis as First-Aid" by J. Jang and H. K. Kim proposes a novel method for function-oriented malware analysis approach based on analysis of suspicious API call patterns. Instead of extracting API call patterns for malware in each family, J. Jang and H. K. Kim focus on extracting such patterns for certain malicious functionalities. The proposed method dumps memory sections, where an application is allocated and extracts suspicious API sequences from bytecode by comparing with predefined suspicious API lists. By matching API call patterns with our functionality database, the proposed method determines whether they are malicious.

The paper titled "A Novel Iterative and Dynamic Trust Computing Model for Large Scaled P2P Networks" by Z. Tan et al. presents an iterative and dynamic trust computation model named IDTrust (Iterative and Dynamic Trust model) according to these properties. First of all, a three-layered distributed trust communication architecture was presented in IDTrust so as to separate evidence collector and trust decision from P2P service. Then an iterative and dynamic trust computation method was presented to improve efficiency, where only latest evidences were enrolled during one iterative computation. On the basis of these, direct trust model, indirect trust model, and global trust model were presented with both explicit and implicit evidences.

In closing, we believe the readers will find these papers interesting and useful, and we hope that they will enjoy them.

Acknowledgments

We would like to thank all of the authors who have submitted their research to this special issue. We are also grateful for the many experts in the field who have participated in the review process and provided helpful suggestions to the authors for improving their work. We would like to express our gratitude to the Editor-in-Chief, Dr. David Taniar, for his support in bringing forth this special issue. We also appreciate Global Research Laboratory Project through the National Research Foundation (NRF-2014K1A1A2043029) that tremendously helped support our effort.

Jeong Hyun Yi
Aziz Mohaisen
Sean Yang
Ching-Hsien Hsu

Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 2946735, 9 pages
http://dx.doi.org/10.1155/2016/2946735

Research Article

The Effective Ransomware Prevention Technique Using Process Monitoring on Android Platform

Sanggeun Song, Bongjoon Kim, and Sangjun Lee

School of Computing, Soongsil University, Sangdo-ro, Dongjak-gu, Seoul 06978, Republic of Korea

Correspondence should be addressed to Sangjun Lee; sangjun@ssu.ac.kr

Received 31 December 2015; Accepted 10 March 2016

Academic Editor: Seung Yang

Due to recent indiscriminate attacks of ransomware, damage cases including encryption of users' important files are constantly increasing. The existing vaccine systems are vulnerable to attacks of new pattern ransomware because they can only detect the ransomware of existing patterns. More effective technique is required to prevent modified ransomware. In this paper, an effective method is proposed to prevent the attacks of modified ransomware on Android platform. The proposed technique specifies and intensively monitors processes and specific file directories using statistical methods based on Processor usage, Memory usage, and I/O rates so that the process with abnormal behaviors can be detected. If the process running a suspicious ransomware is detected, the proposed system will stop the process and take steps to confirm the deletion of programs associated with the process from users. The information of suspected and exceptional processes confirmed by users is stored in a database. The proposed technique can detect ransomware even if you do not save its patterns. Its speed of detection is very fast because it can be implemented in Android source code instead of mobile application. In addition, it can effectively determine modified patterns of ransomware and provide protection with minimum damage.

1. Introduction

Ransomware [1] is a type of malware that uses malicious codes to intrude the system before users notice it, to encrypt important files, to require money using encrypted files as a hostage, and to give monetary damages to users. The rapid growth of the mobile market has been the main target of hackers to obtain illegal gains by using ransomware. The market share of Korea's Android OS is approximately 80% of the total share of smartphone market as shown in Table 1. Compared to other OS such as iOS, Windows Phone, or Blackberry, Android holds a high market share close to monopoly, while the others combined have less than 15% share in the mobile device market [2]. The share of the Android platform is so high that the platform is the main target of ransomware attacks. Damage cases of Android-based smartphones are continuously growing recently.

Traditional vaccine system can detect a system if it is infected with ransomware and cure it. However, it cannot prevent attacks by ransomware without obtaining information on the ransomware. In addition, files cannot be recovered without the encryption key because files are already encrypted even if the traditional vaccine system can remove the ransomware [3]. Users can avoid infections by updating the vaccine system from time to time. However, this method has limited efficacy. Existing vaccine system can detect ransomware using intrusion detection method based on files [4]. However, this approach cannot detect modified ransomware with new patterns because it can only prevent ransomware based on analysis information of the ransomware. Therefore, an active instead of a passive prevention method is urgently required.

In this paper, a ransomware prevention technique on Android platform is proposed. The proposed method can monitor file events that occurred when the ransomware accesses and copies files. This technique can detect and remove the ransomware using the CPU and I/O usage as well as the information stored in the DB. This proposed method can detect modified patterns of ransomware without obtaining information about the ransomware. In addition, it can be implemented on the kernel and framework source of Android so that it can detect ransomware relatively

Table 1: Smart device operating system market share [2].

Period	Android	iOS	Windows Phone	Blackberry OS	Others
2015	82.8%	13.9%	2.6%	0.3%	0.4%
2014	84.8%	11.6%	2.5%	0.5%	0.7%
2013	79.8%	12.9%	3.4%	2.8%	1.2%
2012	69.3%	16.6%	3.1%	4.9%	6.1%

faster than other programs that run at the application level. Furthermore, it can continuously monitor the ransomware without separately downloading or updating.

The remainder of this paper is organized as follows. Related work is briefly discussed in Section 2. Our proposed approach is described in Section 3. Evaluation of the proposed approach is given in Section 4. Finally, several concluding remarks are given in Section 5.

2. Related Research

2.1. Ransomware. Ransomware spreading methods are similar to those of malicious code Trojan Horse [5] that contains malicious routine and pretends as a normal program. Ransomware intrudes into users' devices after pretending as a normal application such as Trojan Horse. Ransomware restricts the use of the system in various ways after intruding the system. It is mainly classified into the following three types: Scareware, Lock-Screen, and Encrypting [6].

(i) Scareware. It informs users that the device has been infected with malicious codes. It suggests the purchase of fake antivirus programs to treat them. It finally extorts money from the user.

(ii) Lock-Screen. It disables users' PC in any way. It locks the system so that the users are not able to run the operating system when executing the system. When a user runs his system, it disables the operating system and sends the message that your PC has illegal contents that you will be fined by impersonating FBI or government agencies.

(iii) Encrypting. This is the most serious type of ransomware. It prevents the use of important files in your device by encrypting them. It extorts money by encrypting users' files in PCs and letting users deposit the ransom for files to a virtual account to decrypt.

Ransomware accesses users and gives damage to them in various ways. For example, CryptoLocker [7] can encrypt files in PC. Reveton [8, 9] will impersonate law enforcement agencies such as FBI. SimpleLocker [10] targets smartphone users of the Android environment. This ransomware can be serious security threat to cloud computing [11] as it becomes the basic infrastructure of information system.

2.2. Existing Techniques

2.2.1. Process Using Hash Information. The processing method of CryptoLocker is to compare Hash information. CryptoLocker generates files encrypted with ".encrypted" [12]. The encrypted files are then added to the Hash Information. Signature, Public Key values, and their sizes will increase. Recovery tools are generally used to process CryptoLocker. They include different decryption key index information by infected users. Recovery tools compare Hash information and encrypted files in the data files, confirm the validation of key from key index information stored therein, and then proceed to decoding [13].

By looking at encrypted files' recovery methods used in existing vaccines, these methods obtain a sample by decompiling the ransomware and perform decryption using the decryption key found by the code analysis of the sample [14]. There is a risk that when a new ransomware appears, users have to wait until a security company finds the decryption key value through sample analysis. Intelligent sensing techniques are required to detect new patterns of ransomware because ransomware constantly threatens the safety of mobile device.

System-based behavior detection technique [15] is based on the detection of occurrences of several behaviors in a computer system. It performs "integrity checking" and "behavior blocking" [16, 17]. Integrity checking technique conducts frequent inspections in order to confirm the integrity of the computing system. This approach calculates and writes the Hash values for execution files and directories on a clean computer system that is not infected by malware. Behavior blocking technique monitors all actions within the computer system. When a suspicious action occurs in similar way of malicious infections, this approach tracks the cause of executable file and blocks the execution of a suspicious action so that it has no progress.

2.2.2. Process Using CPU and I/O Usage. Statistical technique is one malware analysis technique that detects abnormal behaviors by analyzing the resources of the system. NIDES (Next-generation Intrusion Detection Expert System) [18] of SRI (Stanford Research Institute) International is a typical system based on statistical techniques. NIDES sets a goal of detecting abnormal behaviors that occurred in the system with a profiling technique after collecting Processor usage, I/O rate, Memory usage, and so forth, over a long time. Korea Electronics and Telecommunications Research Institute uses the technique using the mean difference of CPU or Memory usage in order to provide a reliable service on the host [19, 20]. However, this technique only operates against the attacks of DDoS. In this paper, a technique is proposed to prevent the intrusion of ransomware on Android platform based on statistical methods using Process, Memory, and Storage I/O usage.

2.3. Android Application Permissions Analysis. Android market applications demand Android system permissions in order to perform the correct operation. Applications registered in an official store show users permission requirements when they are downloaded. However, ordinary users may

Table 2: Difference in permission between Ransomware App and Normal App [21, 22].

Type	Permission	Behavior	Ransomware App	Normal App
System	GET_TASK	Allows an application to get information about currently or recently running tasks	O	O
	WRITE_SETTINGS	Allows an application to read or write system settings	O	O
	SYSTEM_ALERT_WINDOW	Allows an application to alert system	O	O
	RECEIVE_BOOT_COMPLETED	Allows an application to receive the ACTION_BOOT_COMPLETED that is broadcasted after the system finishes booting	O	X
	READ_PHONE_STATE	Allows read only access to phone state	O	X
	READ_EXTERNAL_STORAGE	Allows an application to read from external storage	O	X
	WRITE_EXTERNAL_STORAGE	Allows an application to write to external storage	O	X
	WAKE_LOCK	Allows using PowerManager WakeLocks to keep Processor from sleeping or screen from dimming	O	X
	GET_ACCOUNTS	Allows access to the list of accounts in Accounts Service	O	X
	BIND_DEVICE_ADMIN	Must be required by device administration receiver to ensure that only the system can interact with it	O	X
	DISABLE_KEYGUARD	Allows applications to disable the keyguard if it is not secure	O	X
SMS	RECEIVE_SMS	Allows an application to receive SMS messages	O	O
	SEND_SMS	Allows an application to send SMS messages	O	O
	READ_SMS	Allows an application to read SMS messages	O	X
Contact	READ_CONTACTS	Allows an application to read user's contacts data	O	O
	READ_CALL_LOG	Allows an application to read the user's call log	O	O
	CALL_PHONE	Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call	O	O
Network	INTERNET	Allows applications to open network sockets	O	X
	ACCESS_NETWORK_STATE	Allows applications to access information about networks	O	X
	READ_HISTORY_BOOKMARKS	Allows an application to read the user's browsing history and bookmarks	O	X

unintentionally download or run applications without carefully looking at them. Ransomware distributors will distribute the ransomware and pretend as a normal application on an official store using this security weakness.

To design the proposed method, the different kinds and functions of permissions on the Android system and permissions needed by ransomware are analyzed. Permissions to adversely affect the Android system are largely classified as System, SMS, Contact, and Location [21]. Difference in permissions between Ransomware App and Normal App is shown in Table 2. A total of 14 kinds of ransomware that appeared between 2014.01 and 2015.09 based on the report of virustotal [22] are included in the comparison (Table 2).

The functions of the corresponding permissions are not necessarily safe. These permissions access a lot of information, including the configuration information of the device, the list of applications, resource statistics, and personal information such as location information and SMS information. Normal applications use these permissions. Therefore, users generally agree to install applications without doubt, even when it is the ransomware that requires permissions for the System, SMS, Contact, and Location.

3. The Proposed Technique

To have efficient implementation, the proposed technique is designed with three modules: Configuration, Monitoring, and Processing (Figure 1). Configuration module generates a monitoring list table for a smooth operation of the proposed method. It is the module for the initial setting. Monitoring module is responsible for monitoring Processor, Memory, and Storage I/O usages of every process in real time based on statistical techniques. Finally, processing module determines the handling of the process suspected as ransomware by the Monitoring module and makes an exception or isolation of the process.

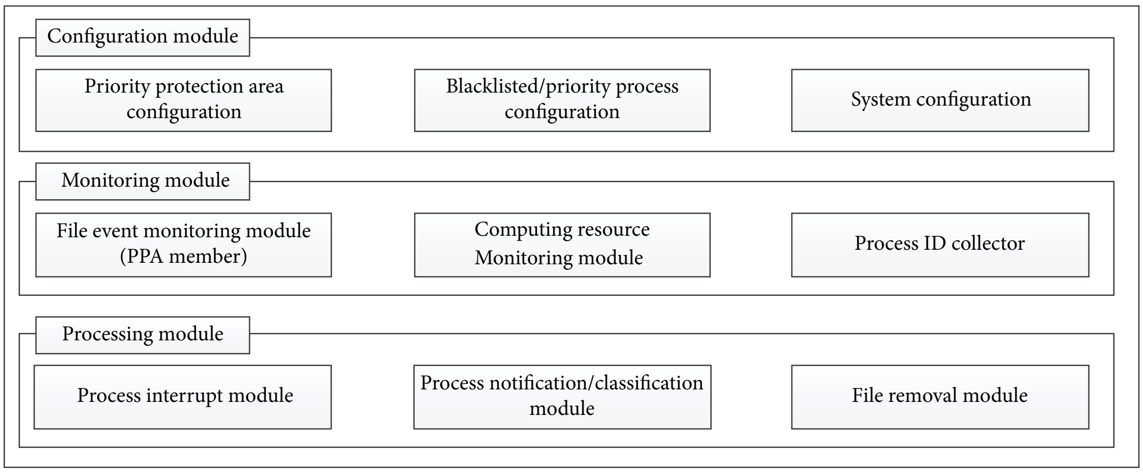

Configuration module
Priority protection area configuration
Blacklisted/priority process configuration
System configuration
Monitoring module
File event monitoring module (PPA member)
Computing resource Monitoring module
Process ID collector
Processing module
Process interrupt module
Process notification/classification module
File removal module

Figure 1: Overview of the proposed system.

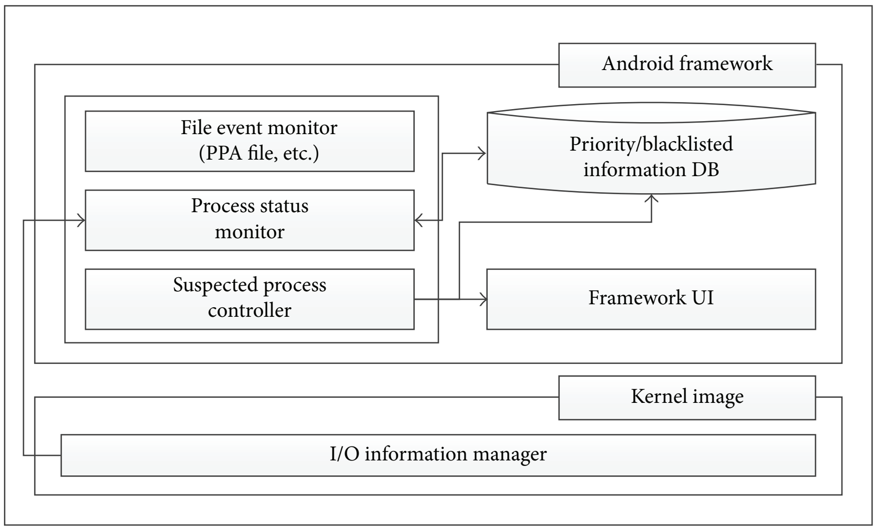

Android framework
File event monitor (PPA file, etc.)
Priority/blacklisted information DB
Process status monitor
Suspected process controller
Framework UI
Kernel image
I/O information manager

Figure 2: Proposed system on Android platform.

The proposed technique implements the configuration module, the monitoring module, and the processing module using the framework and kernel of the Android platform as shown in Figure 2. In addition, user's UI part added to the Android Settings and the database used in the configuration module are implemented within the framework. In the kernel, a part for generating I/O information to monitor the process is implemented, through which a kernel image is produced.

Algorithm 1 shows the operation flow of the basic technique proposed in this paper. Details are described later in different topics.

```
begin
    Input: process id P
    ProcessInfo ← ProcessDatabase(P);
    if ProcessInfo is blacklist then
        KillProcess(P);
        ProcessRemovalProcedure(P);
        return;
    else if ProcessInfo is not priority protection member then
        enqueueMonitoringProcessID(P);
    end
end
```

Algorithm 1: The main procedure of proposed technique.

3.1. Configuration Module. The configuration module is the basic setup to be applied when the proposed technique detects a ransomware. In this paper, default setting values that are information about the process or application installed by default on the Android platform are saved in a database. The foremost role of the configuration module is to specify the location of the files needed to be protected from the attacks of the ransomware. An area of these important files is called priority protection area (hereinafter PPA). If the proposed technique is run correctly, it will collect the information of PPA, register them to the watch list table for the monitoring module, and protect the corresponding files in real time. The second role is to register user's handling for the suspected process detected by the monitoring module into the database and maintain the handling. If the user finally determines the process as a ransomware, it stores the information of the corresponding process. It will automatically detect and delete the process depending on the user's feedback. If the user determines the process as normal, it records the information of the process and forces the system to maintain the process without terminating the process even if the process is redetected.

3.2. Monitoring Module. The monitoring module is responsible for detecting the ransomware by monitoring the PPA area and the process. The monitoring module is largely composed of two modules (file monitoring and process monitoring) based on the roles.

(i) File Monitoring Module. It continuously monitors the status of the input/output events such as reading, writing, copying, and deleting of a file belonging to a PPA set in the configuration module and detects the attacks of the ransomware. Algorithm 2 shows the operation flow of the file monitoring module proposed in this paper.

(ii) Process Monitoring Module. It continuously monitors Processor share by Process, Memory usage, I/O count, Storage I/O count, and so forth and detects the ransomware. Algorithm 3 shows the operation flow of the process monitoring module proposed in this paper.

Upon detecting the suspected process, it also handles malicious or exceptional processes in the database applied in the configuration module. For the process registered as a malicious process, the monitoring module will stop the process at the moment of detection and automatically delete the process. For the process registered as an exceptional process, it will allow the normal execution because it is specified as safe by the user.

3.2.1. File Event Monitoring. The monitoring module monitors the modification and deletion events of files and directories existing in a PPA. Monitoring path is generally through external storage of the device. Basic monitoring path is shown in Table 3.

Observer is arranged to monitor file events in each directory. File event monitoring using Observer is based on the patterns of ransomware to generate encrypted files after reading and writing target files and deleting original files. Observer can detect events of ransomware deleting and modifying files without obtaining data on the ransomware. Observer is responsible for monitoring modification and deletion events that occurred in each path while the device is on. If the event for the file occurs, Observer will pass the file event information to the monitoring module and find which process is the one that produced the event. The process found by the Observer is primarily checked through an exceptional handling process. If the process is not in a list, it is determined as a process suspicious of ransomware. The process will be stopped first. The technique inquires the user about subsequent handling of the process. Depending on the user's determination, the handling of the process in the database will be updated and managed.

```
Begin
  Input: process ids $P_n$
  While
    for all process id $P_i$ do
      Flag ← isOccuriedEventInPPA($P_i$);
      if Flag is enabled then
        KillProcess($P$);
        Result ← ProcessNotification($P_i$);
        if Result is block then
          ProcessRemovalProcedure($P_i$);
        end
        addProcessDatabase($P_i$);
      End
    end
  End
End
```

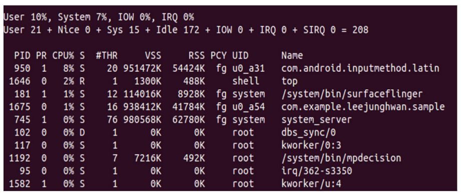
Algorithm 2: The file monitoring module of proposed technique.

```
begin
  Input: process ids $P_n$, Threshold $T$
  while
    for all process id $P_i$ do
      ProcessInfo ← getProcessInformation($P_i$);
      if ProcessInfo has occupied resources then $T$ then
        KillProcess($P$);
        Result ← ProcessNotification($P_i$);
        if Result is block then
          ProcessRemovalProcedure($P_i$);
        end
        addProcessDatabase($P_i$);
      end
    end
  end
end
```

Algorithm 3: The process monitoring module of proposed technique.

Table 3: Basic monitoring path summary.

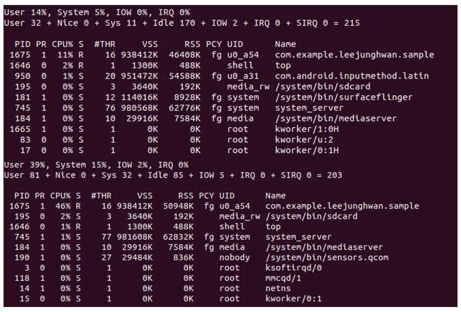
Location	Path
Android	/sdcard/Android/com
Picture	/sdcard/Pictures
DCIM	/sdcard/DCIM
Downloads	/sdcard/Downloads
Music	/sdcard/Music
Movies	/sdcard/Movies



```
User 10%, System 7%, IOW 0%, IRQ 0%
User 21 + Nice 0 + Sys 15 + Idle 172 + IOW 0 + IRQ 0 + SIRQ 0 = 208

  PID PR CPU% S  #THR     VSS     RSS PCY UID      Name
  950  1   8% S    20 951472K  54424K  fg u0_a31   com.android.inputmethod.latin
 1646  0   2% R     1   1300K    488K     shell    top
  181  1   1% S    12 114016K   8928K  fg system   /system/bin/surfaceflinger
 1675  0   1% S    16 938412K  41784K  fg u0_a54   com.example.leejunghwan.sample
  745  1   0% S    76 980568K  62780K  fg system   system_server
  102  0   0% D     1      0K      0K     root     dbs_sync/0
  117  0   0% S     1      0K      0K     root     kworker/0:3
 1192  0   0% S     7   7216K    492K     root     /system/bin/mpdecision
   95  0   0% S     1      0K      0K     root     irq/362-s3350
 1582  1   0% S     1      0K      0K     root     kworker/u:4
```

Figure 3: Process status of the process when a latent ransomware is carried out.



```
User 14%, System 5%, IOW 0%, IRQ 0%
User 32 + Nice 0 + Sys 11 + Idle 170 + IOW 2 + IRQ 0 + SIRQ 0 = 215

  PID PR CPU% S  #THR     VSS     RSS PCY UID      Name
 1675  1  11% R    16 938412K  46408K  fg u0_a54   com.example.leejunghwan.sample
 1646  0   2% R     1   1300K    488K     shell    top
  950  0   1% S    20 951472K  54588K  fg u0_a31   com.android.inputmethod.latin
  195  0   0% S     3   3640K    192K     media_rw /system/bin/sdcard
  181  1   0% S    12 114016K   8928K  fg system   /system/bin/surfaceflinger
  745  1   0% S    76 980568K  62776K  fg system   system_server
  184  1   0% S    10  29916K   7584K  fg media    /system/bin/mediaserver
 1665  1   0% S     1      0K      0K     root     kworker/1:0H
   83  0   0% S     1      0K      0K     root     kworker/u:2
   17  0   0% S     1      0K      0K     root     kworker/0:1H
User 39%, System 15%, IOW 2%, IRQ 0%
User 81 + Nice 0 + Sys 32 + Idle 85 + IOW 5 + IRQ 0 + SIRQ 0 = 203

  PID PR CPU% S  #THR     VSS     RSS PCY UID      Name
 1675  1  46% R    16 938412K  50948K  fg u0_a54   com.example.leejunghwan.sample
  195  0   2% S     3   3640K    192K     media_rw /system/bin/sdcard
 1646  0   1% R     1   1300K    488K     shell    top
  745  1   1% S    77 981608K  62832K  fg system   system_server
  184  1   0% S    10  29916K   7584K  fg media    /system/bin/mediaserver
  190  1   0% S    27  29484K    836K     nobody   /system/bin/sensors.qcom
    3  0   0% S     1      0K      0K     root     ksoftirqd/0
  118  1   0% S     1      0K      0K     root     mmcqd/1
   14  1   0% S     1      0K      0K     root     netns
   15  0   0% S     1      0K      0K     root     kworker/0:1
```

Figure 4: Process status when a latent ransomware performs active encrypting.

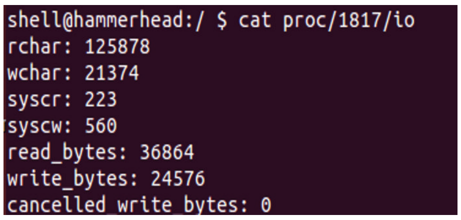
3.2.2. Process Monitoring. The monitoring module uses information such as Processor share for each Process, Memory usage, I/O count, and Storage I/O count to detect suspected process among running processes. It also detects suspected process through monitoring file events. The operation of Process monitoring is based on the information of malicious/exceptional processes stored in the database by the configuration module. It takes advantage of the fact that ransomware process uses a lot of system resources in the process of encrypting files in the storage. The proposed technique checks whether Processor share, Memory usage, I/O count, and Storage I/O count are more than a threshold value based on statistical methods. It will transmit the information of the corresponding process into the Processing module when it is higher than a threshold value.

To prove the change of Processor usage, a sample ransomware is run. The name of the corresponding ransomware process is called "com.example. ***.Sample". Figure 3 shows the status of the process when a latent ransomware process is carried out. Processor share shows 0-1% so that users will not notice it.

Figure 4 shows the situation when the latent ransomware performs encrypting files in earnest. The Processor usage is changed to 11% at the moment of the encryption. It peaked at 46%.

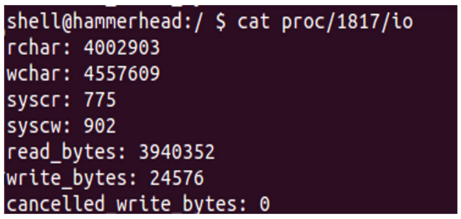
The process of I/O usage of the same ransomware at latency is shown in Figure 5. The top of the figure is the initial stage of the process execution. The amount of bytes read is relatively small.

The process of I/O usage of the same latent ransomware that performs active encrypting is shown in Figure 6. As shown in Figure 6, the file I/O usage is sharply increased because the data of the files to be encrypted are read and written in earnest. The sharp increase in the CPU usage and I/O usage can be used to detect the ransomware.



```
shell@hammerhead:/ $ cat proc/1817/io
rchar: 125878
wchar: 21374
syscr: 223
syscw: 560
read_bytes: 36864
write_bytes: 24576
cancelled_write_bytes: 0
```

Figure 5: I/O status when the ransomware is latent.



```
shell@hammerhead:/ $ cat proc/1817/io
rchar: 4002903
wchar: 4557609
syscr: 775
syscw: 902
read_bytes: 3940352
write_bytes: 24576
cancelled_write_bytes: 0
```

Figure 6: I/O status when the ransomware is active.

3.3. Processing Module. The processing module forcibly stops the process suspicious of ransomware in the monitoring module and inquires users about the appropriate handling of the process. Once the handling is determined, the information of the corresponding process will be stored in the database and used in the configuration module subsequently. Database table structure used in the processing module is shown in Table 4.

ID is used to place the number of each tuple. PackageName is the name of an application. RiskType is a flag to determine whether it is safe/unsafe. Comment is prepared in case a separate explanation is needed.

The processing module also warns users about the risk of the ransomware through Android permission analysis.

(i) System Permission. The ransomware has permissions of the system. It seizes permissions of the device's administrator and prevents users from manipulating the device. This permission involves the risk of ransomware browsing the user's personal files stored in the device without user's permission. It uses administrator's permission.

(ii) SMS Permission. While a normal application provides convenience to users with SMS permission, the ransomware intercepts received messages to use them for illegal purposes by using SMS permission.

(iii) Contract Permission. Permission to access contacts is stored in the device. Typical examples of making ill use of this permission are phishing and smishing.

(iv) Network Permission. Permission to automatically find network connected to the device and allow the ransomware to operate. Ransomware seizes permission of the device so that

Table 4: Database table structure used in the processing module.

Table name	Column name	Data type	NULL constraint	Primary key
IDSDB	ID	INTEGER	NOT NULL	PK
	PackageName	VARCHAR	NOT NULL	PK
	RiskType	Integer	NOT NULL	
	Comment	VARCHAR		

Table 5: Concerns of permission.

Permission	Concerns
GET_TASK	Rooting
WRITE_SETTINGS	Rooting
SYSTEM_ALERT_WINDOW	Rooting
RECEIVE_BOOT_COMPLETED	Rooting
READ_PHONE_STATE	Rooting
READ_EXTERNAL_STORAGE	Rooting, file accessing
WRITE_EXTERNAL_STORAGE	Rooting, file accessing
WAKE_LOCK	Rooting
READ_CONTACTS	Voice phishing, smishing, data spill
READ_CALL_LOG	Voice phishing, smishing, data spill
INTERNET	Rooting, cracking
ACCESS_NETWORK_STATE	Rooting, cracking
RECEIVE_SMS	Spying on sms, smishing, data spill
SEND_SMS	Spying on sms, Smishing, data spill
READ_SMS	Spying on sms, Smishing, data spill

users cannot operate the device. It has the risk of intercepting user's personal information stored in the device.

The processing module inquires of users about whether to keep or delete the corresponding program after stopping the process suspicious of ransomware. If the user shows his intention to delete the application, when the same process appears later, it is automatically removed without asking about user's thoughts because the user recognizes the corresponding application as ransomware. If he determines the process as normal, its safety is guaranteed so the process will not be forcibly stopped by the proposed technique. In addition, the proposed technique will let the user know if any part of the process is vulnerable. Concerns of permission are listed in Table 5.

3.4. User Interface. User Interface shown in Figure 7 provides users with easy access to the proposed method. UI is equipped with a basic format of the Android. It provides an interface of the configuration module. The proposed system functions can be turned on and off at any time using the corresponding interface. At the bottom, the names of the packages registered in the database so far can be checked. Addition, modification, and deletion of the information stored are also possible.

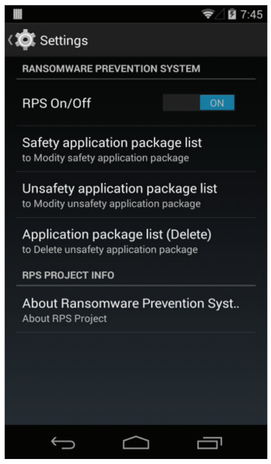

7:45
Settings
RANSOMWARE PREVENTION SYSTEM
RPS On/Off ON
Safety application package list
to Modity safety application package
Unsafety application package list
to Modity unsafety application package
Application package list (Delete)
to Delete unsafety application package
RPS PROJECT INFO
About Ransomware Prevention Syst..
About RPS Project

Figure 7: Implemented user interface.

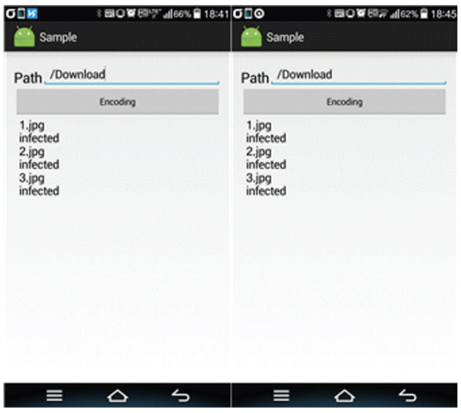
4. Evaluation

Unknown ransomware is used for evaluation of the proposed method compared to existing vaccine systems. A ransomware that encrypts files with 40-byte keys using the AES algorithm was made for testing. This sample ransomware has the function of opening all files on the input path and encrypting.

On the left of Figure 8 is the running result of a testing ransomware after running V3 Mobile one vaccine system that is famous in South Korea. On the right of Figure 8 is the result after running Avast made in the Czech Republic. These vaccine programs have no information about the new ransomware. They failed to detect the unknown ransomware. Therefore, files on /Download are encrypted. It is impossible to cure them either because there is no decryption key value.

In order to verify the proposed technique, PPA was set as /Download directory using the configuration module. Figure 9 shows the result of running the same sample ransomware after activating the proposed technique. In the device using the proposed method, users' files were protected because it found the ransomware before the encryption. Therefore, it stopped the ransomware process and asked about users' thoughts on deletion.

Results of evaluation of existing vaccine systems compared to the proposed technique are shown in Table 6. The proposed technique can deal with modified or new patterns of ransomware because it can detect ransomware using information such as Processor share, Memory usage or I/O count, and Storage I/O count. However, existing techniques need information of the ransomware to detect it. While traditional vaccines require updating the detection pattern



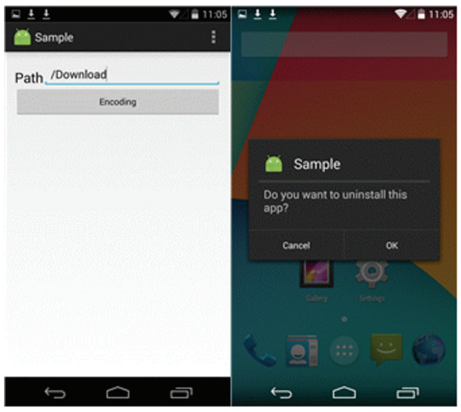
/Download

1.jpg
infected
2.jpg
infected
3.jpg
infected

/Download

1.jpg
infected
2.jpg
infected
3.jpg
infected

Figure 8: Running results of ransomware on existing vaccines (Avast, Ahnlab).



/Download

Do you want to uninstall this app?

Figure 9: Running results of ransomware using the proposed technique.

Table 6: Evaluation of existing vaccine systems compared to the proposed technique.

	Our method	V3	Avast
Protecting ransomware without ransomware info	O	X	X
Operating without updating	O	X	X
Operating without downloading	O	X	X
Operating without executing application	O	X	X

from time to time, the proposed method does not need so many updates because it can detect the ransomware based on its behavior. It does not need to install an application such as existing vaccines as it is implemented in the Android source. In this study, we found a slightly degraded performance of the device after using the proposed technique in order to protect sensitive information.

5. Conclusion

In this paper, a technique is proposed to reduce damage caused by unknown ransomware attacks on Android devices. The proposed method can effectively reduce damage caused by ransomware with modified or new patterns without obtaining information on the ransomware. It uses file input/output events and Processor status information based on the behavior of ransomware, unlike existing techniques that need information about the ransomware. It can automatically prevent damage caused by such ransomware attacks later based on information collected on the detected ransomware. It is possible to use the proposed method in all Android-based smart phones because this technique is added to the open source of Android source file. This technique is expected to allow users to minimize damage caused by attacks of ransomware that existing vaccine systems fail to detect.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Korea government (MSIP) (NRF-2015R1D1A1A01057680).

References

[1] X. Luo and Q. Liao, "Ransomware : a new cyber hijacking threat to enterprises," in *Handbook of Research on Information Security and Assurance*, IGI Global, 2009.

[2] "Worldwide Quarterly Mobile Phone Tracker," IDC, August 2015, http://www.idc.com/tracker/showproductinfo.jsp?prod_id=37.

[3] TREND Micro, *Ransomware Definition—Security Intelligence*, TREND Micro, Irving, Tex, USA, 2015, http://www.trendmicro.com/.

[4] D. Kim and S. Kim, "Design of quantification model for ransom ware prevent," *World Journal of Engineering and Technology*, vol. 3, no. 3, pp. 203–207, 2015.

[5] D. Lim, "Treats and countermeasures of malware," *Journal of IT Convergence Society for SMB*, vol. 5, no. 1, pp. 13–18, 2015.

[6] N. Andronio, S. Zanero, and F. Maggi, "HelDroid: dissecting and detecting mobile ransomware," in *Research in Attacks, Intrusions, and Defenses*, vol. 9404 of *Lecture Notes in Computer Science*, pp. 382–404, Springer, 2015.

[7] A. Beuhring and K. Salous, "Beyond blacklisting: cyberdefense in the era of advanced persistent threats," *IEEE Security & Privacy*, vol. 12, no. 5, pp. 90–93, 2014.

[8] P. Ducklin, "Reveton/FBI ransomware—exposed, explained and eliminated," NakedSecurity, August 2012, https://nakedsecurity.sophos.com/.

[9] J. Milletary, "Citadel Trojan Malware Analysis," Dell Secure Works Counter Threat Unit™ Intelligence Services, Dell Secure Works, September 2012.

[10] T. M. Marengereke and K. Sornalakshmi, "Cloud based security solution for android smartphones," in *Proceedings of the IEEE*

International Conference on Circuit, Power and Computing Technologies (ICCPCT '15), pp. 1–6, Nagercoil, India, March 2015.

[11] Y. Liu, Y. L. Sun, J. Ryoo, S. Rizvi, and A. V. Vasilakos, "A survey of security and privacy challenges in cloud computing: solutions and future directions," *Journal of Computing Science and Engineering*, vol. 9, no. 3, pp. 119–133, 2015.

[12] Ahnlab Security Issue, *How to Attack Us?, Ransomware 'CryptoLocker' That Hit South Korea*, 2015 (Korean), http://www.ahnlab.com/kr/site/securityinfo/secunews/secuNewsView.do?menu_dist=2&seq=23630.

[13] Ahnlab Security Report, "The latest mobile ransomware app and countermeasures," vol. 65, July 2015 (Korean), http://www.ahnlab.com/kr/site/securityinfo/asec/asecView.do?groupCode=VNI001&seq=23834.

[14] Ahnlab ASEC blog, "The ransomware that impersonate," NSB (National Security Bureau), Febuary 2015 (Korean), http://asec.ahnlab.com/1025.

[15] M. E. Wagner, *Behavior Oriented Detection of Malicious Code at Run-Time*, Florida Institute of Technology, 2004.

[16] P. Szor, *The Art of Computer Virus Research and Defense*, Symantec Press; Addison-Wesley Professional, 2005.

[17] J. Aycock, *Computer Viruses and Malware*, vol. 22, Springer Science & Business Media, 2006.

[18] D. Anderson, T. Frivold, and A. Valdes, "Next-generation intrusion detection expert system (NIDES): a summary," Tech. Rep. SRI-CSL-95-07, SRI International, Computer Science Laboratory, 1995.

[19] K. Daewon, "Automated Control Method and Apparatus of DDoS Attack Prevention Policy using The Status of CPU and Memory," Electronics and Telecommunications Research Institute(South Korea), US Patent, US 2012/0054823 A1, 2012.

[20] J. L. Lee and C. S. Hong, "Nonparametric detection methods against DDoS attack," *The Korean Journal of Applied Statistics*, vol. 26, no. 2, pp. 291–305, 2013.

[21] "System Permissions," API Guide, Android Developers, http://developer.android.com/intl/ko/guide/topics/security/permissions.html.

[22] virustotal, https://www.virustotal.com/en-gb/.

Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 3919134, 8 pages
http://dx.doi.org/10.1155/2016/3919134

Research Article

Learning-Based Detection of Harmful Data in Mobile Devices

Seok-Woo Jang[1] and Gye-Young Kim[2]

[1]*Department of Digital Media, Anyang University, 22 Samdeok-ro, 37 Beon-gil, Manan-gu, Anyang 430-714, Republic of Korea*
[2]*School of Software, Soongsil University, 369 Sangdo-ro, Dongjak-gu, Seoul 156-743, Republic of Korea*

Correspondence should be addressed to Gye-Young Kim; gykim11@ssu.ac.kr

Received 29 December 2015; Accepted 15 March 2016

Academic Editor: Seung Yang

The Internet has supported diverse types of multimedia content flowing freely on smart phones and tablet PCs based on its easy accessibility. However, multimedia content that can be emotionally harmful for children is also easily spread, causing many social problems. This paper proposes a method to assess the harmfulness of input images automatically based on an artificial neural network. The proposed method first detects human face areas based on the MCT features from the input images. Next, based on color characteristics, this study identifies human skin color areas along with the candidate areas of nipples, one of the human body parts representing harmfulness. Finally, the method removes nonnipple areas among the detected candidate areas using the artificial neural network. The experimental results show that the suggested neural network learning-based method can determine the harmfulness of various types of images more effectively by detecting nipple regions from input images robustly.

1. Introduction

Recently, digital media players with graphic user interface have rapidly developed along with high-speed wired and wireless communication technology, large-scale storage devices, and light and portable mobile devices. Because of this, diverse types of multimedia content such as photographs, animations, and high-definition videos are being spread freely [1–3]. Thanks to the remarkable growth of computing and networking technologies of mobile devices including smart phones or Internet-using tablets comparable to the existing typical personal computers, mobile device-based multimedia content has been widely used [4–7].

While anyone can obtain and replay multimedia data through Internet-connected high-speed wired and wireless mobile devices, pornographic videos, naked images, or other adult contents are also spread easily to adolescents and children, causing a huge social problem [8]. In this situation, the need for automatic assessment and filtering of adult images is increasing in the image security area, which inflow intentionally or unintentionally through various routes [9].

Recent literature on image processing and pattern recognition describes existing techniques assessing image harmfulness. Shih et al. searched an image of inquiry from adult and nonadult image database [10]. Among similar search results, if the number of adult images exceeds a certain level, the query image is assessed to be harmful. Zheng et al. used multi-Bayes classifier to identify skin areas and acquired the shape features of extracted skin areas [11]. Next these researchers applied the identified shape features to a boosted classifier to assess image harmfulness. Park et al. detected a breast, including the nipple area, using the Hough transformation in gray images to assess image harmfulness [12]. Lee et al. defined a human skin color model in advance and used the model to extract skin areas from input images [13]. Then, based on the shape features of the extracted skin areas, they assessed image harmfulness. In addition to these methods, new technologies are continuously tried to assess image harmfulness automatically [14].

These existing methods may ensure accuracy to some extent in some image databases. However, the methods' accuracy is not relatively high enough to take all of diversified types of images captured in different environments. In this situation, we propose a method to assess image harmfulness robustly by detecting human nipple areas by utilizing a hierarchical artificial neural network. In this paper, an image is viewed as being harmful if nipples of a naked woman are

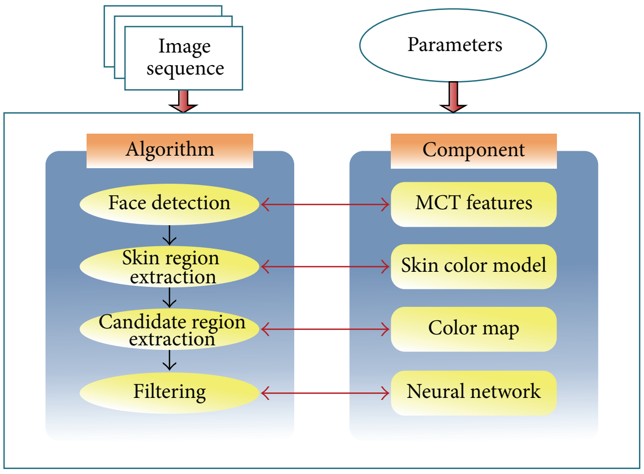
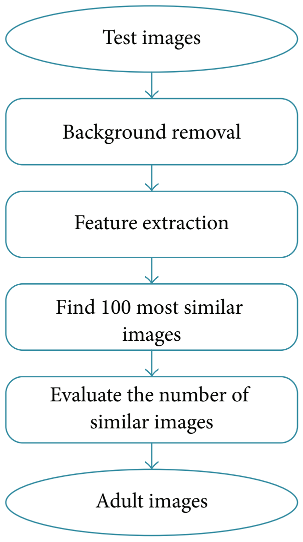


Figure 1: Overall flow of the proposed method.


Test images
Background removal
Feature extraction
Find 100 most similar images
Evaluate the number of similar images
Adult images
Figure 2: Flowchart of the retrieval-based method.

detected. Figure 1 shows the overall outline of harmfulness assessment algorithm proposed in this paper.

As shown in Figure 1, the proposed algorithm first detects a human face area from an input image through the modified census transform (MCT) method. Based on color features, the algorithm then identifies human skin color areas and extracts candidate areas of nipple, one of the human body elements. Lastly, using a hierarchical artificial neural network, the algorithm removes nonnipple areas from the extracted candidate nipple areas to remain the actual nipple areas and assess image harmfulness.

The rest of this paper is organized as follows. Section 2 describes existing studies on adult content detection in the image processing and multimedia area. Section 3 explains a method to extract facial elements from input images based on MCT features. Section 4 explains a technique to screen out candidate nipple areas from images by utilizing color information. Section 5 describes a technique to assess image harmfulness by verifying candidate nipple areas with a neural network. Section 6 shows the results of experiment conducted to compare and evaluate the performance of the proposed method. Section 7 presents the study's conclusions and presents future study expectations.

2. Related Work

With the development of mobile devices such as smart phones and Internet-using tablets in their computing performance and network function, adult multimedia content, including naked images or adult videos, is also freely distributed. In this situation, the need for technologies to automatically screen out such adult content is increasing. Relevant literature introduces methods related to automatic detection of adult content as follows.

The content-based image retrieval method [10] first removes the background area from images by utilizing the skin color distribution and acquires areas of interest in a square form. Next, this method extracts color, texture, and shape features from each image and searches the 100 most similar images to the input image from an image database consisting of adult and nonadult images. If the searched images include more than T_{ad} adult images, the given image is viewed as an adult image. If not, the given image is viewed as a nonadult image. T_{ad} is a predefined threshold value. In other words, the image retrieval-based method resolved the issue of adult image screening-out through image classification. Figure 2 shows the overall structure of the image retrieval-based method.

The shape feature-based method [11] utilizes the multi-Bayes classifier [15, 16] to detect areas with human skin color distribution more precisely. For precise skin detection, the human skin color detection procedures consist of two phases: skin pixel detection phase and skin area refinement phase. From the detected skin areas, shape features are extracted and input into the boosted classifier to assess if the input skin area is a naked image or not. In this paper, the shape features were analyzed by using the three simple shape descriptors of eccentricity, compactness, and rectangularity; seven normal moment invariants presented by Hu [17]; and Zernike moments [18, 19]. In this method, different boosted classifiers and shape features were utilized to compare and test the performance of adult image detection algorithm.

The new method based on breast area detection [12] uses the mean intensity filter and Hough transform [20, 21] to identify breast areas from images to screen out adult images. The proposed adult image identification method is comprised largely of the learning phase, recognition phase, and test phase. In the learning phase, by learning breast nipple part images, the system forms a nipple intensity filter to be used in the recognition phase. In the recognition phase, the input image is taken to extract the edges and connection elements are extracted by utilizing the edge density. Next, by considering the length to width ratio of the connection element, the system determines candidate nipple regions. The system measures the similarity between the learned nipple intensity filter and candidate nipple region in the input image to decide that with the highest similarity as the final candidate nipple area. The Hough transform is utilized to detect the breast line in the image. In the test phase, the locations of breast lines and candidate nipple areas learned in the recognition phase are considered to assess the final harmfulness of the corresponding image.

The human skin color model-based method [13] utilizes adaptive and extensible skin color distribution models capable of enduring the color cast caused by special lighting effects in the YC_bC_r space [22] to segment human skin color regions. Instead of using a predefined skin color model, this model gets human skin colors from the input images themselves to renew its model more adaptively. Then, multiple features are applied to assess the genuineness of segmented skin areas and whether the input image is an adult image or not. The texture smoothness is especially considered in the extracted skin color areas. For effective learning of skin color distribution, the K multilayer feedforward neural network [23, 24] is employed.

In addition to the methods described above, many other techniques regarding adult content detection have also been developed [14]. Although such methods may ensure some accuracy in some databases, they are not accurate enough to process every dynamic image taken in different environments.

3. Detection of Face Regions Using MCT

In this paper, to detect the facial area from input images, MCT features are used [25, 26]. MCT features are region-based features using (0, 1) binary information in the 3×3 kernel. In other words, in the 3×3 kernel, the mean is calculated and if the kernel value is larger than the mean, the value of 1 is assigned; if it is smaller than the mean, the value of 0 is assigned. Thus, in the 3×3 kernel, a total of $2^9 - 1$ or 511 MCT features can be produced. In general, MCT features utilize regional information less sensitive to lighting changes and ensure a simple calculation process. Therefore, they deliver a high detection rate and quick processing time in face detection of multimedia area.

Normally, the MCT features using the 3×3 kernel can be calculated using

$$\Gamma(x) = \bigotimes_{y \in n'} \zeta\left(\overline{I(x)}, I(y)\right). \tag{1}$$

In (1), $I(x)$ represents the brightness of x and $\overline{I(x)}$ is the average brightness of pixels in the kernel. n' represents the center of kernel and adjacent pixels. $\zeta()$ is a comparison function. If $I(y)$ is larger than the average brightness, $\zeta()$ has a value of 1; if not, $\zeta()$ has a value of 0. $\otimes$ is a decimal conversion operator. It changes the 9-digit binary number resulting from $\zeta()$ into a decimal number. Thus, the MCT features used in this paper range from 0 to 510. Figure 3 shows an example of MCT transformation.

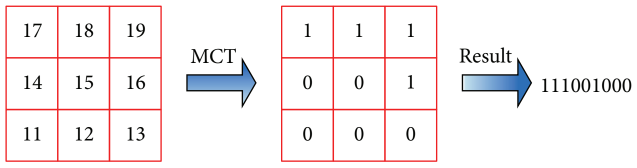


Figure 3: Example of MCT transform.

The MCT features identified in the input image are applied to a face detection classifier generated by Adaboost learner [27] to primarily screen out a facial area. Within the detected facial area, the eye and lip areas are then extracted with EyeMap and LipMap [28, 29]. The EyeMap-based method is a method to use YC_bC_r color space. It uses the Y channel-based EyeMapL and C_bC_r channel-based EyeMapC to perform the AND calculation and produce EyeMap. The LipMap-based method is to estimate the lip area colors based on the colors of overall skin area. It calculates η, variable of overall skin area color, and uses this η to generate LipMap.

Figure 4 shows the results of eye and lip area detection by applying the EypMap and LipMap to the face areas that is detected using the MCT features. Figure 4(a) is an example of eye area detection, and Figure 4(b) is an example of lip area detection.

The minimum enclosing rectangle including the detected eye and lip areas is selected as the final face region. In this paper, the face area, including extracted eye and lip regions, is used to effectively filter out candidate nipple areas to be extracted in the subsequent phase. In other words, because candidate nipple areas cannot exist inside the human face area, if such an area exists in the face region, the system regards it as a nonnipple area and removes it.

4. Extraction of Nipple Regions

In this paper, to assess the harmfulness of diverse input images, the existence of female nipple area in such images is determined. To this end, human skin areas are first detected using a predefined oval-shaped human skin color distribution model [28].

Then, as in (2), candidate nipple areas are extracted from skin regions by using the nipple map defined by the YC_bC_r color model. In (2), $Y(x, y)$, $C_b(x, y)$, and $C_r(x, y)$ represent Y, C_b, and C_r color values in the corresponding (x, y) coordinates. The nipple map is defined by utilizing all of the color elements of the YC_bC_r color model. Each item value of (2) is normalized between 0 and 255:

$$\text{Nipple}(x, y) = C_r^2(x, y) \times \frac{C_r(x, y)}{C_b(x, y)} + \frac{C_r(x, y)}{Y(x, y)} \times \{255 - Y(x, y)\}. \tag{2}$$

Normally, human nipple areas have reddish color values and relatively darker values in terms of brightness. Based on this fact, the definition of nipple map, Nipple(x, y), was made. In (2), $C_r(x, y)/Y(x, y)$ emphasizes the pixels with reddish color and lower brightness. $C_r(x, y)/C_b(x, y)$ stresses the nipple area more relatively to the skin area.

Table 1 shows the quantitative comparison of average and standard deviation of YC_bC_r color distribution on skin area

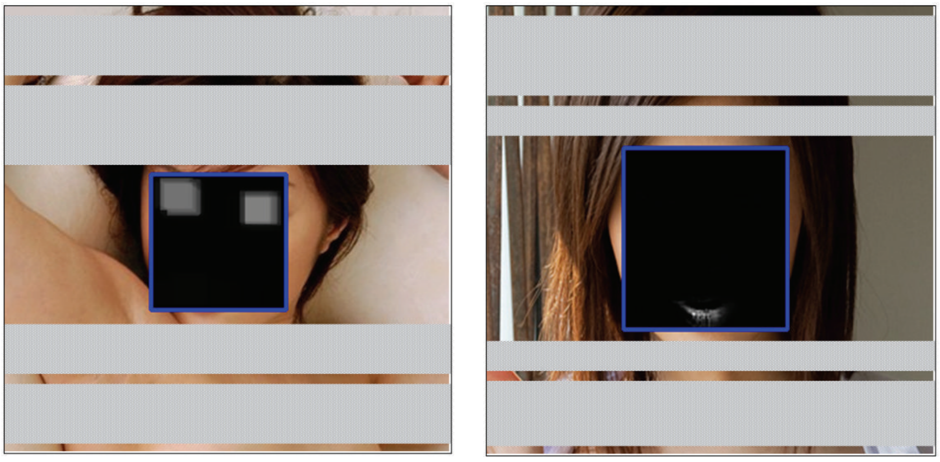

(a) Eye region detection (b) Lip region detection

FIGURE 4: Detection of eyes and lip regions.

TABLE 1: Comparison of color distribution of skin and nipple regions.

Color	Region			
	Nipple		Skin	
	Mean	Variance$^{1/2}$	Mean	Variance$^{1/2}$
Y	116	6.84	187	12.40
C_b	119	2.96	103	2.21
C_r	140	3.87	156	6.65

and nipple area. As shown in Table 1, the skin area, compared with the nipple area, has lower C_b values on average and higher C_r values.

In this paper, the nipple map images extracted by applying the nipple map to skin area are marked as brighter if their likelihood to be a nipple area is higher. The lower the chance of being a nipple area, the darker they are marked. Next, the extracted nipple map images are binarized and labeled to extract candidate nipple areas. The Otsu method is used to binarize nipple map images [30, 31]. This method is known to deliver the best performance when a brightness histogram has two types of probability density functions. The Otsu binarization method finds the best critical value for the brightness histogram binarization based on statistics without preliminary knowledge. It is one of the most frequently utilized binarization algorithms in image processing and computer vision fields.

After detecting candidate nipple areas by binarizing nipple map images, morphological operation is applied to remove relatively smaller areas, such as noise areas [32, 33]. Morphological image processing is a collection of nonlinear operations related to the shape or morphology of features in an image. Morphological operations rely only on the relative ordering of pixel values, not on their numerical values, and therefore are especially suited to the processing of binary images. In general, opening morphological operation conducts a dilation process after an erosion process. It removes areas smaller than a certain size and leaves other regions in similar sizes to their original sizes. The boundaries of the remaining areas become softer.

5. Determination of Harmfulness Using Neural Network

After the candidate nipple areas are extracted in the previous phase, the MCT features and artificial neural network are utilized to remove nonnipple areas and have only actual nipple regions.

In other words, this paper builds a learning database with nipple images normalized in the 50 × 50 pixel size to learn nipple areas. Next, the MCT features are extracted from each nipple area and the set of extracted MCT feature is learned through the artificial neural network to generate a nipple classifier. Although MCT is much sensitive to rotation, such a restriction could be minimized as nipples are circular. Lastly, by using the generated nipple area learning classifier, candidate nipple areas are verified to make a final assessment on image harmfulness. That is, as in (3), if a female nipple area is viewed to be exposed, the system regards the input image harmful; if not, it views the input image as being not harmful:

$$\begin{aligned} &\text{IF} \quad R^{i}_{\text{nipple}} \in I_t(x,y), \\ &\text{THEN} \quad I_t(x,y) \text{ is harmful,} \\ &\text{ELSE} \quad I_t(x,y) \text{ is not harmful.} \end{aligned} \tag{3}$$

In (3), R^{i}_{nipple} represents the ith nipple area included in an image and $I_t(x, y)$ means an input image at the time t.

This paper proposes an algorithm recognizing nipple areas after learning them by using a layered hierarchical artificial neural network. The learning function of the artificial neural network is the error backpropagation algorithm [34]. One hidden layer is also used. The activation function is the binary sigmoid function. The hierarchical artificial network

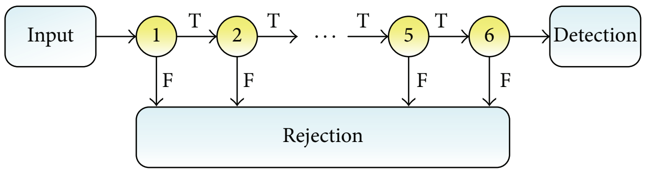


FIGURE 5: Structure of neural network.

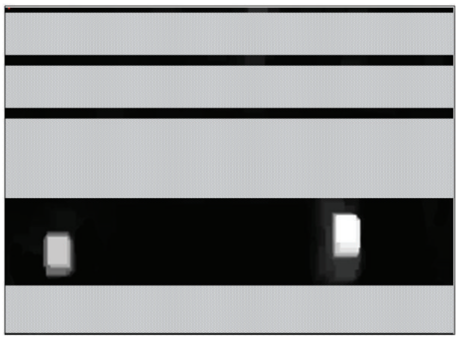
consists of 511 input nodes, 128 hidden nodes, and 1 output node.

The hierarchical artificial neural network used in this paper inputs and relearns detected samples using a threshold value that produces 99% detection rate and 50% misdetection rate. It repeats this process in 6 layers. Candidate nipple areas are normalized into 50 × 50 in this paper and they are tested by using the hierarchical artificial neural network. Figure 5 overviews the classification process of hierarchical artificial neural network.

The candidate nipple areas extracted in this process may include multiple nipple areas. Thus, if the size of candidate nipple areas defined as in (4) is relatively larger than the size of overall images, our experiment tries nipple detection again within the candidate regions to divide the corresponding candidate regions smaller:

$$R_{\text{size}}^{i} = \frac{\text{MER}_W(R_i) \times \text{MER}_H(R_i)}{I_W \times I_H}. \quad (4)$$

(a) Input image



(b) Nipple map

FIGURE 6: Generation of nipple map.

6. Experimental Results

The computer used in this paper has an Intel Core™ i7 2.93 Ghz CPU and 8 GB memory. The operating system is Microsoft Window 7. The programming tool to realize the proposed harmfulness detection method is Microsoft Visual C++ and OpenCV. To compare and evaluate the performance of the proposed algorithm, diverse types of adult images and nonadult images were collected, which had been captured in normal outdoor and indoor environments without a specific constraint.

Figure 6(a) shows an adult image and Figure 6(b) shows the result of nipple map estimation from the input image.

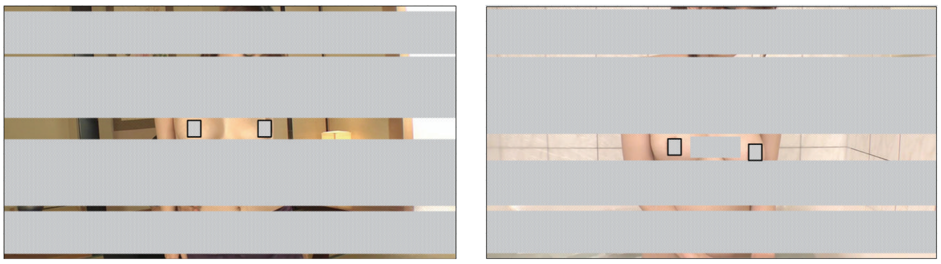
Figures 7(a) and 7(b) display the examples of final nipple area detection based on the proposed method.

To qualitatively evaluate the performance of the proposed image harmfulness assessment method, an accuracy criterion was employed, which is defined in (5). N_{TP} used in these equations represents the number of accurately detected nipple areas. N_{FP} means the number of misdetected nipple areas that are actually nonnipple areas. N_{FN} is the number of nipple areas that are not detected. $R_{\text{precision}}$ represents the relative ratio of nipple areas accurately detected in all of the detected nipple areas from an input image. R_{recall} represents the relative ratio of accurately detected nipple areas in the entire nipple areas that actually exist in a given image. In pattern recognition and information retrieval with binary classification, precision (also called positive predictive value) is the fraction of retrieved instances that are relevant, while recall (also known as sensitivity) is the fraction of relevant instances that are retrieved. Both precision and recall are therefore based on an understanding and measure of relevance [35]:

$$R_{\text{precision}} = \frac{N_{\text{TP}}}{N_{\text{TP}} + N_{\text{FP}}}, \qquad R_{\text{recall}} = \frac{N_{\text{TP}}}{N_{\text{TP}} + N_{\text{FN}}}. \quad (5)$$

In this paper, the proposed method was compared with the existing intensity filter-based method and geometric filtering-based method to test its accuracy. Figures 8 and 9 display the accuracy test results of the proposed algorithm through (5). As in Figures 8 and 9, the hierarchical artificial neural network-based method was found to reduce the misdetection rate, providing higher accuracy in image harmfulness assessment.

Of the three methods, the intensity filter-based method showed the lowest accuracy. Because color information was not sufficiently utilized, the method had many errors. The geometric filtering-based method detects candidate nipple areas first and then removes nonnipple areas using key geometric features. This method possibly causes misdetection because more specific filtering is difficult. The proposed method employs the hierarchical artificial neural network to learn the main features of human nipple areas in depth before detection, realizing a higher accuracy. However, if main parameters used in the suggested algorithm are not tuned sufficiently in the initialization procedure, the accuracy



(a) Nipple detection 1 (b) Nipple detection 2

Figure 7: Nipple detection.

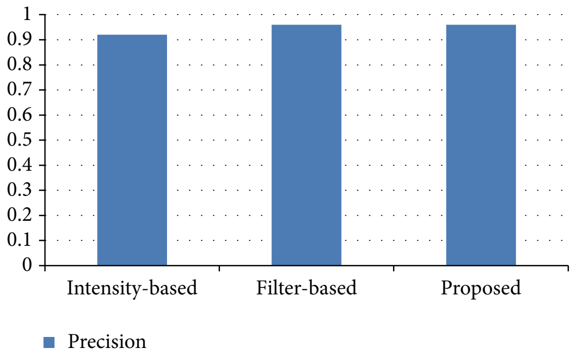


Figure 8: Precision rates.

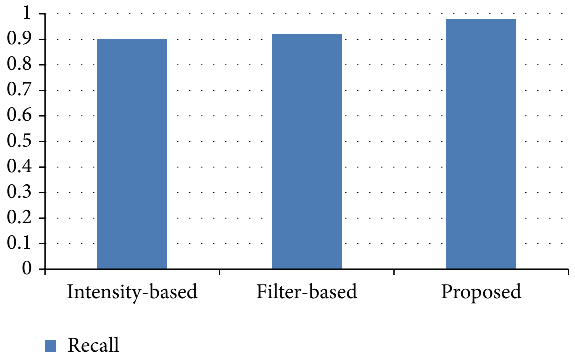


Figure 9: Recall rates.

of image harmfulness detection may become somewhat low. In addition, when nipple areas are contained in sections of the input image where the picture quality is decreased, the proposed method may indicate a decrease in the precision rate of image harmfulness detection.

7. Conclusions

This paper proposed a new method of automatically assessing the harmfulness of input images based on an artificial neural network. The proposed method first detects a human face area in the input image by using the MCT features. Next, based on the color features, the method obtains human skin areas and extracts candidate nipple areas. Lastly, among the candidate nipple areas, nonnipple areas are removed using the hierarchical artificial neural network and actual nipple areas are robustly detected to finally assess image harmfulness.

In the experiment, the proposed algorithm was applied to diverse types of adult and nonadult images captured in usual indoor and outdoor environments without a specific constraint to test its performance. As a result, the proposed method using the hierarchical artificial neural network was found to provide more robust detection performance than other existing methods. In other words, the proposed method showed a higher accuracy as it first learns the features of nipple areas through the neural network and then detects nipple areas.

Future work of this study will include determining the harmfulness of various types of input images more effectively by dividing the harmfulness of the images into multiple levels instead of the present two statuses, harmful and not harmful. We will also attempt to tune the predefined parameters used in the suggested algorithm adaptively for improving the stability of the overall system. In addition, in assessing the image harmfulness, how to consider other human body parts representing harmfulness other than the nipple area, such as navels, hips, and genital regions, will also explored continuously.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science, and Technology (2011-0021984).

References

[1] G.-T. Zhou, K. M. Ting, F. T. Liu, and Y. Yin, “Relevance feature mapping for content-based multimedia information retrieval,” *Pattern Recognition*, vol. 45, no. 4, pp. 1707–1720, 2012.

[2] A. Mostefaoui, H. Noura, and Z. Fawaz, "An integrated multimedia data reduction and content confidentiality approach for limited networked devices," *Ad Hoc Networks*, vol. 32, pp. 81–97, 2015.

[3] F. Amato, F. Colace, L. Greco, V. Moscato, and A. Picariello, "Semantic processing of multimedia data for e-government applications," *Journal of Visual Languages and Computing*, vol. 32, pp. 35–41, 2016.

[4] R. Sanchez-Iborra, M.-D. Cano, J. J. P. C. Rodrigues, and J. Garcia-Haro, "An experimental QoE performance study for the efficient transmission of high demanding traffic over an Ad Hoc network using BATMAN," *Mobile Information Systems*, vol. 2015, Article ID 217106, 14 pages, 2015.

[5] Y.-S. Lee and S.-B. Cho, "A mobile picture tagging system using tree-structured layered Bayesian networks," *Mobile Information Systems*, vol. 9, no. 3, pp. 209–224, 2013.

[6] F. M. Borrego-Jaraba, I. L. Ruiz, and M. Á. Gómez-Nieto, "An ubiquitous and non-intrusive system for pervasive advertising using NFC and geo-location Technologies and Air Hand Gestures," *Mobile Information Systems*, vol. 10, no. 4, pp. 361–384, 2014.

[7] P. M. P. Rosa, J. J. P. C. Rodrigues, and F. Basso, "A weight-aware recommendation algorithm for mobile multimedia systems," *Mobile Information Systems*, vol. 9, no. 2, pp. 139–155, 2013.

[8] V. M. T. Ochoa, S. Y. Yayilgan, and F. A. Cheikh, "Adult video content detection using machine learning techniques," in *Proceedings of the 8th International Conference on Signal Image Technology and Internet Based Systems (SITIS '12)*, pp. 967–974, IEEE, Naples, Italy, November 2012.

[9] S.-W. Jang, Y.-J. Park, G.-Y. Kim, H.-I. Choi, and M.-C. Hong, "An adult image identification system based on robust skin segmentation," *Journal of Imaging Science and Technology*, vol. 55, no. 2, 2011.

[10] J.-L. Shih, C.-H. Lee, and C.-S. Yang, "An adult image identification system employing image retrieval technique," *Pattern Recognition Letters*, vol. 28, no. 16, pp. 2367–2374, 2007.

[11] Q.-F. Zheng, W. Zeng, G. Wen, and W.-Q. Wang, "Shape-based adult images detection," in *Proceedings of the 3rd International Conference on Image and Graphics (ICIG '04)*, pp. 150–153, Hong Kong, December 2004.

[12] Y.-J. Park, S.-H. Weon, J.-K. Sung, H.-I. Choi, and G.-Y. Kim, "Identification of adult images through detection of the breast contour and nipple," *Information*, vol. 15, no. 7, pp. 2643–2651, 2012.

[13] J.-S. Lee, Y.-M. Kuo, P.-C. Chung, and E.-L. Chen, "Naked image detection based on adaptive and extensible skin color model," *Pattern Recognition*, vol. 40, no. 8, pp. 2261–2270, 2007.

[14] M.-J. Tsai and H.-S. Chang, "The design of a hybrid feature detector for adult images," in *Proceedings of the 7th International Conference on Semantic Computing (ICSC '13)*, pp. 389–390, IEEE, Irvine, Calif, USA, September 2013.

[15] G. Feng, J. Guo, B.-Y. Jing, and T. Sun, "Feature subset selection using naive Bayes for text classification," *Pattern Recognition Letters*, vol. 65, article 6297, pp. 109–115, 2015.

[16] H.-C. Lin and C.-T. Su, "A selective Bayes classifier with meta-heuristics for incomplete data," *Neurocomputing*, vol. 106, pp. 95–102, 2013.

[17] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE Transactions on Information Theory*, vol. 8, no. 2, pp. 179–187, 1962.

[18] C.-W. Tan and A. Kumar, "Accurate iris recognition at a distance using stabilized iris encoding and Zernike moments phase features," *IEEE Transactions on Image Processing*, vol. 23, no. 9, pp. 3962–3974, 2014.

[19] M. Liang, J. Du, S. Cao, and L. Li, "Super-resolution reconstruction based on multisource bidirectional similarity and non-local similarity matching," *IET Image Processing*, vol. 9, no. 11, pp. 931–942, 2015.

[20] H. Chen, Y. Lin, and B. Chen, "Co-segmentation guided hough transform for robust feature matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 12, pp. 2388–2401, 2015.

[21] K.-Y. Guo, E. G. Hoare, D. Jasteh, X.-Q. Sheng, and M. Gashinova, "Road edge recognition using the stripe Hough transform from millimeter-wave radar images," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 825–833, 2015.

[22] K. B. Shaik, P. Ganesan, V. Kalist, B. S. Sathish, and J. M. M. Jenitha, "Comparative study of skin color detection and segmentation in HSV and YCbCr color space," *Procedia Computer Science*, vol. 57, pp. 41–48, 2015.

[23] X.-B. Liang and J. Wang, "A recurrent neural network for nonlinear optimization with a continuously differentiable objective function and bound constraints," *IEEE Transactions on Neural Networks*, vol. 11, no. 6, pp. 1251–1262, 2000.

[24] S. Qin and X. Xue, "A two-layer recurrent neural network for nonsmooth convex optimization problems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 6, pp. 1149–1160, 2015.

[25] B. Froba and A. Ernst, "Face detection with the modified census transform," in *Proceedings of the 6th IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 91–96, Seoul, South Korea, May 2004.

[26] L. Bai, J. Liang, C. Dang, and F. Cao, "A novel attribute weighting algorithm for clustering high-dimensional categorical data," *Pattern Recognition*, vol. 44, no. 12, pp. 2843–2861, 2011.

[27] K.-K. Kong and K.-S. Hong, "Design of coupled strong classifiers in AdaBoost framework and its application to pedestrian detection," *Pattern Recognition Letters*, vol. 68, part 1, pp. 63–69, 2015.

[28] R.-L. Hsu, M. Abdel-Mottaleb, and A. K. Jain, "Face detection in color images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 696–706, 2002.

[29] K.-M. Lee, "Component-based face detection and verification," *Pattern Recognition Letters*, vol. 29, no. 3, pp. 200–214, 2008.

[30] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[31] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 146–168, 2004.

[32] J. E. Arco, J. M. Górriz, J. Ramírez, I. Álvarez, and C. G. Puntonet, "Digital image analysis for automatic enumeration of malaria parasites using morphological operations," *Expert Systems with Applications*, vol. 42, no. 6, pp. 3041–3047, 2015.

[33] R. Su, C. Sun, C. Zhang, and T. D. Pham, "A new method for linear feature and junction enhancement in 2D images based on morphological operation, oriented anisotropic Gaussian function and Hessian information," *Pattern Recognition*, vol. 47, no. 10, pp. 3193–3208, 2014.

[34] Y.-H. Lee, D.-H. Kim, and H.-S. Ko, “License plate detection with improved adaboost learning based on Newton’s optimization and MCT,” *Journal of the Korea Society of Computer and Information*, vol. 17, no. 12, pp. 71–82, 2012.

[35] X. Zhang, X. Feng, P. Xiao, G. He, and L. Zhu, “Segmentation quality evaluation using region-based precision and recall measures for remote sensing images,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 102, pp. 73–84, 2015.

Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 8019830, 9 pages
http://dx.doi.org/10.1155/2016/8019830

Research Article

Enhancing the Security of Personal Identification Numbers with Three-Dimensional Displays

Mun-Kyu Lee,[1] Jin Bok Kim,[2] and Matthew K. Franklin[3]

[1]*Department of Computer and Information Engineering, Inha University, Incheon 402-751, Republic of Korea*
[2]*Kakao Corporation, Jeju 63309, Republic of Korea*
[3]*Department of Computer Science, UC Davis, CA 95616, USA*

Correspondence should be addressed to Mun-Kyu Lee; mklee@inha.ac.kr

Received 28 October 2015; Accepted 20 March 2016

Academic Editor: Ching-Hsien Hsu

Passwords and personal identification numbers (PINs) are convenient and ubiquitous, but they are quite vulnerable to attackers who stand near the user ("shoulder-surfers"). This problem may be partially resolved by changing the user interface, but previous solutions of this kind still give shoulder-surfing attackers a significant advantage over brute force search. This paper provides a novel solution based on three dimensions, particularly suitable for glasses-free three-dimensional (3D) displays found in many smartphones and handheld game consoles. A user at the "3D spot" may log in easily, while nearby shoulder-surfers gain no advantage. A detailed experimental usability analysis is performed to demonstrate the effectiveness of the proposed scheme in comparison to the existing methods.

1. Introduction

User authentication is a procedure that enables a user of a system to prove his or her identity and to access the system. Although there are various approaches to user authentication [1, 2], the practical reality is that authentication based on the user's memory is still used in a majority of cases [3]. For example, passwords are frequently used to access desktop personal computers, terminals, and websites, and personal identification numbers (PINs) are used to withdraw cash from an automated teller machine (ATM), unlock a mobile device, and even open a door. While there have been various efforts to improve the security of passwords and PINs by helping users choose good ones and keep them secret; for example, [3–5], the essential problem of passwords and PINs is that they are vulnerable to shoulder-surfing attacks. In other words, anyone who observes the log-on procedure can easily memorize the password after looking over the user's shoulder [6].

One direction of research in the literature to solve this problem is to revise the interface to input a password or PIN [6]. That is, instead of entering directly the secret itself, the user is given randomized challenges and is asked to input appropriate responses that are computed using the password or PIN. The challenge-response task should be designed in an asymmetric manner so that the user may easily compute the responses, while the observer may not obtain useful information on the secret by observing a session. For example, the binary PIN-entry method [6] modifies the regular PIN pad and displays digits with black and white background colors as a challenge. The user recognizes the color of the current PIN digit and touches one of the two keys, "Black" and "White." To uniquely determine a PIN digit, this task is iterated four times, requiring 16 stages in total to input a 4-digit PIN. Figure 1 shows an example wherein "1" is being inputted via this method over a smartphone.

However, the modification of the input interface does not solve the problem completely. Because the attacker observes the challenge-response pairs, s/he may get partial information on the secret, if not the whole PIN. Moreover, if the attacker is helped by additional material such as a recording device, the attack will be much easier. For example, an

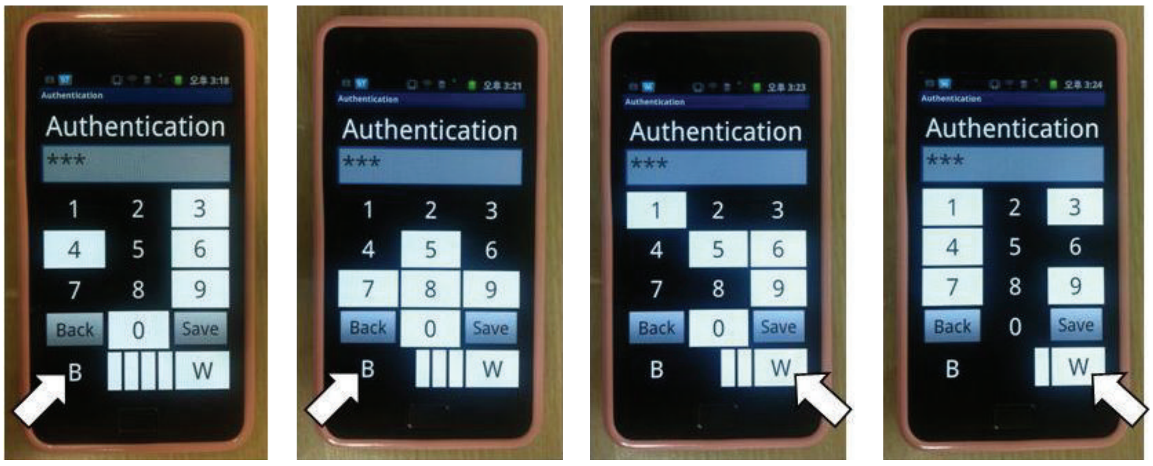


Figure 1: Example round to input "1" in the binary PIN-entry method [6] (reproduction of Figure 1 in [7]). The user enters "Black," "Black," "White," and "White" in sequence. (a) Stage 1. (b) Stage 2. (c) Stage 3. (d) Stage 4.

attacker against the above binary method [6] can determine the PIN uniquely if s/he can record all challenge-response pairs.

Therefore, a more promising solution is to physically prevent an attacker from observing an input session [7]. This may be realized using secure secondary channels such as an audio channel [9, 10] and a vibration channel [9–13] that are not accessible by the attacker. In these methods, the challenge transmitted through secondary channels is combined with the information shown over the open visual channel. That is, they are multimodal methods. However, it is known that unimodal performance is much better than multimodal performance [14], and the visual channel is still the most effective data transmission channel because about 80% of the information that we obtain comes through our eyes [15].

Then, a natural question would be "can we construct a *secure* visual channel?" In this paper, an affirmative answer to this question is given, and a secure PIN-entry method using a three-dimensional (3D) display is presented. In the proposed method called *3DPIN*, the random challenge is displayed to the user as a stereoscopic image. An attacker who does not have access to the 3D channel obtains no information on the challenge. The proposed method is particularly useful for glasses-free 3D displays such as parallax barrier displays [16] used in various smartphones and handheld game consoles. The 3D challenge is only visible to the legitimate user who is at a specific spot in front of the display, which we call the *3D spot*, while the attacker who is located in any other spot does not recognize the 3D effect. In a setting with 3D glasses such as a shutter system or a polarization system, the legitimate user who wears the 3D glasses can recognize the 3D challenge, but any person who does not wear the glasses cannot.

A part of this paper was presented at GCCE 2014 [8]. In this extended version, more technical details for the design of 3DPIN and realization of 3D effects are provided. In addition, the results of enhanced performance analyses are explained. Finally, the security against a brute force attack and a shoulder-surfing attack is analyzed in comparison to the existing methods.

2. 3DPIN

Figure 2 shows the layout of the proposed method, 3DPIN. Although the proposed method was implemented and tested over a smartphone that employs the parallax barrier display for 3D effects, we modified the screenshots so that the readers may understand the concept by using easily accessible anaglyph glasses with red (left eye) and cyan (right eye) filters. That is, each digit was modified to have two components with red and cyan colors. If the reader looks at the figures through the anaglyph glasses, it gives similar effects to the situation where the reader is at the 3D spot in front of the smartphone. For example, "8" has a different depth from the other digits, as shown in Figure 2(b), because the red and cyan components are in the opposite direction of each other as compared to the other digits. However, it should be noted that this does not imply that the reader's view without the anaglyph glasses is equivalent to the attacker's view who is at a wrong position with respect to the smartphone. Because the red and cyan figures are displayed in black in the real smartphone implementation, as shown in Figure 4, a prominent digit and a depressed digit look the same in two-dimensional (2D) vision. That is, the attacker who is not at the 3D spot sees either ten digits with the same depth or ten blurred digits.

The basic principle of the proposed method is similar to that of a traditional safe with a dial lock. That is, the user is required to enter a PIN digit by rotating it and aligning it with an indicator symbol. However, a few novel techniques were used in the design of the proposed method as follows:

(i) Phantom indicator: in the proposed method, there is no explicit indicator such as an arrow or a marker on a dial lock. It is sufficient that one random digit

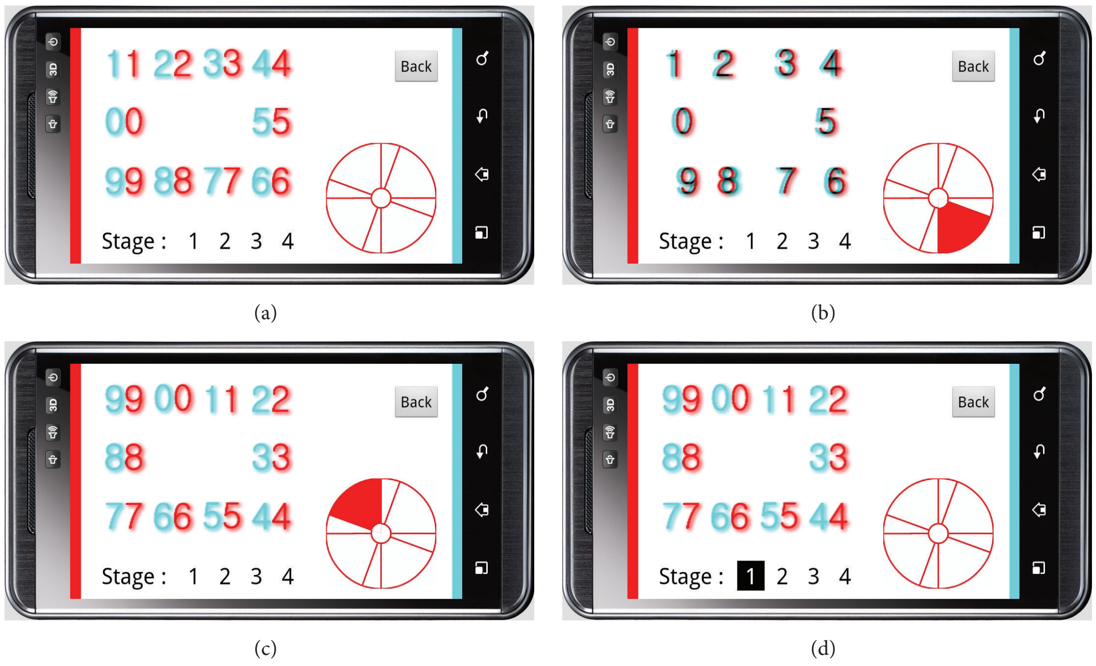


Figure 2: Example procedure for one digit entry using the proposed method (reproduction of Figure 1 in [8]). (a) Initial state. (b) Challenge transmission. (c) Commitment of response. (d) Next digit.

is displayed with a different depth from those of the others. For example, in a challenge given in Figure 2(b), the position of "8" is remembered as a phantom indicator. For the sake of convenience, we will call "8" an indicator digit in this case. The user's task is to align the target PIN digit with the phantom indicator by rotating the digit array. For maximum security, we designed the method so that the depth is displayed for a minimum time and all digits become located in the same layer (as shown in Figure 2(c)) right after the user starts rotation.

(ii) Rectangular arrangement: the digits are arranged as a 3 × 4 array as shown in Figure 2(a), not as a circle, because we found out from a pilot test that the users better recognize the depth of each digit when the digits are arranged in a rectangular form than when they are in a circular or diamond arrangement.

(iii) Position perturbation: when the digits are displayed in three dimensions, as shown in Figure 2(b), the horizontal position of each digit is slightly perturbed for security reasons. The principle of 3D displays is to show different images to the left and right eyes. Figures 3(a) and 3(b) are the simulated images shown to the left and right eyes, respectively, when there is no position randomization. When the user who is at the 3D spot sees the image shown in Figure 3(c), the attacker who is not at the right position may see the image in either Figure 3(a) or Figure 3(b). Whereas the distances between the adjacent digits in Figure 3(c) are the same, they are different in Figures 3(a) and 3(b). That is, "9," which is a prominent digit in the 3D image, has a relatively shifted position in the separated image for each eye. This may reveal the indicator to the attacker. We solve this problem by slightly moving each digit horizontally by a random amount.

(iv) Indirect touch: in order to rotate the digit array, the user does not directly touch the digits but uses the scroll wheel displayed at the right bottom corner of the touch screen. This indirect interface enhances security, because the users tend to directly touch the correct PIN digit unconsciously if they are allowed to touch the rectangular dial.

As a result of adopting the above four techniques, the proposed method works as follows: initially, the rectangular array is displayed as shown in Figure 2(a). All digits are displayed at the same depth. At the moment the user puts his/her finger on the scroll wheel, the touched region in the wheel changes red and the digits change their depth. The phantom indicator digit, for example, "8" in Figure 2(b), is displayed as a prominent object and the other digits are displayed as depressed objects, or vice versa. After the user recognizes the phantom indicator, s/he scrolls the wheel to rotate the array if needed. For example, if the user wants to input "6," s/he rotates the array by rotating the scroll wheel by two positions clockwise, moving "6" to the position where "8" was located initially. At the moment the user starts rotation, the difference in 3D depth of the digits disappears for higher security. After rotation by an appropriate amount as shown in Figure 2(c), the user releases the finger from the display, which confirms that the user's choice is "6." Then, another rectangular array is displayed for the second digit as shown

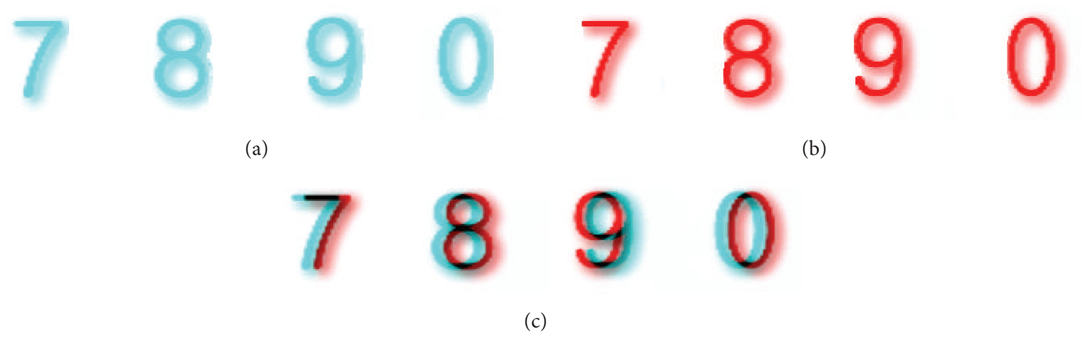


Figure 3: Digits with no position randomization. (a) Image for left eye. (b) Image for right eye. (c) Overlapped image.

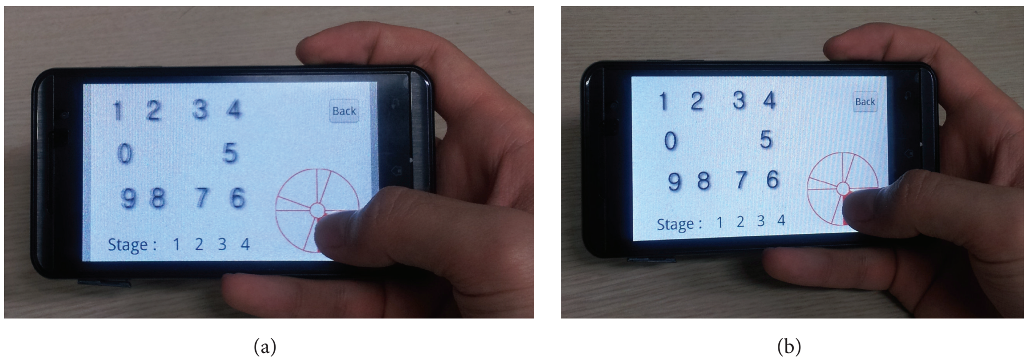


Figure 4: Implementation of 3DPIN (reproduction of Figure 2 in [8]). (a) Case wherein the camera sees blurred digits. (b) Case wherein the camera sees only one image.

in Figure 2(d). The completed stages are highlighted using black squares. If the user recognizes that a digit was input incorrectly, then s/he can revoke it by touching the "Back" button.

3. Implementation Details

We implemented 3DPIN on a smartphone equipped with the parallax barrier 3D display whose resolution is 800 × 480 pixels. The program was written in Java over Android 2.3.3. Figure 4 illustrates the proposed method at the moment when the 3D challenge shown in Figure 2(b) is given to the user. Figure 4(a) shows a photograph taken from the 3D spot right in front of the smartphone. The camera has only one lens, and we located this lens at the orthogonal point in front of the smartphone. Then, its view is slightly different from that of the left eye and that of the right eye. As a result, it only sees a blurred mixture of these two images. Figure 4(b) shows a photograph taken at a slight angle from the position of the left eye. We can see that the camera clearly captured the single image for the left eye.

We explain the realization of 3D effects in more detail. The smartphone that we used for the implementation allows us to express various levels of depth specified by an integer. If the depth is set as 0, it is located on the same plane of the LCD display, and the images for the left and right eyes are identical. On the other hand, a negative depth implies that the corresponding object is located at a deeper plane. This effect is realized by horizontally shifting an object for the left and right eyes to the left and right by an appropriate amount. The amount of shift is determined by the absolute value of the depth. The larger the absolute value, the more the object shifted. In contrast, to represent a positive depth, the object is shifted in the opposite direction.

Although various depth values may be assigned to each digit, we use only the numbers with fixed absolute values for a session. For example, we may assign +2 to the indicator digit and −2 to the other nine digits, or vice versa. The rationale for this assignment is related to the security and usability of the proposed method. First, we explain the security aspect. In Figure 4(a), we already observed that a one-lens camera may see a blurred image, which is caused by the horizontal shift that we explained in the previous paragraph. The shift of the indicator digit, that is, "8," should be done in the opposite direction from the other nine digits, because its depth has a different sign from that of the other digits. However, as shown in Figure 4(a), it is not easy to identify the image for either the left or the right eye from the blurred image. Therefore, it is not easy for a camera to find the indicator digit if all digits have the same absolute values in their depths. Our choice of depth is also justified from the viewpoint of usability. If all digits have distinct depths, a user may have difficulty discerning which one is the most prominent one or the deepest one. Moreover, the user has to know in advance whether the phantom indicator should be the most prominent one or the deepest one. By fixing the absolute values of depths, the user

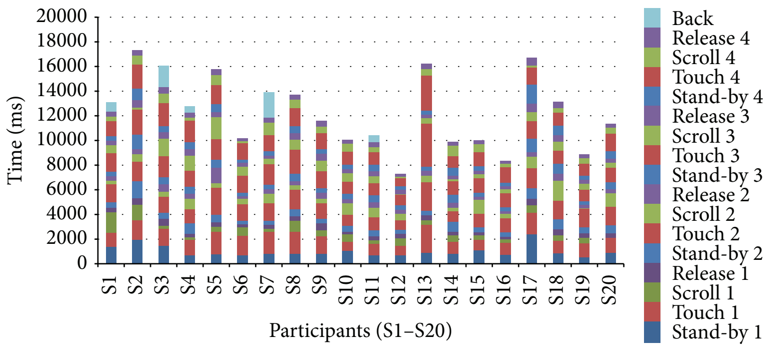


FIGURE 5: Distribution of authentication speed for 20 subjects (ms).

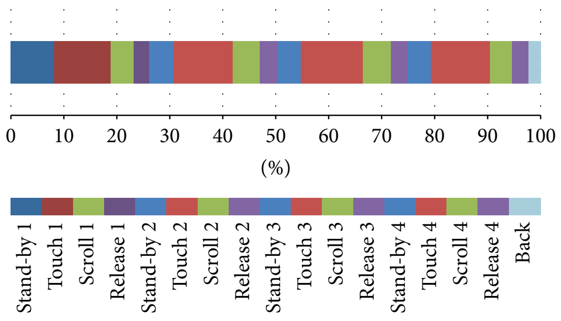


FIGURE 6: Tasks for an authentication session.

does not need to know this choice in advance, but s/he only has to find out the digit with a different depth from the other digits.

4. Performance Analysis

We conducted a usability study with twenty experimental subjects. Their ages ranged from 22 to 40 years, and six of the subjects were female. The purpose of this study was to analyze the rough performance of users and identify the bottleneck among various tasks comprising the authentication procedure for 3DPIN. At the beginning, the working mechanism of 3DPIN was explained in detail to each participant. Then, the participant was trained for 5 min to become accustomed to using 3DPIN and was guided to perform four authentication sessions. Two sessions were conducted using a fixed 4-digit PIN that the participant chose beforehand, and the other two sessions were conducted using randomly generated 4-digit PINs. When the participant failed to enter a correct PIN, s/he was asked to perform another session.

Now, we analyze the experimental results. Among the 80 sessions, 8 were failures and 8 more sessions were conducted. There was no failure in the retrial sessions. Therefore, the probability of erroneous input was 8/80 = 10.0%. Figure 5 shows the average time required for participants S1 to S20 to complete a session. Because we did not find any significant difference in the sessions using a fixed PIN and a random PIN, we did not distinguish between them in the graph. As a result, we obtained the median authentication time of 12.7 s with an average of 12.9 s and standard deviation of 2.9 s. Figure 5 also shows the tasks constituting an authentication session. That is, one stage to enter a single PIN digit is composed of four tasks, that is, *standby*, where the user gets ready to enter the next PIN digit; *touch*, where the user touches the scroll wheel and recognizes the phantom indicator; *scroll*, where the user scrolls the wheel so that the PIN digit is aligned with the phantom indicator; and *release*, where the user releases the finger after verifying the alignment. In addition, the user may perform one or more "*back*" tasks when s/he wants to cancel the incorrect input and redo the current stage.

Figure 6 shows the breakdown of times for these tasks, averaging all sessions of all participants. According to the measured data, the most time-consuming task is the "touch" task, where the participants try to identify the indicator digit by recognizing the difference in its 3D depth from the 3D depth of the other digits. It took about 1.38 seconds per stage. The participants also consumed nonnegligible time for the standby task in the first stage. We conjecture that this is because the users require some time to adjust themselves to execute the authentication application. The time for the "scroll" task in each stage is the sum of multiple movements. That is, a user has to rotate the digits by zero to five positions in an appropriate direction. However, the experimental results show that some participants failed to choose the optimal path in some stages. To be precise, nine among 320 stages had more than five movements. As a result, the average number of movements was 2.58, which is slightly greater than 2.5, the theoretically expected value. The total time required for the scroll task in each stage was 0.58 s on average.

We also had a short questionnaire session with each participant after the test. According to the survey data, 17

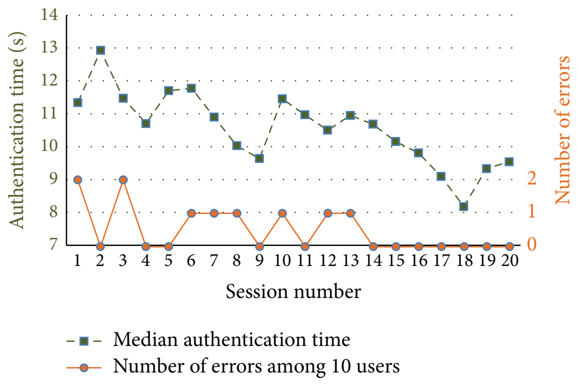


Figure 7: Change in median authentication time and error rate according to training.

Table 1: Comparison of PIN-entry methods for 4-digit PINs.

Method	Channel	Authentication time (s)	Error (%)	P_B	P_S
Regular PIN pad	—	<3	Approx. 0	1/10,000	Approx. 1.0
Undercover [11]	Haptic	32–45	>31.5	1/20,480	1/20,480
VibraPass [12]	Haptic	3.9–8.2	>14.8	1/10,000	1/70–1/5
Haptic Wheel [13]	Haptic	23.0	16.4	1/15,625	1/15,625
Phone Lock [9]	Haptic	28.2	10.4 (+5.6)	1/10,000	1/10,000
	Audio	12.2	4.8 (+6.9)	1/10,000	1/10,000
Spinlock [10]	Haptic	13.9–20.1	8.3 (+62.3)	<1/10,000	<1/10,000
	Audio	10.8–16.9	3.3 (+64.0)	<1/10,000	<1/10,000
3DPIN	3D	10.6	5.0	1/10,000	1/10,000

participants agreed that there should be a more secure PIN-entry method than the regular PIN pad, whereas the answers of 1 and 2 participants were negative and neutral, respectively. To the question asking whether they would use 3DPIN in daily life, 15 participants answered affirmatively, and 3 and 2 participants were negative and neutral, respectively. The reason for this encouraging result was that the participants felt very secure with 3DPIN. We asked whether they think that 3DPIN is more secure than the regular PIN pad and let them assign a score between 1 and 5 on the Likert scale. The average score was 4.25. However, they thought that 3DPIN was not as convenient as the regular PIN pad, giving a score of 2.05 on average to the question asking whether 3DPIN was more convenient.

Based on the results of the initial test, we designed another test. The purpose of this second test was to precisely analyze performance by using more session data. In addition, we tried to figure out the effect of training through repeated authentication sessions. It should be noted that the data in the initial test had been collected from participants who were not trained in 3DPIN. Then, the natural question is whether we may enhance the performance of the participants by training or not. Therefore, we formed a focus group of 10 volunteers from the 20 participants such that their average performance was approximately the same as the average of all participants. Then, each member of the focus group was guided to select his/her own PIN and perform an intensive experiment with 20 sessions with the fixed PIN. The dashed line shown in Figure 7 shows that the time required for a session gradually decreases, although there are a few exceptional values. For example, the median value among the 10 subjects was 9.5 s in the last session, and it temporarily dropped to even 8.2 s in the third-from-the-last session. The average over 20 sessions was 10.6 s. The solid line shown in Figure 7 shows that the error rate also decreases as the users get accustomed to the new method. For example, there is no erroneous input in the final seven sessions, whereas the error rate of 20 consecutive sessions is $10/(20 \times 10) = 5.0\%$ on average. Therefore, we may conjecture that the error rate will converge to zero after the users are sufficiently trained.

5. Comparison with Related Works

In this section, we compare the performance of 3DPIN with that of the previous secondary channel-based PIN-entry methods as well as the regular PIN pad. Table 1 summarizes the results. It lists various PIN-entry methods with the secondary channels that they use and shows the authentication time and error rate. For reference, we also compare the resistance against a brute force attack and a shoulder-surfing attack. The resistance against a brute force attack, denoted by P_B, represents the probability that the attacker may pass the authentication test by randomly guessing a PIN. The resistance against a shoulder-surfing attack, denoted by P_S, represents the attacker's success probability after observing one session.

As shown in Table 1, the proposed method guarantees much better performance than the previous methods. First, Undercover [11] and Haptic Wheel [13] require a considerable amount of time for authentication and the rate of erroneous input is too high to be deployed in real-life applications. Although VibraPass [12] is almost as fast as the regular PIN pad in some settings (with a "low lie overhead" using the term defined by De Luca et al. [12]), its security is not significantly better than that of the regular PIN pad. That is, a shoulder-surfer can reduce the size of the candidate PIN set to as small as five after observing only one authentication session. Moreover, it suffers from a relatively high error rate (De Luca et al. [12] defined an error as the case where the user inputs incorrect values in three consecutive trials, which represents the common practice in ATMs. We recalculated the error rate according to our definition of error). Phone Lock [9] and Spinlock [10] have a novel feature that they may be used with two different channels, that is, either a haptic channel or an audio channel. In particular, the schemes with an audio interface guarantee competitive authentication time. However, it should be noted that it does not include the time for reset where a user cancels a PIN-entry process, but it only includes the successful case with no error and no reset. The figures in the parentheses in the "Error" column of Table 1 represent the reset rate. We see that Spinlock requires a reset task in more than half of the authentication trials. Therefore, the authentication time will be significantly greater if we take the time for the reset task into account. As explained in the previous section, our data for authentication time already include the time for touching the "back" button and reentering the current PIN.

We should also take into account other usability aspects. First, note that an earphone should be prepared for a secure audio channel, which significantly degrades the usability and further increases the authentication time if we include the time for the user to pick up an earphone and connect it to the device. In addition, most of the previous methods are not fully compatible with 4-digit PINs. For example, Spinlock [10] uses the combination of digits and directions as in a dial-based safe, and the PIN is written as, for example, "5 to the right, 3 to the left, 4 to the right, and 2 to the left." If we want to enter the PIN for our bank account on a banking application of a smartphone, there should be a compatible mapping between the regular PIN and the Spinlock PIN. Because this mapping is not straightforward, the user essentially has to memorize two different PINs, that is, the 4-digit PIN for ATM banking and the Spinlock PIN for smartphone banking. Note that this is not the case in 3DPIN.

In summary, the PIN-entry time, the error rate, and the other usability aspects of 3DPIN are very promising as compared to those of related works, and it is a practical solution for authentication. In addition, as verified in the training test in the previous section, its performance may be significantly improved as compared to the values given in Table 1.

Finally, we briefly mention other related works. The concept of using a sweet spot for security is not a completely new one but has already been suggested in the context of visual secret sharing [17]. In this scheme, the secret information can be recovered only at a sweet spot when the two transparent shares are located in parallel with an exact amount of space between them. However, this approach cannot be applied to authentication. A 3D visual channel was also used by Lee and Nam [18] as a secure interface, but their method did not fully utilize the advantage of the 3D channel and did not aim at achieving the maximum level of security. As a result, its P_S was 1/10, which was significantly higher than that of the proposed 3DPIN although it was ten times lower than that of the regular PIN pad. One may also consider an approach to physically obstruct the attacker's view. For example, a user may shield the device with his/her hand [19]. However, this method may leak some partial information about a PIN. There is also a method that uses a back-of-device panel [20], but this interface is not available in most current off-the-shelf smartphones. Finally, we remark that another kind of secure visual channel may be implemented if an additional device with short-range communication capability is available [21, 22].

6. Discussion on Interface Design

In this section, we explain the interface design of 3DPIN in more detail and discuss a few issues to improve its performance and usability. First, we would like to remark that the final layout of 3DPIN shown in Figure 2 is the result of our multiple-round pilot study. In our initial design stage, we have considered the following four factors:

(i) Arrangement of digits: we considered three alternatives: linear (see Figures 8(a) and 8(b)), diamond (see Figure 8(c)), circular, and rectangular (see Figure 2) arrangements.

(ii) Indicator types: we considered two alternatives. The first one was to use visible and explicit indicator symbols as in the traditional dial lock. For indicators, ten distinct graphic symbols such as a diamond (◆), a star (★), and a club (♣) were selected and each symbol was located next to each digit from 0 to 9 (see Figure 8(a)). In this version, which can be regarded as a 3D version of the method in [7], the 3D effect was given to the indicators instead of digits. The second alternative was not to use any indicator, which is the final design shown in Figure 2.

(iii) Interface for digit movement: in the initial prototype shown in Figure 8(a), the digits were moved by two buttons, "Left" and "Right," which required too many button touches. As an alternative, we adopted scrolling interface, that is, a linear scroll pad for the linear arrangement (see the pad with five sections in Figure 8(b)) and a scroll wheel for the diamond, circular, and rectangular arrangements (see Figure 8(c)).

(iv) Fonts of digits: line thickness, size, and shapes of PIN digits may affect the performance of PIN-entry.

To decide the best design in the aspect of authentication time and error rate, we tried various combinations of the above factors in an informal pilot study and finalized the current design, that is, a rectangular arrangement with a phantom indicator and a scroll wheel. As for the choice of

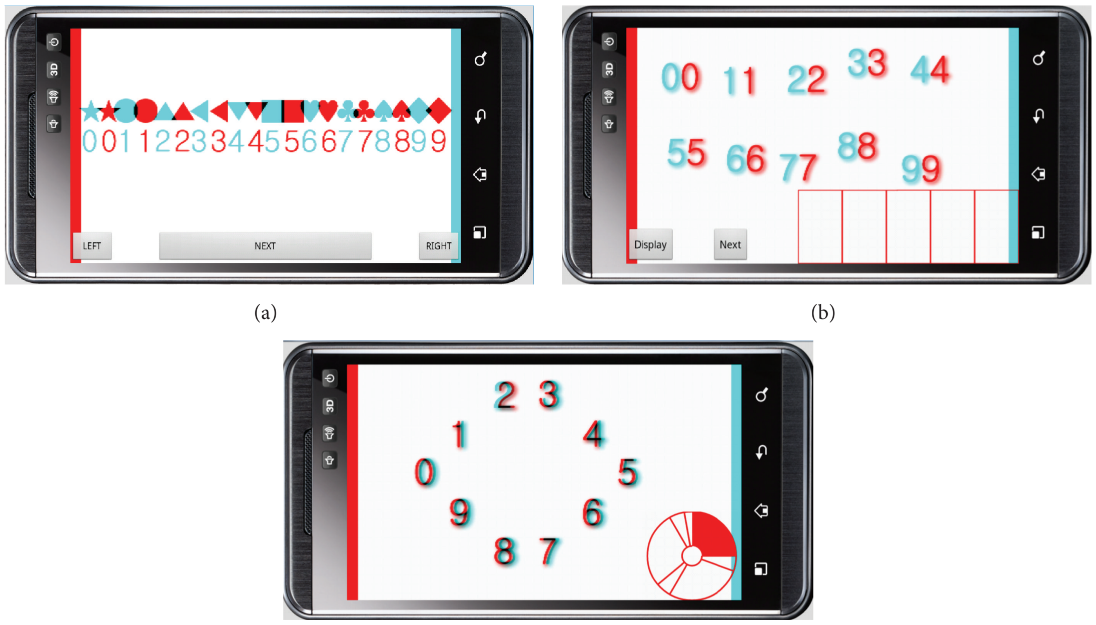


Figure 8: Alternative design of 3DPIN. (a) Linear arrangement, visible indicators (graphic symbols), and button interface for digit movement. (b) Linear arrangement, phantom indicator, and linear scroll pad. (c) Diamond arrangement, phantom indicator, and scroll wheel.

font, it was found out that a large and slim figure with a shadow effect was the best choice as shown in Figure 4.

However, we do not claim that the current design of 3DPIN is the optimal one. In the interview with the participants of our usability test, some participants raised questions about the layout of 3DPIN. The main issues were that the rectangular layout and the scroll wheel do not match each other intuitively and that sometimes participants tend to forget the position of the phantom indicator while rotating the digit array. This implies that, for better usability, qualitative aspects should be considered as well as quantitative aspects such as authentication time and error rate. In addition, the quantitative aspect itself could be improved. For example, according to the experimental results shown in Figure 6, the most time-consuming task is the "touch" task. Then, it would be promising to find an alternative design which focuses on reducing the bottleneck. For this purpose, a more thorough and in-depth adjustment and evaluation on user interface should be done. In addition, it would also be interesting to devise a tool to educate the users for better performance and verify the effectiveness of this tool by a more rigorous inspection, for example, by performing a regression analysis. We leave these issues as our future work.

7. Conclusion

In this paper, we proposed a PIN-entry method that uses the 3D display of a smartphone as the secure channel for data transmission. The proposed method, 3DPIN, guarantees fast authentication and low error rate by using an easy user interface and minimizing the amount of finger movement. However, there are still many open research issues to improve the usability of 3DPIN as mentioned in the previous section. In addition, it would be an interesting research direction to develop a 3D interface for entering a general password instead of a numeric PIN. Finally, the clear limit of 3DPIN is that it is restricted to devices with 3D displays. Therefore, the development of a unimodal PIN-entry method using only a standard visual channel would be promising.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work was supported by INHA University Research Grant. We would like to thank Jae Hyeong Lee and Hyeonjin Nam, for their help in the implementation and performance analysis, and all voluntary participants of our experiments.

References

[1] H.-M. Sun, "An efficient remote use authentication scheme using smart cards," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 4, pp. 958–961, 2000.

[2] J.-J. Shen, C.-W. Lin, and M.-S. Hwang, "A modified remote user authentication scheme using smart cards," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 2, pp. 414–416, 2003.

[3] S. Furnell, "An assessment of website password practices," *Computers and Security*, vol. 26, no. 7-8, pp. 445–451, 2007.

[4] J. Bonneau, S. Preibusch, and R. Anderson, "A birthday present every eleven wallets? The security of customer-chosen banking

PINs," in *Financial Cryptography and Data Security: 16th International Conference, FC 2012, Kralendijk, Bonaire, Februray 27-March 2, 2012, Revised Selected Papers*, vol. 7397 of *Lecture Notes in Computer Science*, pp. 25–40, Springer, Berlin, Germany, 2012.

[5] J. Bonneau, "The science of guessing: analyzing an anonymized corpus of 70 million passwords," in *Proceedings of the 33rd IEEE Symposium on Security and Privacy (SP '12)*, pp. 538–552, San Francisco, Calif, USA, May 2012.

[6] V. Roth, K. Richter, and R. Freidinger, "A PIN-entry method resilient against shoulder surfing," in *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS '04)*, pp. 236–245, October 2004.

[7] M.-K. Lee, "Security notions and advanced method for human shoulder-surfing resistant PIN-entry," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 4, pp. 695–708, 2014.

[8] M.-K. Lee, J. B. Kim, and M. K. Franklin, "3DPIN: enhancing security with 3D display," in *Proceedings of the IEEE 3rd Global Conference on Consumer Electronics (GCCE '14)*, pp. 129–130, Tokyo, Japan, October 2014.

[9] A. Bianchi, I. Oakley, V. Kostakos, and D.-S. Kwon, "The phone lock: audio and haptic shoulder-surfing resistant PIN entry methods for mobile devices," in *Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '11)*, pp. 197–200, ACM, 2011.

[10] A. Bianchi, I. Oakley, and D.-S. Kwon, "Spinlock: a singlecue haptic and audio PIN input technique for authentication," in *Haptic and Audio Interaction Design (HAID 2011)*, vol. 6851 of *Lecture Notes in Computer Science*, pp. 81–90, Springer, 2011.

[11] H. Sasamoto, N. Christin, and E. Hayashi, "Undercover: authentication usable in front of prying eyes," in *Proceedings of the 26th Annual CHI Conference on Human Factors in Computing Systems (CHI '08)*, pp. 183–192, ACM, April 2008.

[12] A. De Luca, E. von Zezschwitz, and H. Hußmann, "VibraPass: secure authentication based on shared lies," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI '09)*, pp. 913–916, ACM, Boston, Mass, USA, April 2009.

[13] A. Bianchi, I. Oakley, J. K. Lee, and D. S. Kwon, "The haptic wheel: design & evaluation of a tactile password system," in *Proceedings of the 28th Annual CHI Conference on Human Factors in Computing Systems (CHI '10)*, pp. 3625–3630, ACM, Atlanta, Ga, USA, April 2010.

[14] A. Bianchi, I. Oakley, and D.-S. Kwon, "Open sesame: design guidelines for invisible passwords," *Computer*, vol. 45, no. 4, pp. 58–65, 2012.

[15] A. S. Seiderman and S. E. Marcus, *20/20 is Not Enough: The New World of Vision*, Knopf, 1990.

[16] F. June, *An Introduction to 3D Computer Graphics, Stereoscopic Image, and Animation in OpenGL and C/C++*, CreateSpace Independent Publishing Platform, 2nd edition, 2011.

[17] K. Kobara and H. Imai, "Limiting the visible space visual secret sharing schemes and their application to human identification," in *Advances in Cryptology—ASIACRYPT 96*, vol. 1163 of *Lecture Notes in Computer Science*, pp. 185–195, Springer, Berlin, Germany, 1996.

[18] M.-K. Lee and H. Nam, "Secure and fast PIN-entry method for 3D display," in *Proceedings of the 7th International Conference on Emerging Security Information, Systems and Technologies (SECURWARE '13)*, pp. 26–29, August 2013.

[19] Q. Yan, J. Han, Y. Li, J. Zhou, and R. H. Deng, "Designing leakage-resilient password entry on touchscreen mobile devices," in *Proceedings of the 8th ACM Symposium on Information, Computer and Communications Security (SIGSAC '13)*, pp. 37–48, ACM, May 2013.

[20] A. De Luca, E. von Zezschwitz, N. D. H. Nguyen et al., "Back-of-device authentication on smartphones," in *Proceedings of the 31st Annual CHI Conference on Human Factors in Computing Systems (CHI '13)*, pp. 2389–2398, ACM, Paris, France, May 2013.

[21] D. K. Yadav, B. Ionascu, S. V. K. Ongole, A. Roy, and N. Memon, "Design and analysis of shoulder surfing resistant PIN based authentication mechanisms on google glass," in *Financial Cryptography and Data Security*, M. Brenner, N. Christin, B. Johnson, and K. Rohloff, Eds., vol. 8976 of *Lecture Notes in Computer Science*, pp. 281–297, 2015.

[22] D. Zhang, D. Zhang, H. Xiong, C.-H. Hsu, and A. Vasilakos, "BASA: building mobile Ad-Hoc social networks on top of android," *IEEE Network*, vol. 28, no. 1, pp. 4–9, 2014.

Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 5254141, 10 pages
http://dx.doi.org/10.1155/2016/5254141

Research Article

Anomaly Detection for Internet of Vehicles: A Trust Management Scheme with Affinity Propagation

Shu Yang, Zhihan Liu, Jinglin Li, Shangguang Wang, and Fangchun Yang

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

Correspondence should be addressed to Jinglin Li; jlli@bupt.edu.cn

Received 31 December 2015; Accepted 1 March 2016

Academic Editor: Seung Yang

Anomaly detection is critical for intelligent vehicle (IV) collaboration. Forming clusters/platoons, IVs can work together to accomplish complex jobs that they are unable to perform individually. To improve security and efficiency of Internet of Vehicles, IVs' anomaly detection has been extensively studied and a number of trust-based approaches have been proposed. However, most of these proposals either pay little attention to leader-based detection algorithm or ignore the utility of networked Roadside-Units (RSUs). In this paper, we introduce a trust-based anomaly detection scheme for IVs, where some malicious or incapable vehicles are existing on roads. The proposed scheme works by allowing IVs to detect abnormal vehicles, communicate with each other, and finally converge to some trustworthy cluster heads (CHs). Periodically, the CHs take responsibility for intracluster trust management. Moreover, the scheme is enhanced with a distributed supervising mechanism and a central reputation arbitrator to assure robustness and fairness in detecting process. The simulation results show that our scheme can achieve a low detection failure rate below 1%, demonstrating its ability to detect and filter the abnormal vehicles.

1. Introduction

Internet of Vehicles (IoV) is an open converged network system supporting human-vehicles-environment cooperation [1]. Fusing multiple advanced terms, such as VANET [2], autonomous driving [3], cloud computing [4], and multiagent system (MAS) [5], this hybrid concept plays a fundamental role towards a cooperative and effective intelligent transport system. An anomaly detection scheme is desirable in an environment filled up with uncertainty. Primarily, the security problem is motivated by the question [6] "How can I trust the information content I receive?" This issue is then decomposed into two subterms: "Is the communication channel via which I receive messages from a sender secure?" and "How can I trust the sender of the messages I receive?" The decomposition allows us to tell the difference between *computational trust* and *behavioral trust*. Being a complementary part of *computational trust* (such as encryption and tamper-proofing), the model of *behavioral trust* admits information's imperfection in an open system; therefore, individuals need extra trust-related information in decision-making [7]. Extra trust-related information could be extracted from history reputation or could be elicited from interaction experience between two individuals. Being capable of providing a measurement of trustworthiness, behavioral-based trust management enables intelligent vehicles to improve collaborations by reducing false or malicious behaviors. Anomaly detection technology is the key method to build behavioral trust.

This paper aims to detect anomaly vehicles in autonomous driving environment. As commercial IVs are drawing near, we have to face the facts that vehicles are more and more intelligent. Meanwhile, cyber vehicles are unprecedented and vulnerable when supported by an uncertain and dynamic network [8]. Malicious attacks and information tampering, along with system failures, will directly threaten human lives and properties. Anomaly vehicles include malicious vehicles and incapable vehicles. Malicious vehicles are entities with intentions to make damage in driving environment. Incapable vehicles are not intentionally to give negative influence; however, they may disturb order due to their limited capability. For example, an incapable intelligent vehicle could not behave properly in a rigid and accurate-ordered automatic driving platoon but may behave well in a normal driving pattern. On the other hand, a malicious intelligent vehicle should be forbidden in any situation. To highlight our motivations, we present the following two illustrative scenarios. *Scenario 1*: in cluster/platoon-based driving, IVs frequently communicate with each other to maintain lateral/longitudinal control. Vehicles with incapability or malicious intention may join the cluster or platoon. Their malicious or false behaviors are very likely to temper/disturb collaboration. In this safety-oriented case, local vehicles should be able to maintain robust intracluster trust to wipe out unqualified vehicles. *Scenario 2*: in efficiency-oriented case, where IVs need to collaborate in a broad area, they exchange message to presence traffic conditions, request parking plot information through VANET, and even negotiate routes to prevent traffic congestions. None of these three functions would be efficient without trustworthy collaboration. The above two scenarios suggest that a trust management scheme with anomaly detection is urgently need.

Solutions on IoV's anomaly detection still face many challenges raised by mobility including dynamic vehicle groups, real-time constraints, and intrinsic dynamic property of trust itself, which makes single or static trust measurement ineffective. Considering the mobile nature of vehicles, topology is changing so rapidly that preestablished trust relationships are likely to be invalid. As a result, two nodes need to build up trust in a timely fashion. Moreover, trust is not constant but changing along with different driving situations. An accurate trust should capture the context of interaction and history reputation. For example, a car with good reputation may not be trustworthy when it is over speed. Trust management system therefore calls for the ability to synthesize multiple resources, either from roads or from cloud. The essence of Internet of Vehicles is to obtain more safety and efficiency by integrating multiple infrastructures, networks, and vehicle intelligence. In accordance with this idea, we propose a hybrid approach called Cluster-Based Anomaly Detection (CAD). Figure 1 describes the framework of CAD. CAD is composed of two big components, namely, cluster-based trust component and central reputation component. Cluster-based trust component builds time-fashioned trust to reflect dynamic situation while central reputation component is to evaluate one's trust from a long-term perspective. These two components interact by evidence uploading and reputation providing. Cluster-based trust component has two major functions, namely, trust-based AP clustering and mutual supervision to maintain the robustness of dynamic trust.

The major contribution of this paper lies in the following two aspects:

(i) We identify cluster-based trust and reputation as two major components of anomaly detection. To exploit cluster-based trust, we propose a cluster-based trust evaluation algorithm, which modifies *Affinity Propagation Clustering* to generate the most trustworthy cluster head based on evaluation and communication. The algorithm runs in a distributed manner and shows robustness to malicious/incapable vehicles.

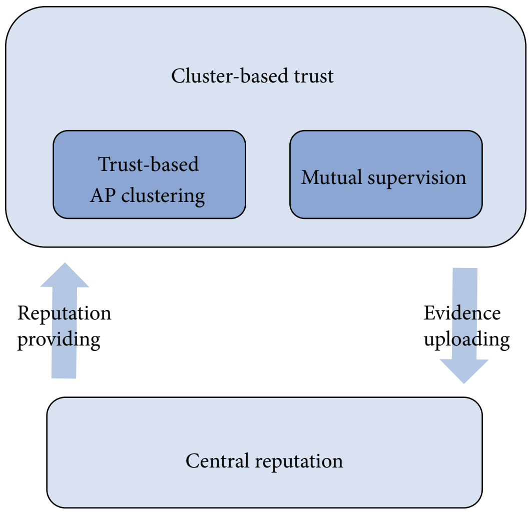


FIGURE 1: Framework of CAD.

(ii) We adopt a sparse RSU-enhanced reputation provision scheme. Central Arbitrator (CA) collects evidences from sparse RSUs. Then, a reputation system is established to evaluate global and history reputation from accumulated data.

2. Related Work

Trust issues stem from secure and social psychology fields and have been growing theoretically in organization management. More recently, as network technology is constantly changing the way people interact, former stable and well-structured organizations are likely to transform into another paradigm featured by agile structures and ad hoc groups. IoV, for example, is a typical agile structure that calls for collaboration among agents. Ramchurn et al. [9] pointed out that "trust pervades multiagent interaction at all levels," generally including (1) individual-level trust, whereby an agent has some beliefs about the honesty or reciprocative nature of its interaction partners, and (2) system-level trust, whereby the agents in the system are forced to be trustworthy by the rules of encounter that regulate the system. Although various schemes have been investigated, the author noticed that trust at these two levels has been dealt with separately in most times. This insight inspired us to develop a hybrid framework which takes both levels of trust into consideration.

Most existing systems in VANETs use distributed approach. Raya et al. [10] argue that the trust should be attributed to data per se in ephemeral ad hoc networks and proposed a framework for data-centric trust establishment. Their scheme shows high resilience to attackers and could converge to stable right decision. However, Raya's trust mechanism may make no contribution to reduce attackers in system level; since there is no punishment for cheating, attackers are seldom suppressed. Chen et al. [11] present a decentralized framework combined with message propagation and trust evaluation in VANET. Specifically, trust

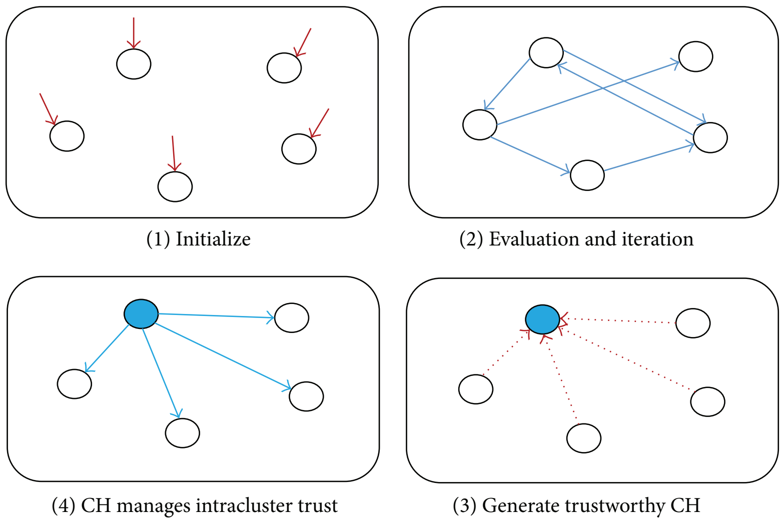

(1) Initialize
(2) Evaluation and iteration
(4) CH manages intracluster trust
(3) Generate trustworthy CH

Figure 2: Four steps of Peer Detecting-based trust establishment.

measurement consists of role-based trust and experience-based trust. It is a good attempt to synthesize static priori trust (role-based trust) with dynamic situational trust (experience-based trust). Nonetheless, they did not take historical reputation into consideration. Rostamzadeh et al. [12] focus on trustworthy information dissemination by assigning trust value to each road segment. The dissemination task is to find a path which consists of a series of safe road segments. Their work is featured by good scalability and thus potential in many applications. DTM^2 [13] is a distributed trust model inspired by Job Market model. With the help of third party hardware, system could incent good behaviors and punish malicious behaviors by changing each vehicle's signal value. To conclude, the decentralized approach is developed under the assumption that there is no centralized third party to evaluate and maintain the trust value.

Recently, the RSU deployment is promoted by intelligent transport system group. Centralized trust management is not an ambiguous goal with the help of RSUs. Centralized approach is able to evaluate trust value from a global and historical view. Therefore, many works have preliminarily emerged centralized trend as a complementary of distributed system. Wang et al. [14] proposed a vertical handoff method, which improves availability of network access. Their method therefore makes contributions to building centralized trust management system. Machado and Venkatasubramanian [15] aim to aggregate advantages of both centralized and distributed trust computation. The authors categorize the messages exchanged in VANET into *alerts* and *reports*; alerts are time-critical in response to an incident while reports are evidence to evaluate quality of alerts. RSUs play Central Authority (CA) who keep track of messages and accordingly maintain a global reputation for each vehicle. Their central grading system could efficiently distinguish dishonest nodes in real-life scenarios. Huang et al. [16] utilize identity-based cryptography to integrate entity-based trust and social trust in proxy server. The email interactions among individuals are mined to obtain social trust. Trust measurement should be requested and acquired from this server. One disadvantage of this system, as the author mentioned, is that service may experience long delay due to network latency and the management entities to mine the email source. Such latency problem bothers centralized reputation system. The author then proposes a situation-aware trust architecture for VANETs [17]. A predictive trust setup system is designed to reduce on-the-scene trust setup latency. They also envision that the roadside infrastructure deserves more attention and research.

3. Trust Establishment by Peer Detecting

In this section, we illustrate the establishment of cluster-based trust. To establish trust among IVs, the key is to generate the trustworthy CH. Cluster and its head are generated after several rounds of iteration. The generated CH is an authoritative node managing intracluster trust. One of the cluster algorithms which works by passing messages between nodes is *Affinity Propagation* (AP). To start, measures of similarities are calculated for each pair; real-valued messages are then exchanged between pairs of nodes until high quality exemplars and corresponding clusters gradually emerge. The schematic is shown in Figure 2.

AP works by passing messages between nodes, which is naturally more suitable for trust establishment than other clustering algorithms because of the following characteristics: (1) transitivity: in trust theory, if $node_A$ has no direct trust with $node_C$, it could still build an indirect trust relation via $node_B$ to $node_C$; likewise, in our AP, $vehicle_A$ makes a judgement about $vehicle_C$ with the help of indirect judgement from other nodes; the primitive AP clustering algorithm therefore well-reflects transitivity, making it fit into trust establishment; (2) asymmetry: trust is not symmetric; that $node_A$ trusts $node_B$ does not guarantee $node_B$ trusts $node_A$. AP has the ability to cluster by asymmetric "distance measurement"; (3) distributed manner: AP runs in a completely distributed manner, increasing robustness to attacks; (4) moreover, it achieves a much lower average squared error than normal clustering method [18].

The AP algorithm works iteratively. The similarity $s(i, j)$ is sent from $node_i$ to $node_j$ to measure "distance" between a pair. The responsibility $r(i, j)$ is sent from $node_j$ to $node_i$ to tell how eager i wants j to be CH. The availability $a(i, j)$ is sent from $node_i$ to $node_j$ to tell how eager j wants to be i's CH. The self-responsibility $r(i, i)$ and self-availability $a(i, i)$ both represent accumulated evidence reflecting if i is suitable to be CH. The updating process for responsibility and availability in every iteration procedure is illustrated below. More detailed works are [18, 19], which have lain the foundation of our work.

Primitive AP Iteration Process is as follows:

$$\begin{aligned} r(i,j) &\longleftarrow s(i,j) - \max_{k \text{ s.t.} k \neq j} \{a(i,k) + s(i,k)\}, \\ a(i,j) &\longleftarrow \min \left\{ 0, r(j,j) + \sum_{\forall k \neq i,j} \max\{0, r(k,j)\} \right\}, \\ a(j,j) &\longleftarrow \sum_{k \text{ s.t.} k \neq j} \max\{0, r(k,j)\}. \end{aligned} \quad (1)$$

To make real-valued message converge, messages are damped by λ, $Message_{new} = \lambda Message_{old} + (1 - \lambda) Message_{new}$, where λ is a weighing factor that ranges from 0 to 1. When messages converged, a CH is generated:

$$\mathrm{CH}_i = \max_j \{a(i,j) + r(i,j)\}. \quad (2)$$

3.1. UntrustDegree. Our proposed scheme uses the fundamental idea of Affinity Propagation from a trust perspective. In general, AP could detect anomaly vehicles in a group. We design an *UntrustDegree* function as "distance measurement" for AP algorithm to find "the most trustworthy node," that is, to find the node which minimizes overall UntrustDegree. The function *UntrustDegree*(i, j) is automatically calculated by IV. An IV can observe other vehicles' behaviors and give an UntrustDegree according to its knowledge:

$$\begin{aligned} &UntrustDegree(i,j) \\ &\quad = F_i\left(Identity, \overrightarrow{Situation}, \overrightarrow{Behavior}_j\right) \in [0,1]. \end{aligned} \quad (3)$$

Identity is one item from set $Id = \{bus, taxi, police, private, \ldots\}$ denoting real identity of one car and could be represented by a unique digital number. $\overrightarrow{Situation}$ is a vector predefined as some basic values which gives environmental context (e.g., the weather). $\overrightarrow{Behavior}_j$ is a vector recording basic actions that IV_j has done recently. With the help of behavior detection technologies [20] or interactive gaming [21], we reasonably assume that IVs are intelligent enough to evaluate each other. The value, $UntrustDegree(i, j) \in [0, 1]$, is primarily positive but set negative, namely, $-UntrustDegree(i, j) \in [-1, 0]$, to fit AP algorithm.

The self UntrustDegree, *UntrustDegree*(i, i), is initialized to the same value. It should be noted that a higher self-trust degree makes it more likely to become the cluster head. In our final model (discussed in Section 5.2), valid self-trust is set at a value which balances *IV's evaluation* and *historical reputation*. *Historical reputation* can only be legally announced by CA. When a group of IVs pass by a RSU, RSU will proactively download/broadcast reputations to IVs.

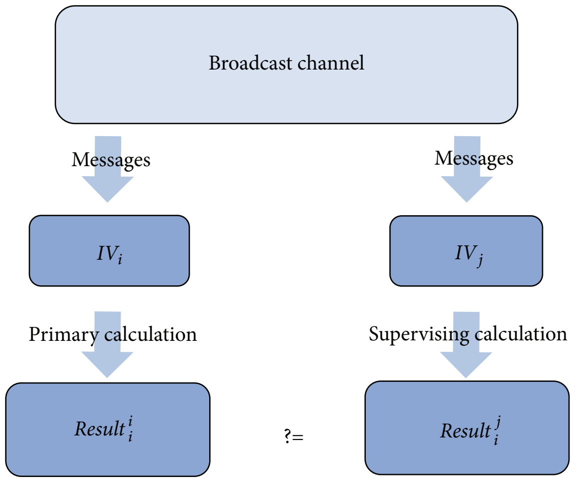


FIGURE 3: Mutual supervisor model.

3.2. Mutual Supervisor Model. Each IV_i receives responsibility $r(i, j)$ from the neighborhood. Also, IV_i broadcasts $a(i, j)$ to the neighborhood to claim how suitable it is to become a CH. However, a malicious/incapable node can cheat/mistake in this message passing process by broadcasting a false $a(i, j)$. For example, if IV_i broadcasts very high $a(i, j)$ to other nodes, it is more likely to be elected CH according to the AP algorithm. We need a mechanism to prevent nodes broadcast false availability or responsibility.

We proposed a supervisor model to alleviate cheating/mistaking in this process. The core of mutual supervisor model is to match IV_i with a supervisor IV_j. Among moving companions of one vehicle, a supervisor is another *IV* which can receive almost the same broadcast information by sharing the same wireless channel. A supervisor therefore listens to the supervisee related message to validate availability/responsibility by repeating the calculation of suspicious IV_i. The result calculated by IV_i itself is $Result_i^i$. The supervisor IV_j's calculation result for IV_i is $Result_i^j$. If the two results $Result_i^i$ and $Result_i^j$ have large difference, then this means IV_i is very likely to have cheated in message passing process. The integral mechanism of supervisor model is illustrated in Figure 3.

To assure a stable and honest supervisor, we apply Algorithm 1. From this algorithm, we see that IV_i has possibility to supervise another IV_j only when (1) IV_i does not tend to believe IV_j and (2) IV_i and IV_j have small relative mobility. That is, they are stable driving companions. This mechanism builds up mutual supervision relationship between two adversary nodes so that supervisor and supervisee are not likely to collude. More important, it can identify cheating nodes in message passing process.

```
**Input**: a supervisor $IV_i$, nearby node' states (position, speed)
**Output**: pair(supervisor, supervisee)
(1) For Node $IV_j$ in DSRC(Dedicated Short Range Communication) range
(2) $M_{i,j} = MobMetr((Pos_i, Pos_j), (Speed_i, Speed_j))$ //calculate mobility metric
(3)   If $IV_j$ has no supervisor and $UntrustDegree(i, j) \leq Threshold$
(4)         Then Add $IV_j$ in Supervisee Candidate List
(5) End For
(6) For $IV_j$ in Supervisee Candidate List
(7)     If $M_{i,j} < M_{\min}$ Then $k = j$      //find the most stable supervisee
(8) End For
(9) Return matched pair($IV_i$, $IV_k$)           //i supervises k
```

Algorithm 1: Supervisor Matching algorithm.

Table 1: Neighbor and supervision field.

Neighbor field		Supervision field	
$(x, y)_j$	Position of IV_j	$a'(k, j)$	IV_k's last availability received from IV_j
$(v_x, v_y)_j$	Speed of IV_j		
$UntrustDegree\,(i, j)$	UntrustDegree from IV_i to IV_j	$a'(j, k)$	IV_k's last availability sent to IV_j
$a(i, j)$	Last availability received from IV_j		
$a(j, i)$	Last availability sent to IV_j	$r'(k, j)$	IV_k's last responsibility received from IV_j
$r(i, j)$	Last responsibility received from IV_j		
$r(j, i)$	Last responsibility sent to IV_j	$r'(j, k)$	IV_k's last responsibility sent to IV_j
$CH_{cnvg,j}$	Cluster head converge flag for IV_j		
$SUPVE_j$	IV_j's supervisee		

The input of Algorithm 1 is IV's state tuples (position, speed). For IV_i, running this algorithm will work out a supervisee. For each IV_j in DSRC range, IV_i calculates mobility metric $M_{i,j}$ (lines (1)-(2)); the smaller the metric is, the more similar the two motions are. Thus, a small metric indicates a stable driving companion. If any IV_j has no supervisor and $UntrustDegree(i, j) \leq Threshold$ (this indicates that i does not tend to trust j), IV_i adds IV_j in Supervisee Candidate List (Line (3)-(4)). After that, IV_i chooses the most stable candidate (with the smallest mobility metric) to be the supervisee (lines (6)–(8)). Finally, a pair (IV_i, IV_j) is returned (line (9)).

3.3. Generating CH by Message Passing. We try to use a distributed algorithm to reach a consensus among large amounts of opinions. Each IV_i maintains a neighbor list N_i. As Table 1 shows, the list consists of N_i^j for each neighbor IV_j. Additionally, IV_i also maintains a supervision field for a supervisee IV_k.

Generating CH needs several iterations which are periodically triggered by time. Besides, broadcasting and supervising also need a synchronous clock. *Hello beacons* are broadcast and received to maintain local awareness.

Broadcast and Receive Hello Beacons Process is as follows:

(1) For every T_{hello}, each IV_j broadcast *hello beacon* is

$$\left\langle j, (x, y)_j, (v_x, v_y)_j, CH_j, SUPVE_j \right\rangle . \quad (4)$$

(2) Each receiving neighbor IV_i calculates *UntrustDegree*(i, j) if they are traveling in the same direction.

(3) IV_i adds/updates N_i^j in its neighbor list:

$$\left\langle j, (x, y)_j, (v_x, v_y)_j, UntrustDegree\,(i, j), CH_j, SUPVE_j \right\rangle . \quad (5)$$

Availability and responsibility messages should be broadcast periodically. We define this period as $T_{message}$. Each IV_i will calculate $a(j, i)$ and $r(j, i)$ for each neighbor IV_j. This value is damped with the previous value stored in the neighbor list. IV_j then broadcasts $a(j, i)$ and $r(j, i)$ of all neighbors IV_j.

According to mutual supervisor model, the process of calculating $a(j, i)$ and $r(j, i)$ should be supervised. Each IV automatically chooses a supervisee by *Supervisor Matching algorithm*. Supervisor checks supervisee's calculation result and releases *alert* on condition that supervisee's message is suspicious. The process enhanced by mutual supervisor model is illustrated below.

Supervising and Message Passing Process. For every $T_{message}$, each IV_i will do the following:

(1) It will find a matching supervisee IV_k which is prepared for the next T_{round}'s iteration. If found in this $T_{message}$, it is claimed by $SUPVE_j$. If failed, it will try next $T_{message}$.

(2) If it hears an *alert* about IV_n, each IV_i will ignore IV_n's messages in this T_{round}.

(3) It will calculate responsibility $r(i, j)$ for each neighbor IV_j.

(4) It will update with damping factor and store: $r(i, j) = (1-\lambda)r(i, j)_{new} + \lambda r(i, j)_{old}$.

(5) It will calculate availability $a(j, i)$ for each neighbor IV_j.

(6) It will update with damping factor and store: $a(j, i) = (1-\lambda)a(j, i)_{new} + \lambda a(j, i)_{old}$.

(7) It will determine if itself is converged to CH: if $r(i, i) + a(i, i) > 0$, then set $CH_{cnvg,j}$.

(8) It will broadcast Responsibility and Availability array, $r(i, j)$ and $a(j, i)$.

(9) It will supervise IV_k: IV_i updates and calculates $r'(j, k)$ and $a'(j, k)$ for IV_k.

(10) IV_i listens to IV_k's messages: $r(k, j)$ and $a(j, k)$; if $|r(k, j) - r'(k, j)| > Threshold$ or $|a(k, j) - a'(k, j)| > Threshold$, then it will broadcast *alert* about IV_k.

$T_{message}$ must be small enough to allow algorithm converged within a T_{round}. We have injected a supervision mechanism into clustering process. Any node that broadcasts false availability and responsibility would very likely be discovered. The punishment to malicious nodes is twofold: first, its message would be ignored by neighbors through *alert*; second, a malicious behavior would be reported to CA.

In any $round_r$, there is a CH_{r-1} generated from $round_{r-1}$. CH_{r-1} will claim its role and broadcast *Final Message*, which represents CH's final evaluation to each cluster member VI_i:

$$FinalMessage = \{UntrustDegree(CH_{r-1}, i)\}. \quad (6)$$

FinalMessage is trustworthy since it is sent from CH, which is elected as "the most trustworthy node" by all group members. Built upon *FinalMessage*, intracluster trust management is relatively reliable to support IVs' collaborations.

4. Degrading Anomaly by Evidence Evaluation

Reputation-based method has been widely used in web service [22, 23] and cloud computing [24] to enhance system reliability and robustness. We believe this method could also improve system performance in Internet of Vehicles. In this scenario, IVs will observe and evaluate qualities of each other. Moreover, they form *evidences* and report them to CA. CA is supported by strong storage and computational resources, thus being capable of computing reputation from a global view. A global reputation is valuable for on-the-road IVs to choose potential collaborators. More importantly, reputation can be increased or degraded, as a system-level enforcement, to incent good behaviors as well as to punish bad ones.

IVs leverage "store-upload" mechanism in delivering evidences to a CA. Since RSUs are sparsely deployed, each IV would store evidences in its storage firstly and then upload them when moving into a RSU's service range. Evidence evaluation lies in the core of reputation. CA is able to make a conclusion on certain behavior by evaluating and merging different pieces of evidences from different individuals. Note that not all evidences are consistent, and not all evidences are trustworthy. For instance, in order to disturb reputation system, a malicious node may report false evidences.

To mathematically model evidence evaluation, assume CA has to decide among several basic behaviors $\beta_i \in \Omega$, based on K pieces of evidences $\{e_k^j\}$ to IV_j which are uploaded from different k IVs. Let B^j denote the final judgement on behavior type of IV_j. The following three methods are leveraged to get a consensus evaluation, with the ability to filter false evidences.

(A) Majority Voting. The final evaluation accords with the majority. Given counts of each type of observed behaviors, $count_i$, the behavior type of VI_j is defined by

$$B^j = \beta_{\operatorname{argmax}(count_i)}. \quad (7)$$

(B) Weighted Voting. For each behavior, this method sums up all the votes value supporting this behavior. The votes are weighed by corresponding trust level r_k^i. Then, the type with the highest value is final evaluation:

$$B^j = \beta_{\operatorname{argmax}((1/count_i)\sum r_k^i)}. \quad (8)$$

(C) Bayesian Inference. Among the data fusion techniques, Bayesian Inference (BI) is the most popular one used for trust building and managing. To use BI, the a priori probability of each action β_i is firstly assigned. A posterior probability of each action β_i is calculated given a set of evidences $e = \{e_1^j, e_2^j, e_3^j, \ldots, e_k^j\}$ using Bayes' theorem. For IV_j,

$$P[\beta_i \mid e] = \frac{P[\beta_i] \times \prod_{k=1}^{K} P[e_k^j \mid \beta_i]}{\sum_{l=1}^{I} \left(P[\beta_l] \times \prod_{k=1}^{K} P[e_k^j \mid \beta_l]\right)}. \quad (9)$$

Final consensus is the actions type with the maximum posterior probability:

$$B^j = \beta_{\operatorname{argmax}(P[\beta_i|e])}. \quad (10)$$

Besides evidence evaluation, reputation evolution rule is another critical issue. An effective reputation system requires appropriate reputation evolving rules. We will discuss rules in Section 5.1.

5. Performance and Analysis

To evaluate performance of our scheme, we ran an extensive simulation in TransModeler with real map and high fidelity data. We use a map of urban area of San Antonio, USA. We feed real macroscopic traffic data, which are measured in critical roads and sections, to reconstruct real traffic scenario. We believe that macroscopic data could reflect traffic dynamic to a high extent. We do not simulate the wireless medium in this case since it is orthogonal to our evaluation. All simulations were performed with approximately 400 vehicles on a 6 miles' expressway. Five RSUs are sparsely deployed along the expressway as Figure 4. The DSRC range is set at

Table 2: Three basic behaviors.

β_i	Basic behavior type	UntrustDegree	Reputation change	Example
β_1	Life-critical events	0.9	Exponential degrade	2^7 (128) $\to$ 2^6 (64)
β_2	Efficiency reduction events	0.5	Linear degrade	128 $\to$ 120
β_3	Normality	0.1	No change	128 $\to$ 128

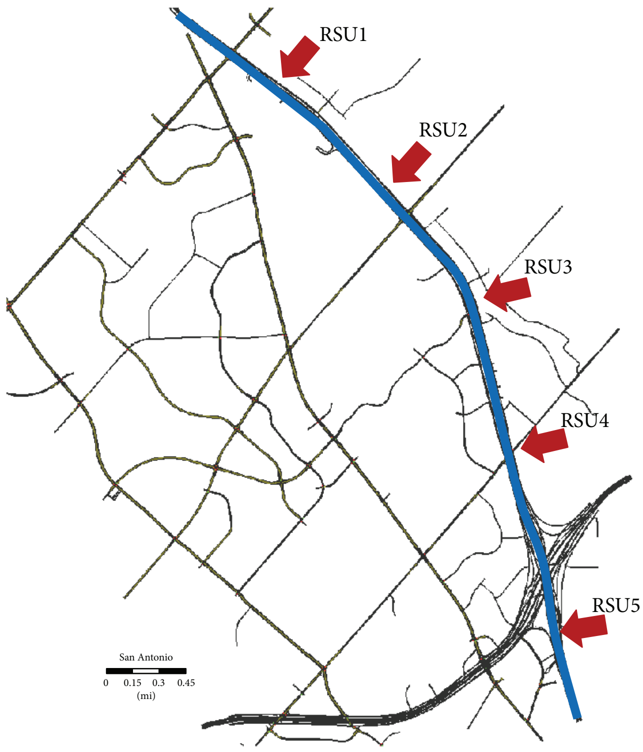


Figure 4: RSUs' deployment.

Table 3: Five behavior patterns.

Scenario number	Anomaly nodes behavior pattern		
	β_1	β_2	β_3
1	100%	0	0
2	0	100%	0
3	50%	50%	0
4	30%	50%	20%
5	20%	30%	50%

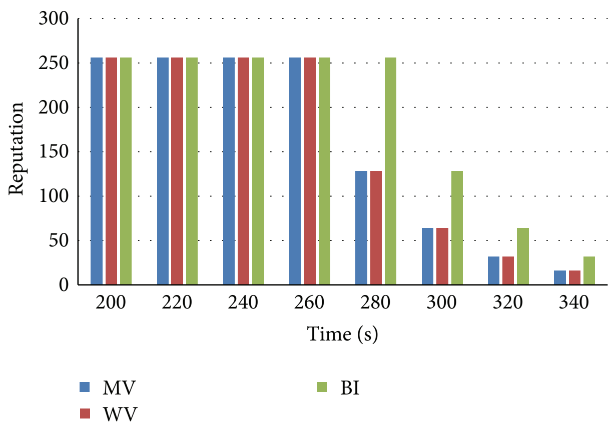


Figure 5: Effects of three merging techniques.

300 m. Each simulation ran for 600 s; however, only the last 400 s were used for performance metric calculations.

As noted earlier, an IV will be observed and evaluated by neighbor IVs. We use the example with three *Basic Behaviors* in Table 2. Each behavior causes different interactive trust. According to reputation evolution rules, one behavior deserves change in reputation.

To depict complex malicious/inappropriate behaviors, which are often mixed with different basic behaviors, we simulate several *behavior patterns* in Table 3. An anomaly node produces one behavior in every T_{round}. These patterns are simplified to make simulation feasible. We believe they could still well-reflect validity of our designed scheme.

5.1. The Effect of AP Algorithm. Ideally, AP clustering would generate a CH for every on-the-road vehicle. However, a small portion of vehicles, N_{left}, could be left alone when iterations are finished. There are two major reasons for these nodes: (1) the node could not find a converged CH candidate in its neighborhood and (2) the node itself is the CH but is the only member of cluster. Beside N_{left}, there are N_{in} nodes which form M normal clusters. In anomaly node-free simulation, several results are shown in Table 2. *Covered ratio* is a parameter describing how much the clustering results could cover the whole participants:

$$CoverRatio = \frac{N_{in}}{N_{in} + N_{left}}. \tag{11}$$

In anomaly simulation, several results are shown in Table 3. The simulations are ran several times so Figures 5, 6, 7, and 8 are averaged. A trade-off between *Covered Ratio* and *Cluster Member Number* could be found through Table 4. The higher the *Covered Ratio,* the lower the *Cluster Member Number*.

According to reference [18], *Damping Factor* is critical for convergence. Different *Damping Factors* result in different cluster outcomes. In general, a bigger *Damping Factor* leads to a relatively higher *Covered Ratio* and a lower *Cluster Member Number*. We recommend to set $\lambda \in [0.6, 0.8]$ so that algorithm tends to come out as an approximate but stable solution.

Another important parameter not mentioned in [18] is *Iteration Cycle*. Mathematically, the convergence of AP clustering is only influenced by *Damping Factor*, because

Table 4: The results of primary AP clustering.

Damping factor	Iteration cycle	DSRC range	Average cover-ratio	Average normal cluster number	Average member number	State
0.6	6	300 m	85.8%	26.5	13.2	Underdamping
0.7	5	300 m	38.9%	9.3	17.8	Underiteration
0.7	6	200 m	91.5%	46.7	8.5	Ok
0.7	6	300 m	94.3%	39.9	10.0	Ok
0.7	7	300 m	95.5%	85.5	4.8	Overiteration
0.8	6	300 m	74.3%	18.5	22.5	Overdamping

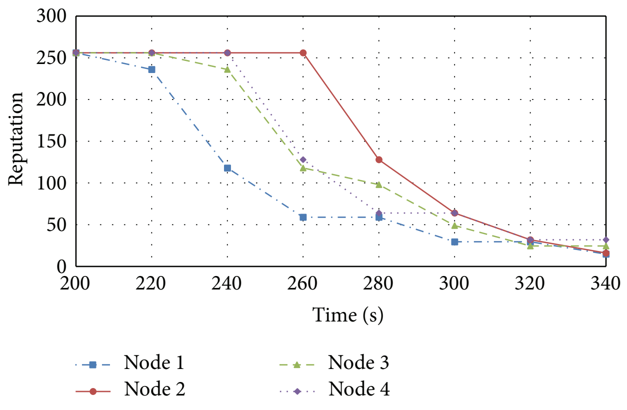


Figure 6: Reputation evolution of four anomaly nodes.

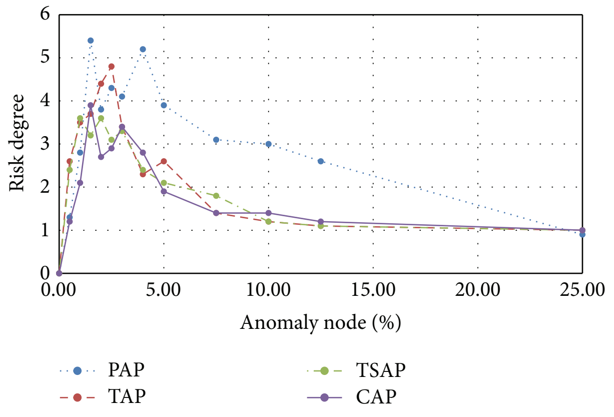


Figure 8: Percentage-Risk Degree curve.

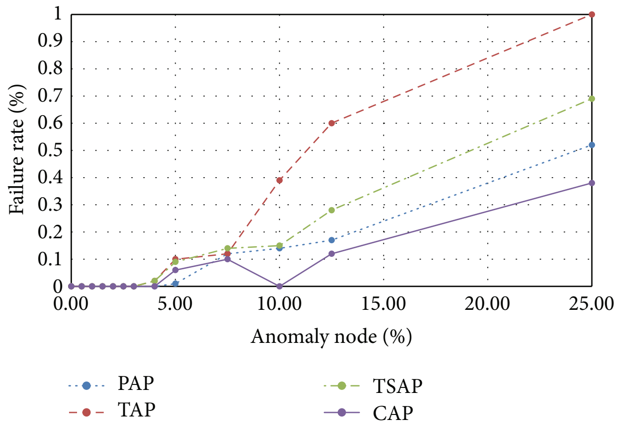


Figure 7: Percentage-Failure Rate curve.

the authors implicitly assume AP clustering could always have enough time for iteration. However, we have modified and applied this algorithm for anomaly detection where the communicating topology is constantly changing. Thus, the communication environment could not always provide plenty of time for clustering. So *Iteration Cycle* should be regarded as a critical parameter. If it is too short, clustering process will not be able to produce enough CHs to cover most nodes. On the other hand, if the cycle is too long, over-iteration will generate too many CHs. To conclude, modified AP clustering is oriented to real application other than a pure math problem and several parameters should be meticulously adjusted for real deployment, among which the most important ones are *Damping Factor* and *Iteration Cycle*.

We use two metrics to measure effectiveness of modified AP algorithm.

(1) Direct Influence. We define one *Failure* as an anomaly node elected to be CH; *Failure Rate* is to measure direct influence of one anomaly node, also called unsuccessful anomaly detection rate:

$$\text{Failure Rate} = \frac{\text{Number}_{\text{anomaly-CH}}}{\text{Number}_{\text{anomaly-node}}}. \quad (12)$$

(2) Indirect Influence. We define *Risk Degree* to feature how much potential influence an anomaly VI_i has when it is in one cluster:

$$\text{Risk Degree} = (1 - \text{UntrustDegree}(CH, i)) \times \text{Number}_{\text{cluster members}}. \quad (13)$$

If an anomaly node becomes CH, UntrustDegree is 0; otherwise, UntrustDegree is referred to CH's *Final Message*, which expresses CH's opinion of each node. *Risk Degree* could feature the indirect influence of an unqualified node according to its role (CH or member) and UntrustDegree.

Table 5: Prior distribution of behaviors and observed behaviors.

Observed	Behavior		
	β_1	β_2	β_3
β_1	0.7	0.15	0.1
β_2	0.2	0.7	0.1
β_3	0.1	0.15	0.8

For example, if an unqualified is admitted into a 20-member cluster, and CH's final message claims its UntrustDegree is 0.5, then its *Risk Degree* is 10. If it is admitted into a 5-member cluster, Risk Degree is 2.5. The later risk is much smaller because the anomaly node has fewer potential partners and thus may have fewer threats.

5.2. Comparison of Four Models. Our performance evaluation is based on four models: Primary AP model (PAP), Tempering AP model (TAP), Tempering&Supervising AP model (TSAP), and Converged AP model (CAP). PAP is directly derived from AP clustering algorithm. TAP models the clustering scenario where anomaly nodes could temper/disturb message passing process. In short, TAP considers tempering/disturb behaviors over PAP. To alleviate influence of tempering/disturbing, TSAP model injects *Mutual Supervision Model* into PAP to identify anomaly nodes. Finally, CAP is a converged model which enhanced TSAP with historical reputation.

As a converged model, CAP combines historical reputation with real-time cluster-based trust. CA collects uploaded evidences from on-road vehicles and uses three techniques to fuse evidences: (1) Majority Voting, (2) Weighted Voting, and (3) Bayesian Inference. For Bayesian Inference, the prior distribution of behaviors and observed results are defined in Table 5.

We assume that anomaly nodes use a random reporting strategy, which means they generate evidences randomly regardless of what other nodes really have done. Normal nodes will always report true evidences. Figure 5 describes the effects of different evidence merging techniques. In this simulation, three techniques are almost equally effective. However, MV and WV are more suitable for data merging since they have less computation overhead. Figure 6 shows four anomaly nodes' reputation evolves in system. Anomaly nodes would be distinguished and punished by CA.

In a process of iteration, IVs with larger values of Self-UntrustDegree are more likely to be chosen as CH. These values are "preferences." In PAP/TAP/TSAP, IV_n preference is set as median of $UntrustDegree(i, n)$. However, in CAP where historical reputation is considered by algorithm, IV_n's preference is calculated by

$$Preference_n = PunishFactor_n \times Median\left(UntrustDegree\left(i, n\right)\right). \quad (14)$$

When IV_n's reputation is low, $PunishFactor_n \in [1, 2]$ is big; preference therefore becomes small (preference is a negative real number), indicating IV_n is not suitable to be CH.

Figure 7 shows the comparison of four models. We set unqualified node percentage as variable. We simulate with different percentages ranges from $[0, 0.25]$ because too high percentage is not realistic.

Generally, when anomaly nodes percentage is low (≤5%), *Failure Rate* is 0%. As percentage goes up, *Failure Rate* also goes higher. TAP is a model with tempering/disturbing and no supervision mechanism, so it performs worse than PAP (no tempering/disturbing) and TSAP (tempering/disturbing, supervision model). In contrast, CAP is a converged model (tempering/disturbing, supervision model, and reputation) with a strong defense to anomaly nodes, so it shows the highest robustness among four models. Furthermore, either model could limit failure rate below 1% even when anomaly nodes percentage is up to 25%.

Risk Degree features how much potential influence an anomaly node has when it is in one cluster. Figure 8 shows the *Anomaly Percentage-Risk Degree* curve for four models. Risk Degree is firstly low when *Anomaly Node Percentage* is low. However, it suddenly goes to peak when *Percentage* slightly increases. Finally, it stably declines with increasing *percentage*. The explanation for this curve is as follows: (1) when *Percentage* is very low (≤1%), tempering/disturbing is few, and anomaly nodes therefore are easily distinguished by normal nodes. As a result, anomaly nodes are very likely to be left alone. That is, they are excluded from big clusters by AP algorithm. So the overall *Risk Degree* is low. (2) When *Percentage* goes higher but not that high (≤5%), this percentage still indicates a "safe environment"; IVs tend to form "big clusters." However, with more anomaly nodes percentage, more anomaly nodes have chances to join big clusters by more tempering/disturbing. According to formula (13), even one anomaly node in a big cluster would cause a big risk degree. (3) When *Percentage* increases over 5%, our algorithms tend to be conservative and form "small clusters," which have fewer cluster members. Fewer members render lower *Risk Degree*. According to Figure 8, CAP could limit Risk Degree under 4, demonstrating that our trust management is effective on risk control.

6. Conclusion and Future Work

Our system aims to build a trustworthy platform to detect abnormal vehicles. To this end, we modified *Affinity Propagation* to elect a most trustworthy node, called cluster head, among vehicles. CH maintains trust management during a period until a new CH is elected. We also considered that AP is executed in a distributed manner thus easily tempered by malicious nodes. So we presented a mutual supervision model to tackle tempering behaviors. Lastly, we blend another component, CA, into our system. CA consisted of servers and sparse RSUs and is able to provide historical reputation for better decision-making. Overall, this trust management system could detect and filter anomaly nodes.

In the future, great efforts are needed on both the in-vehicular system and RSUs to strengthen our secure system. These efforts include deploying mobile and local CA using cloud computing techniques, improving intelligence of mutual trust evaluation, and reducing overhead of detection process.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work is supported by the National High-Tech Research and Development Program (863) of China under Grant no. 2012AA111601 and 2015 Construction of key discipline under no. 700200253.

References

[1] F. Yang, S. Wang, J. Li, Z. Liu, and Q. Sun, "An overview of internet of vehicles," *China Communications*, vol. 11, no. 10, pp. 1–15, 2014.

[2] Q. Yuan, Z. Liu, J. Li, J. Zhang, and F. Yang, "A traffic congestion detection and information dissemination scheme for urban expressways using vehicular networks," *Transportation Research Part C: Emerging Technologies*, vol. 47, no. 2, pp. 114–127, 2014.

[3] M. Gerla, E.-K. Lee, G. Pau, and U. Lee, "Internet of vehicles: from intelligent grid to autonomous cars and vehicular clouds," in *Proceedings of the IEEE World Forum on Internet of Things (WF-IoT '14)*, pp. 241–246, Seoul, Republic of Korea, March 2014.

[4] Y. Leng and L. Zhao, "Novel design of intelligent internet-of-vehicles management system based on cloud-computing and Internet-of-Things," in *Proceedings of the International Conference on Electronic and Mechanical Engineering and Information Technology (EMEIT '11)*, vol. 6, pp. 3190–3193, Harbin, China, August 2011.

[5] D. Weyns, T. Holvoet, and A. Helleboogh, "Anticipatory vehicle routing using delegate multi-agent systems," in *Proceedings of the 10th IEEE Conference on Intelligent Transportation Systems (ITSC '07)*, pp. 87–93, IEEE, Seattle, Wash, USA, October 2007.

[6] V. Gligor and J. M. Wing, "Towards a theory of trust in networks of humans and computers," in *Security Protocols XIX*, Lecture Notes in Computer Science, pp. 223–242, Springer, Berlin, Germany, 2011.

[7] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *Proceedings of the 17th IEEE Symposium on Security and Privacy*, pp. 164–173, IEEE, Oakland, Calif, USA, May 1996.

[8] M. Raya and J.-P. Hubaux, "The security of vehicular ad hoc networks," in *Proceedings of the 3rd ACM workshop on Security of ad hoc and Sensor Networks (SASN '05)*, pp. 11–21, ACM, November 2005.

[9] S. D. Ramchurn, D. Huynh, and N. R. Jennings, "Trust in multi-agent systems," *The Knowledge Engineering Review*, vol. 19, no. 1, pp. 1–25, 2004.

[10] M. Raya, P. Papadimitratos, V. D. Gligor, and J.-P. Hubaux, "On data-centric trust establishment in ephemeral ad hoc networks," in *Proceedings of the 27th IEEE Communications Society Conference on Computer Communications (INFOCOM '08)*, pp. 1912–1920, Phoenix, Ariz, USA, April 2008.

[11] C. Chen, J. Zhang, R. Cohen, and P.-H. Ho, "A trust modeling framework for message propagation and evaluation in VANETs," in *Proceedings of the 2nd International Conference on Information Technology Convergence and Services (ITCS '10)*, IEEE, August 2010.

[12] K. Rostamzadeh, H. Nicanfar, N. Torabi, S. Gopalakrishnan, and V. C. M. Leung, "A context-aware trust-based information dissemination framework for vehicular networks," *IEEE Internet of Things Journal*, vol. 2, no. 2, pp. 121–132, 2015.

[13] N. Haddadou and A. Rachedi, "DTM2: adapting job market signaling for distributed trust management in vehicular ad hoc networks," in *Proceedings of the IEEE International Conference on Communications (ICC '13)*, pp. 1827–1832, IEEE Press, Budapest, Hungary, June 2013.

[14] S. Wang, C. Fan, C.-H. Hsu, Q. Sun, and F. Yang, "A vertical Handoff method via self-selection decision tree for internet of vehicles," *IEEE Systems Journal*, vol. 99, pp. 1–10, 2014.

[15] R. G. Machado and K. Venkatasubramanian, "Short paper: establishing trust in a vehicular network," in *Proceedings of the IEEE Vehicular Networking Conference (VNC '13)*, pp. 194–197, IEEE, December 2013.

[16] D. Huang, Z. Zhou, X. Hong, and M. Gerla, "Establishing email-based social network trust for vehicular networks," in *Proceedings of the 7th IEEE Consumer Communications and Networking Conference (CCNC '10)*, IEEE, Las Vegas, NVev, USA, January 2010.

[17] D. Huang, X. Hong, and M. Gerla, "Situation-aware trust architecture for vehicular networks," *IEEE Communications Magazine*, vol. 48, no. 11, pp. 128–135, 2010.

[18] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007.

[19] C. Shea, B. Hassanabadi, and S. Valaee, "Mobility-based clustering in VANETs using affinity propagation," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '09)*, IEEE, Honolulu, Hawaii, USA, December 2009.

[20] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: a survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1773–1795, 2013.

[21] T. Lei, S. Wang, J. Li, I. You, and F. Yang, "Detecting and preventing selfish behaviour in mobile ad hoc network," *The Journal of Supercomputing*, 2015.

[22] S. Wang, Z. Zheng, Z. Wu, M. R. Lyu, and F. Yang, "Reputation measurement and malicious feedback rating prevention in web service recommendation systems," *IEEE Transactions on Services Computing*, vol. 8, no. 5, pp. 755–767, 2015.

[23] S. Wang, L. Huang, C.-H. Hsu, and F. Yang, "Collaboration reputation for trustworthy Web service selection in social networks," *Journal of Computer and System Sciences*, vol. 82, no. 1, pp. 130–143, 2016.

[24] K. Hwang, S. Kulkarni, and Y. Hu, "Cloud security with virtualized defense and reputation-based trust mangement," in *Proceedings of the 8th IEEE International Symposium on Dependable, Autonomic and Secure Computing (DASC '09)*, pp. 717–722, IEEE, Chengdu, China, December 2009.

Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 8973828, 11 pages
http://dx.doi.org/10.1155/2016/8973828

Research Article

Security Analysis and Improvement of Fingerprint Authentication for Smartphones

Young-Hoo Jo,[1] Seong-Yun Jeon,[2] Jong-Hyuk Im,[2] and Mun-Kyu Lee[2]

[1]*Electronics and Telecommunications Research Institute, Daejeon 34129, Republic of Korea*
[2]*Department of Computer and Information Engineering, Inha University, Incheon 22212, Republic of Korea*

Correspondence should be addressed to Mun-Kyu Lee; mklee@inha.ac.kr

Received 6 November 2015; Accepted 24 January 2016

Academic Editor: Seung Yang

Currently, an increasing number of smartphones are adopting fingerprint verification as a method to authenticate their users. Fingerprint verification is not only used to unlock these smartphones, but also used in financial applications such as online payment. Therefore, it is very crucial to secure the fingerprint verification mechanism for reliable services. In this paper, however, we identify a few vulnerabilities in one of the currently deployed smartphones equipped with fingerprint verification service by analyzing the service application. We demonstrate actual attacks via two proof-of-concept codes that exploit these vulnerabilities. By the first attack, a malicious application can obtain the fingerprint image of the owner of the victimized smartphone through message-based interprocess communication with the service application. In the second attack, an attacker can extract fingerprint features by decoding a file containing them in encrypted form. We also suggest a few possible countermeasures to prevent these attacks.

1. Introduction

Recent advances in smartphone technologies enabled users to do various tasks using their smartphones. These tasks include not only simple ones such as playing mobile games and surfing the web, but also more critical ones, in particular, those dealing with private information and financial data. Therefore, a reliable mechanism is required to verify the identity of a person who tries to use the device. However, traditional secret knowledge-based solutions such as passwords, numeric PINs, and pattern locks have security issues such as password guessing attacks, brute-force attacks, and shoulder-surfing attacks. Moreover, they also have usability issues because a user must memorize some information and do a cumbersome task for log-on such as typing a password and drawing a pattern. In order to address these issues, fingerprint recognition is now being used for many smartphones, for example, iPhone 5s, Galaxy S5, and VEGA Secret Note. Fingerprint recognition is used both for unlocking a smartphone and for activating other security-critical functionalities in the smartphone, for example, for approving transactions in financial applications [1].

Therefore, it is very crucial to secure the fingerprint recognition service from possible threats such as intercepting a fingerprint image between an image sensor and a fingerprint recognition application and stealing the fingerprint data stored in a smartphone. Unfortunately, however, some of the currently deployed devices do not seem sufficiently safe against those threats. In this paper, we disclose the vulnerabilities in the fingerprint recognition service of VEGA Secret Note by analyzing the service application and demonstrate possible attacks against this service. (The VEGA series is one of the earliest smartphones with fingerprint recognition service, which is prior to recent popular ones such as iPhone 5s and Galaxy S5 [2]. The vulnerabilities were found on the device with Android 4.2.2 as of April, 2014. We reported these two vulnerabilities to the vendor. The second vulnerability was already addressed through a patch, and the vendor commented that the first vulnerability will also be addressed in the upcoming version.) VEGA Secret Note is an Android-based smartphone with a Qualcomm Snapdragon CPU (Krait 400), 3 GB RAM, and a 5.9-inch IPS touch display. It is equipped with an FPC fingerprint sensor on its back.

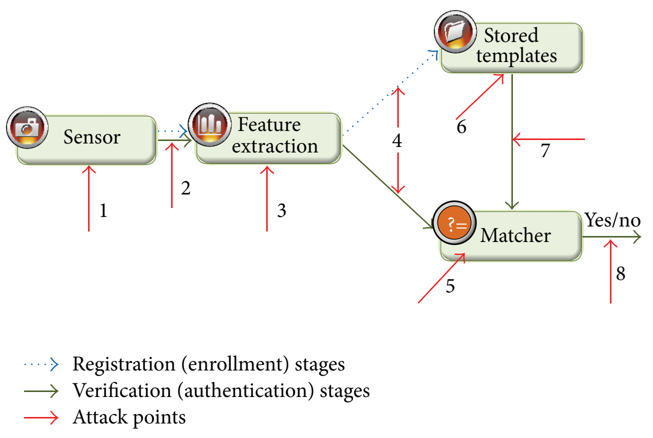


Figure 1: Generic structure of a biometric verification system and possible attack points (adopted from [4, 5]).

Our first attack is to enable a malicious application to acquire the fingerprint image of the owner of the victimized smartphone by accessing the memory space that the fingerprint recognition service application uses to temporarily store the image. In a nutshell, this attack exploits the design flaw of the service application which violates the principle of least privilege for access control [3]. To be precise, when a client application requests the service application to do fingerprint authentication, the service application activates a component which deals with the image of a scanned fingerprint. This component has been ill-designed so that it calls back an event handler in the client application with a reference to the memory location containing this image. As a result, the malicious client application can obtain the bitmap image by letting the component be activated and handling the event raised by that component.

Our second attack is to extract a stored template from the nonvolatile memory and restore fingerprint feature points by decoding the template. By identifying and analyzing a fingerprint service application on the target device, we identified the location of the stored template. In addition, we discovered that the template was encrypted, but the same key and initial vector (IV) are hard-coded and are the same for all devices. This design results in a vulnerability that a malicious user may be successfully authenticated if she/he overwrites a template by another template copied from his/her own device. In addition, by analyzing the structure of the decrypted template file, we were able to restore all feature points constituting the fingerprint template. This implies that a carefully forged template according to the file structure also may pass the authentication test.

Although we concentrated on a specific device in conducting our experiments, the technical flaws we have found in this device are a common trap that developers may fall into. Therefore, we suggest a few possible countermeasures to mitigate those vulnerabilities. We expect that the findings we obtained through our analysis may be used as a general guideline to design a secure biometric verification service on smartphones.

The remainder of this paper is organized as follows. Section 2 provides the preliminary information about the organization of a generic biometric system, a standard format for a fingerprint template, and the message-based communication mechanism between Android processes. In Section 3, two vulnerabilities and their possible consequences are explained in detail. A few possible countermeasures to these vulnerabilities are discussed in Section 4. Finally, we conclude the paper in Section 5.

2. Preliminaries

2.1. Threat Model against Biometric Verification. A generic biometric system can be cast in the framework of a pattern recognition system [4]. Figure 1, which was adopted from [4, 5], summarizes the typical stages in this generic system. A biometric system has two main procedures: registration (enrollment) of biometric data and verification (authentication) of biometric data, which are represented as blue dotted lines and green solid lines in Figure 1, respectively. The first stage of registration is to acquire the original biometric signal (typically, an image) using a sensor. The next stage is to extract invariant features from this original signal to construct a robust representation for biometric data that can uniquely determine an individual. The extracted features are stored as a form of a template. In the case of fingerprint recognition, a template contains fingerprint minutiae points. A minutia point is a peculiar point in a fingerprint image, for example, where a ridge either begins or divides into two ridges. A typical fingerprint may have tens of such points, and those points forming a template uniquely determine the characteristic of a specific fingerprint. Current fingerprint recognition systems are very accurate; in particular, they can provide a false rejection rate of 0.01% at a false acceptance rate of 0.1% [6].

The first and second stages of biometric verification are similar to those of registration. However, instead of storing the extracted features, the system runs a matching algorithm to compare the features derived from the current input biometric with those of the stored template. The matcher makes a decision, that is, whether to accept the user or not, based on the matching score.

Figure 1 also specifies eight places in the generic biometric system where attacks may occur. These points are represented as red lines in the figure and correspond to each item in the following list. This list is an extended version of the lists in [4, 5]. By a *passive attacker* we mean an attacker who steals or eavesdrops the secret information about the biometrics but who does not modify anything. On the contrary, an *active attacker* is an attacker who modifies the original biometric signal, template, or matching result to thwart a biometric verification service.

(1) A passive attacker may steal the original biometric signal by accessing the memory space the sensor uses to temporarily store this signal. In addition, fake biometrics such as a fake fingerprint, a copied signature, and a face mask can be presented for an active attacker to impersonate a legitimate user.

(2) A passive attacker may eavesdrop the original biometric signal sent from the sensor and store it in its own storage for later use. On the other hand, an active attacker may replay previously stored biometric signals bypassing the sensor. As a result, the attacker can impersonate the owner of that biometric. Note that a passive attacker may use the eavesdropped data to play a role of an active attacker.

(3) An active attacker may override the feature extraction module so that it produces only preselected features, ignoring the input from the sensor. A passive attacker may mount a backdoor which sends the extracted features back to him/her.

(4) The communication channel from the feature extraction module to either the template storage or the matcher may be tapped by a passive attacker. An active attacker may modify the packets and let the transmitted template be replaced with his/her own one. The purposes of these attacks are the same as those of the above type 3 attacks.

(5) An active attacker may corrupt the matcher so that it produces preselected matching scores without reference to the actual matching algorithm.

(6) A passive attacker may steal the stored templates, and an active attacker may modify the stored templates to force the system to authorize a fraudulent user or deny service to a legitimate user.

(7) The data sent from the template storage to the matcher may be intercepted by a passive attacker or modified by an active attacker. The results of these attacks are the same as those of the above type 6 attacks.

(8) The attacker may override the final decision with his/her intended result.

In this paper, we will present two passive attacks against the fingerprint recognition system of a VEGA Secret Note smartphone. Our first attack was to acquire the original biometric signal by injecting a malicious code independent of the original biometric application program and accessing the memory space where the biometric signal was stored. Our second attack was to directly access the stored template, not passing through the biometric application program. Therefore, the first attack can be viewed as a passive type 1 or type 2 attack, and the second attack can be viewed as a passive type 6 attack. We remark that even though we only demonstrate passive attacks, the output of our attacks may also be immediately used for active attacks. Although type 3, type 5, and type 8 attacks need an attacker's modification of the original biometric application, the effects of these attacks are the same as those of our attacks. Thus, we did not try to mount type 3, type 5, and type 8 attacks. In addition, type 4 and type 7 attacks were not required either, because stored templates were already manipulated by our type 6 attack.

2.2. Biometric Verification Using Fingerprint Minutiae.

Many devices that deal with fingerprints, including our target device, use the fingerprint minutia formats based on ISO/IEC 19794-2 [7] and ANSI INCITS 378 [8]. According to these standards, four main characteristics of minutiae are considered. These four characteristics are the x and y coordinates of the minutia on the original fingerprint image, the angle (θ) of the ridge corresponding to this minutia point, and ridge types. Although there are many distinct ridge types, two major types among them, that is, a ridge ending (also known as a ridge termination) and a ridge bifurcation, are frequently used in most settings [7–10], where a ridge ending stands for a point where a ridge suddenly ends and a ridge bifurcation is a point where a ridge divides into two ridges. See Figure 1 in [11] for the concrete examples of these two ridge types.

For biometric verification, a matcher compares the features extracted from the current sensor image with the stored template which is composed of multiple minutia points. The comparison is done by comparing (x, y, θ) of each fingerprint minutia point in the stored template with those from the sensor. A matching score is increased whenever each point matches. If the score is larger than a predefined threshold, the user is permitted to access the target device (see Figure 2).

2.3. Message-Based Communication between Processes in Android.

Android supports messages for interprocess communication (IPC) [12]. It enables an application to share an object with another application by sending a reference to the object to the target application. Figure 3 shows an example procedure where two applications communicate with each other through messages. As shown in this figure, a typical communication between two applications is done according to the following scenario. Throughout the paper, an item written in `typewriter` font represents a name of a class, an object, or their field.

(1) First, application A sends an intent to initiate a communication with application B, where an intent is a kind of signal to abstractly describe an operation to be performed [13]. An intent contains a parceled `Messenger` object which is a reference to the data which A wants to share. In addition, the intent specifies which component in B should use the parceled object and what this component should do with this object. That is, the application initiating

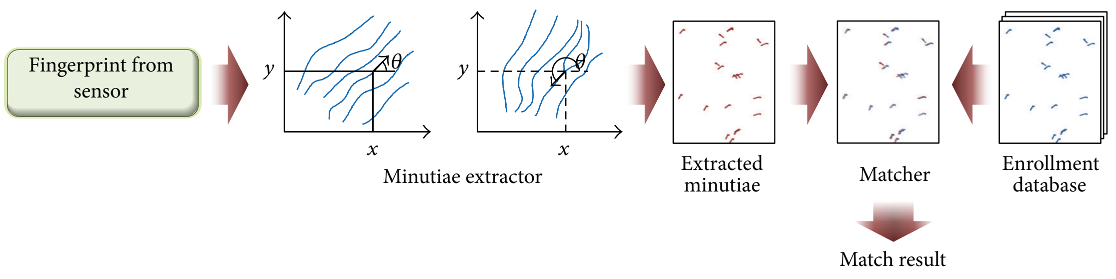


FIGURE 2: Matching of fingerprints (modified from Box 1 in [6]).

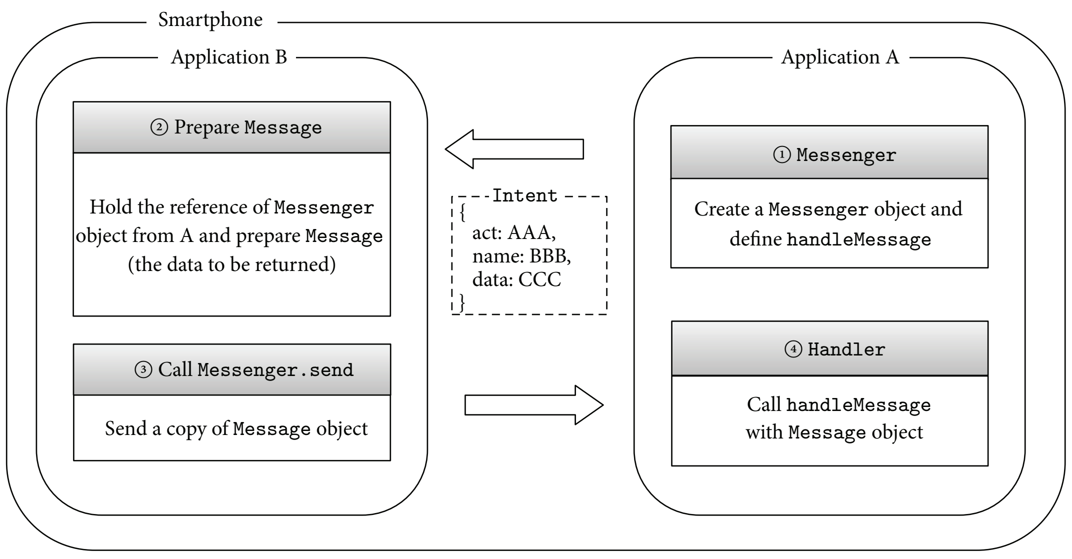

Smartphone
Application B
Application A
② Prepare Message
Hold the reference of Messenger object from A and prepare Message (the data to be returned)
③ Call Messenger.send
Send a copy of Message object
Intent
{
act: AAA,
name: BBB,
data: CCC
}
① Messenger
Create a Messenger object and define handleMessage
④ Handler
Call handleMessage with Message object

FIGURE 3: Message-based communication between Android processes.

the communication can designate a specific action the target application should do if only this action is defined in one of the components of the target application.

(2) While the specified component in B is being executed using the parceled object, this component may prepare the data to be returned to A, if required.

(3) The component then calls a public method of parceled object, `Messenger.send`, after setting the data to be returned, that is, a `Message` object, as its parameter. Among the various fields defined in a `Message` object are the `what` and `obj` fields. The `what` field specifies what kind of this message is and the `obj` field stands for data itself.

(4) Next, the `Handler` in the `Messenger` object, which has been used by A to initiate the communication, receives this object, and the `handleMessage` method of the `Handler` utilizes the data contained in the `Message` object.

3. Vulnerability Analysis

The fingerprint recognition service application on a VEGA Secret Note supports three main functionalities, registration, verification, and deletion.

(i) Registration. To register a fingerprint, a user is asked to swipe a fingerprint over the fingerprint sensor. For high reliability, the user should swipe his/her fingerprint multiple times. At the moment when the user's fingerprint is scanned, the scanned fingerprint image is displayed on the screen. See Figure 4.

(ii) Verification. The verification operation is usually used to unlock the smartphone. In this case, the user's task is just to scan his/her finger over the fingerprint sensor on the locked smartphone. The device recognizes the scanned fingerprint and decides whether to permit this user's access based on the matching result. In addition, other applications may request the fingerprint recognition application to activate the verification functionality to verify if the person who attempts to use the application is the legitimate owner of this smartphone.

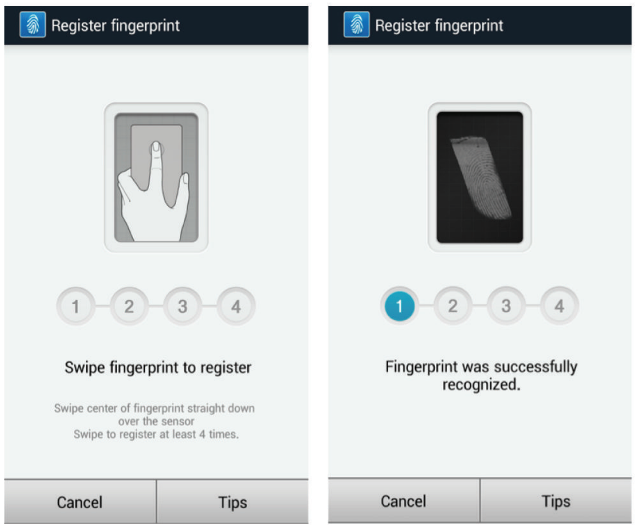

Register fingerprint
1
2
3
4
Swipe fingerprint to register
Swipe center of fingerprint straight down over the sensor
Swipe to register at least 4 times.
Cancel
Tips
Register fingerprint
1
2
3
4
Fingerprint was successfully recognized.
Cancel
Tips

Figure 4: The example procedure for fingerprint registration on VEGA Secret Note.

(iii) Deletion. It is also possible to reset the registered fingerprint by conducting a deletion operation. After unlocking the smartphone by scanning the correct fingerprint, a user may delete the stored fingerprint by scanning his/her fingerprint once again. If this fingerprint matches the registered one, it is deleted from the database.

Because a scanned fingerprint image is displayed on the screen when a user's fingerprint is scanned over the fingerprint sensor, it should be the case that an Android `Bitmap` object in the `View` object related to the fingerprint registration interface is loaded on memory. Therefore, the original fingerprint image may be extracted if we can access the memory location that contains the corresponding `Bitmap` object. Our first attack is to find a way to access the fingerprint image on memory.

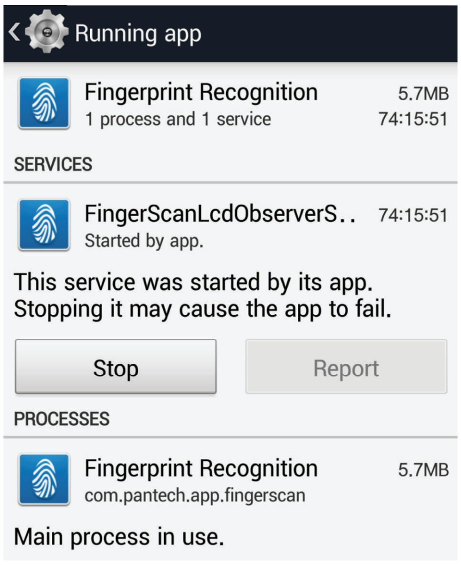
On the other hand, a registered fingerprint should be stored somewhere in nonvolatile memory storage for later use in fingerprint verification. Therefore, we may try to find the location of the stored template and restore the original minutia points. Our second attack is to achieve this objective.

3.1. Reverse Engineering of Fingerprint Service Mechanism. First of all, it is important to know where the binary code of the fingerprint recognition service application is located in flash memory. To find this location, we examined the list of running applications when the fingerprint service application is running as shown in Figure 5. To double-check, we also examined the result of the execution of a `ps` command through *Android Debug Bridge (adb)* [14]. As a result, we successfully identified application `com.pantech.app.fingerscan`. The next step was to extract the Android package file of this application for analysis. To this end, we ran an *adb* shell on a PC and tried to extract the package file using the backup functionality of *adb*, that is, by executing `adb backup -apk com.pantech.app .fingerscan`. After acquiring root user permission through rooting, we analyzed the package file. We remark that the root user permission is only required for the analysis of the application package file, but not all actual attacks such as the fingerprint disclosure attack explained in Section 3.2 require this permission.

Next, by analyzing the package file using a few tools such as *dex2jar 0.0.9.15* and *jd-gui 0.3.7*, we found out that this application uses JNI (Java Native Interface) to use the low-level functions implemented in a C++ library for fingerprint management, and we identified the path of this library loaded by the application. As a result, we successfully extracted an Android framework file, `framework.odex` (and its corresponding `framework.jar`), and a shared library file, `libfpc1080_jni.so`. We used `framework.odex` to understand the interaction between `class.dex` and `libfpc1080_jni.so`. For the analysis of `framework.odex`, we used a disassembler, *baksmali 2.0.3*.

The above implementation stack is summarized in Figure 6. According to our analysis, the library file, `libfpc1080_jni.so`, which is an ARM-based dynamic linking library, contains the core routines for fingerprint authentication, in particular, fingerprint image processing. Therefore, in order to find attack vectors against fingerprint authentication service, we traced a source code decompiled from `libfpc1080_jni.so` line by line. The detailed operation mechanism of this library will be explained in Sections 3.2 and 3.3.

3.2. Acquisition of Original Fingerprint Image through a Malicious Application. As briefly explained in the introductory part of this section, a scanned fingerprint image is displayed on the screen when a user's fingerprint is scanned. Therefore, an Android graphic data object such as a `Bitmap` object should be generated to show a fingerprint image while the fingerprint was being enrolled. We tried to find the code segments referring to this object in the decompiled source code, and, eventually, we successfully identified the following


Running app

Fingerprint Recognition 5.7MB
1 process and 1 service 74:15:51

SERVICES

FingerScanLcdObserverS.. 74:15:51
Started by app.

This service was started by its app.
Stopping it may cause the app to fail.

Stop | Report

PROCESSES

Fingerprint Recognition 5.7MB
com.pantech.app.fingerscan

Main process in use.

FIGURE 5: List of Android applications which are running currently.

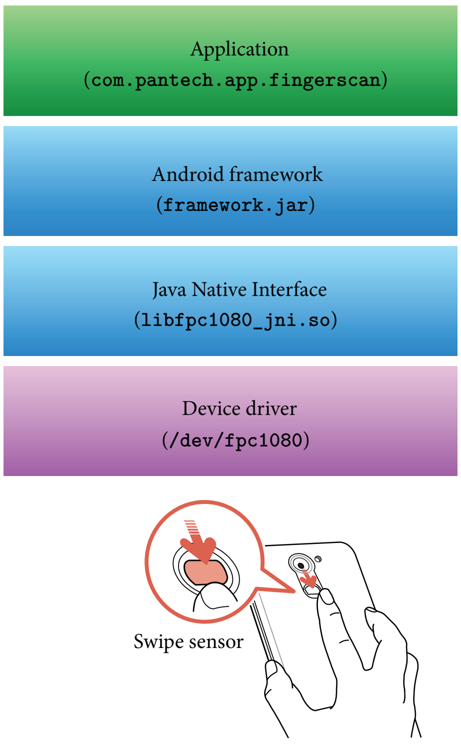


FIGURE 6: Implementation stack of fingerprint recognition service in VEGA Secret Note (the figure of a smartphone was adopted from [15]).

three locations, (1) setting up a `View` object for the fingerprint image during fingerprint registration; (2) getting a fingerprint image from the sensor during both fingerprint registration and verification; and (3) responding to an authentication request from an external application. Note that, in the last case, the external application may directly handle the `Bitmap` object corresponding to the fingerprint image if the response from the service application includes this object. This finding motivated us to analyze the communication procedure between the service application and the external client application requesting this service.

Figure 7 shows the analyzed result for the organization of the fingerprint authentication service in VEGA Secret Note. As shown in this figure, fingerprint authentication is conducted as follows.

(1) First, a client application A who wishes to use the fingerprint authentication service sends an

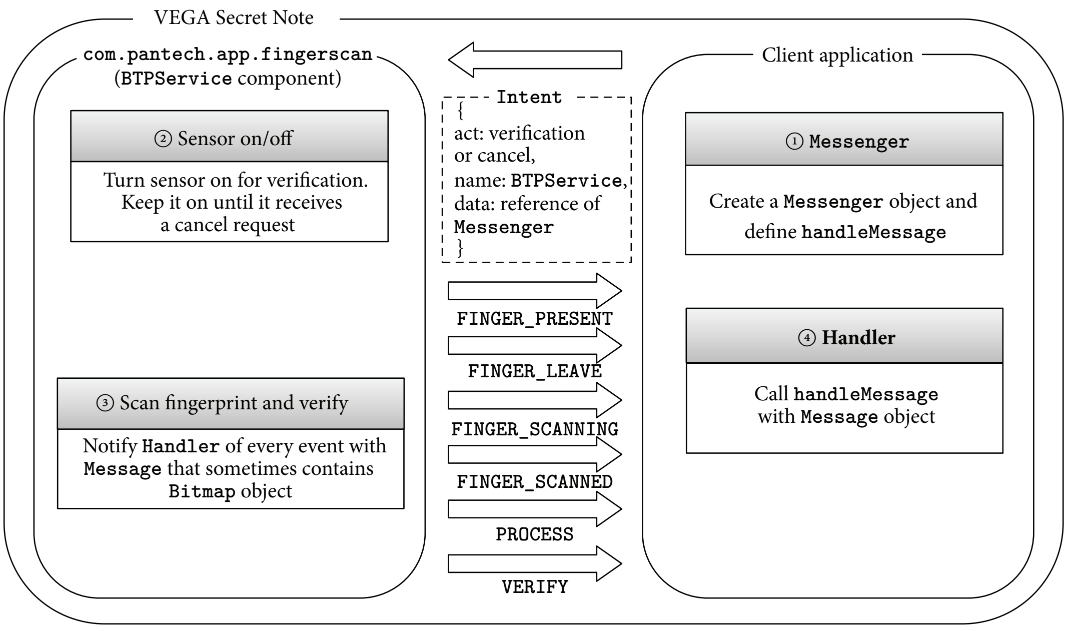

VEGA Secret Note
com.pantech.app.fingerscan (BTPService component)
② Sensor on/off
Turn sensor on for verification. Keep it on until it receives a cancel request
③ Scan fingerprint and verify
Notify Handler of every event with Message that sometimes contains Bitmap object
Intent
{ act: verification or cancel, name: BTPService, data: reference of Messenger }
FINGER_PRESENT
FINGER_LEAVE
FINGER_SCANNING
FINGER_SCANNED
PROCESS
VERIFY
Client application
① Messenger
Create a Messenger object and define handleMessage
④ Handler
Call handleMessage with Message object

FIGURE 7: Fingerprint authentication service in VEGA Secret Note.

`Intent` object to initiate a communication with the fingerprint recognition service application, `com.pantech.app.fingerscan`. For this purpose, A does not have any data to share. The `Intent` contains the name of the target component, `BTPService`, and its requested action, `btp.intent.action.verification`. This request allows A to occupy the fingerprint sensor and prevent another application from using the fingerprint authentication service until `btp.intent.action.cancel` is sent by A.

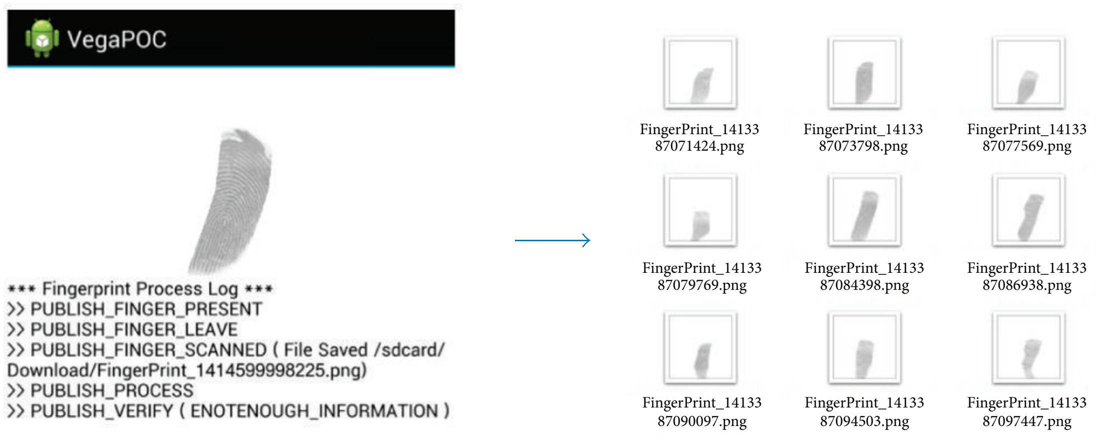
(2) While `BTPService` in `com.pantech.app.fingerscan` is being executed, it turns on the fingerprint sensor and asks the user to scan his/her fingerprint. A `Bitmap` object is defined to contain the scanned fingerprint image.

(3) From the moment that a finger contacts the sensor, `BTPService` notifies A of every event that occurs, which can be one of the following seven events: `FINGER_PRESENT` (the finger touches the sensor), `FINGER_SCANNING` (the sensor is scanning the fingerprint), `FINGER_SCANNED` (the sensor completed a scan), `FINGER_LEAVE` (the finger leaves the sensor), `PROCESS` (the fingerprint verification operation is being done), `VERIFY` (the fingerprint verification operation has been completed), and `IGNORE_NOTIFY` (it seems that this event is not actually used). The procedure to send these notices is as follows. Whenever `BTPService` needs to send a notice, it first creates a new `Message` object and sets the `what` field in this `Message` to one of the constants corresponding to the current state of the sensor, which is defined in the Android framework file, `framework.odex`. When the event is `FINGER_SCANNED`, `BTPService` sets the `obj` field in `Message` to the `Bitmap` object containing the scanned fingerprint image. Finally, `BTPService` calls the function `Messenger.send` defined by A after setting the parameter to its own `Message`.

(4) The `Handler` in `Messenger` receives `Message` from `BTPService`, and then A can utilize the `Bitmap` object in `Message` in the way that its own `Handler.handleMessage` defines.

Our task is now to design a proof-of-concept (PoC) appliction that plays A's role. In addition, this PoC application should contain an event handler function, `Handler.handleMessage`, so that it may export the `Bitmap` object to a standard image file. To achieve this objective, we first analyzed the bitmap configuration of the object and found out that it was `Android.Bitmap.Config.ALPHA_8`. This constant is defined in the `Android.Bitmap.Config` class and it implies that pixels are stored as a single translucency channel. Next, we tried to use `Bitmap.compress` which is a typical method to export a bitmap image from memory to a file. Unfortunately, however, the `Bitmap.compress` method did not support the exportation of an image configured with `Android.Bitmap.Config.ALPHA_8`. To solve this problem, we converted the image into a 32-bit color image using the `Android.Bitmap.Config.ARGB_8888` configuration. However, because the original fingerprint is a grey-scale image, we set $R = 0, G = 0$, and $B = 0$, keeping the original alpha value unchanged. Finally, we succeeded in generating a png-format file containing the scanned fingerprint image.

The above attack vector was implemented in our PoC application. Figure 8 demonstrates its execution result. We call this attack a *fingerprint disclosure attack*. It should be


VegaPOC
FingerPrint_14133 87071424.png
FingerPrint_14133 87073798.png
FingerPrint_14133 87077569.png
FingerPrint_14133 87079769.png
FingerPrint_14133 87084398.png
FingerPrint_14133 87086938.png
FingerPrint_14133 87090097.png
FingerPrint_14133 87094503.png
FingerPrint_14133 87097447.png

```
*** Fingerprint Process Log ***
>> PUBLISH_FINGER_PRESENT
>> PUBLISH_FINGER_LEAVE
>> PUBLISH_FINGER_SCANNED ( File Saved /sdcard/
Download/FingerPrint_1414599998225.png)
>> PUBLISH_PROCESS
>> PUBLISH_VERIFY ( ENOTENOUGH_INFORMATION )
```

Figure 8: PoC that stores scanned images to files.

noted that this PoC application may be normally installed if only the smartphone owner accepts this application's request for a few application-level permissions. In other words, the above attack does not require any privilege such as root user permission.

The fingerprint disclosure attack can be viewed as a passive type 1 or type 2 attack explained in Section 2.1, though the actual extraction of the full image is done at the moment when the matcher makes the decision for a matching result. As the next step of this attack, an attacker may forge a fake fingerprint with the obtained fingerprint image to mount an attack fooling an image sensor, that is, an active type 1 attack, or to bypass the image sensor by injecting the disclosed image, that is, an active type 2 attack. Moreover, this fake fingerprint may be used for other environments where a fingerprint is used for authentication. For example, an attacker may unlock the victim's doorlock with a fake fingerprint obtained via a fingerprint disclosure attack against the victim's smartphone.

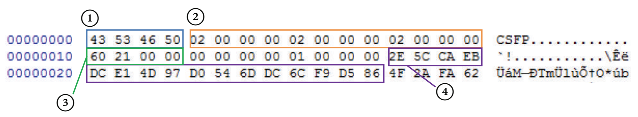
3.3. Extraction of Fingerprint Minutiae from an Encrypted Fingerprint Template. A registered fingerprint should be stored somewhere in nonvolatile memory storage for later use in fingerprint verification. If the fingerprint template is stored as a readable data file, we may try to analyze its structure and get minutiae points. Therefore, our second attack began with finding the location of the stored template file. As explained in Section 3.1, we analyzed the fingerprint application and identified a few essential functions in the library file `libfpc1080_jni.so` dedicated for fingerprint matching. In addition, by analyzing the code segment where the file storing the fingerprint data is accessed, we found out that the name and path of this file were hard-coded irrespective of a specific device. The file name was `csfp.tpl`.

According to our analysis, the file `csfp.tpl` starts with a 48-byte header and the remainder is a main body containing fingerprint data. To be precise, as shown in Figure 9, the header contains a 4-byte identifier (signature) which stands for `CSFP`, a 12-byte field which stands for file version, a 4-byte field which stands for file size, a 16-byte field which stands for MD5 checksum, and some additional data. This structure was found by tracing the header updating function in the library file.

The main body starting at the 49th byte is not in a readable form, but it is encrypted. By analyzing the point in `libfpc1080_jni.so` when a user's fingerprint template data is stored in `csfp.tpl`, we could decode the encryption logic. According to our analysis, the encryption of a template is performed using the CBC mode of AES [16, 17] with a 256-bit key and a 128-bit IV. We also found out that most routines used for this encryption procedure resemble those of OpenSSL [18], which was helpful for our analysis. The key and IV are generated using very simple `for` statements without involving any randomness. As a result, the key and IV are fixed as 0x00010203...1F and 0x00010203...0F, respectively, and all devices use the same key and IV.

We remark that the fact that all devices use the same key may be a critical issue. If the `csfp.tpl` in device A is copied and overwritten to another template file with the same name in another device B, then B will accept the fingerprint of the owner of A. This implies that virtually any attacker may bypass the fingerprint authentication. We will call this attack a *template replacement attack*, which may be regarded as an active type 6 attack according to the taxonomy in Section 2.1. Below is a practical scenario where this attack may be a potential threat. If an attacker has a temporary access to B when B is temporarily unattended (e.g., the owner of B may go to a restroom leaving his/her smartphone on the desk.), the attacker may inject his/her own `csfp.tpl` into B after rooting B. The attacker then may execute a financial transaction which is approved with fingerprint verification, for example, a payment on PayPal [1]. If time is sufficient, the attacker may recover the original template file and unroot B, which prevents the owner of B from recognizing what happened with his/her device.

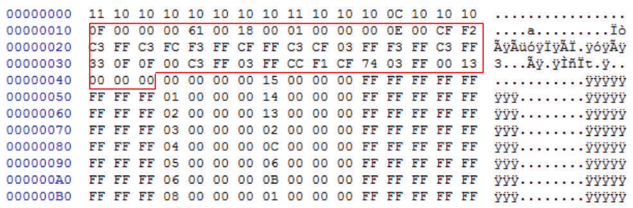
Because the encryption mechanism and its key information have been analyzed, our next step was to decrypt the encrypted template file and obtain the information about the original template, which we call a *template restoration attack*. We wrote a C code to perform AES decryption using



```
00000000  43 53 46 50 02 00 00 00 02 00 00 00 02 00 00 00  CSFP............
00000010  60 21 00 00 00 00 00 00 01 00 00 00 2E 5C CA EB  `!..........\Êë
00000020  DC E1 4D 97 D0 54 6D DC 6C F9 D5 86 4F 2A FA 62  ÜáM—ÐTmÜlùÕ†O*úb
```

① 🟦 0x00 ~ 0x03: file type (CSFP)
② 🟧 0x04 ~ 0x0F: file version
③ 🟩 0x10 ~ 0x13: file size
④ 🟪 0x1C ~ 0x2B: MD5 checksum of body

Figure 9: Header format of `csfp.tpl`.



```
00000000  11 10 10 10 10 10 10 10 11 10 10 10 0C 10 10 10  ................
00000010  0F 00 00 00 61 00 18 00 01 00 00 00 0E 00 CF F2  ....a.........Ïò
00000020  C3 FF C3 FC F3 FF CF FF C3 CF 03 FF F3 FF C3 FF  ÃÿÃüóÿÏÿÃÏ.ÿóÿÃÿ
00000030  33 0F 0F 00 C3 FF 03 FF CC F1 CF 74 03 FF 00 13  3...Ãÿ.ÿÌñÏt.ÿ..
00000040  00 00 00 00 00 00 00 15 00 00 00 FF FF FF FF FF  ...........ÿÿÿÿÿ
00000050  FF FF FF 01 00 00 00 14 00 00 00 FF FF FF FF FF  ÿÿÿ........ÿÿÿÿÿ
00000060  FF FF FF 02 00 00 00 13 00 00 00 FF FF FF FF FF  ÿÿÿ........ÿÿÿÿÿ
00000070  FF FF FF 03 00 00 00 02 00 00 00 FF FF FF FF FF  ÿÿÿ........ÿÿÿÿÿ
00000080  FF FF FF 04 00 00 00 0C 00 00 00 FF FF FF FF FF  ÿÿÿ........ÿÿÿÿÿ
00000090  FF FF FF 05 00 00 00 06 00 00 00 FF FF FF FF FF  ÿÿÿ........ÿÿÿÿÿ
000000A0  FF FF FF 06 00 00 00 0B 00 00 00 FF FF FF FF FF  ÿÿÿ........ÿÿÿÿÿ
000000B0  FF FF FF 08 00 00 00 01 00 00 00 FF FF FF FF FF  ÿÿÿ........ÿÿÿÿÿ
```

Figure 10: Decrypted content of `csfp.tpl` (extended from Figure 1 in [19]).

Table 1: Size and meaning of each field in a minutia in the decrypted `csfp.tpl` [19].

Index	Size (B)	Meaning
0x00	4	Node (minutia) id
0x04	2	x coordinate
0x06	2	y coordinate
0x08	4	Duplicate count (weight)†
0x0c	2	θ $(0 \leq \theta \leq 47)$
0x0e	32	Additional information of node
0x2e	1	Minutia type
0x2f	4	(Distance to next minutia in bytes)/16

†A weight is the number of occurrences of the same minutia point when a fingerprint is scanned multiple times to enhance accuracy.

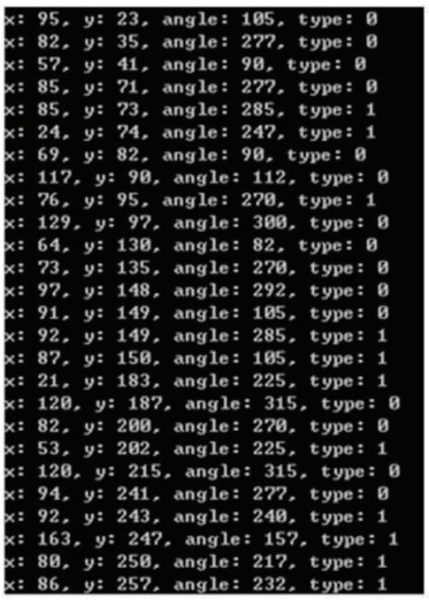
the procedures provided by OpenSSL [18]. For decryption, we used the fixed IV vector and key revealed in the above analysis. This C code reads the `csfp.tpl` as input and outputs the decrypted data. An example result is shown in Figure 10. At a glance, it does not seem to be feasible to identify the meaningful fields such as x, y, and θ and ridge types from the decoded binary data in Figure 10. However, we could identify x and y coordinates by analyzing the code segments in `libfpc1080_jni.so` which refer to the coordinates. In addition, we could identify θ by analyzing the code segment for rotation. According to our analysis, θ was represented in a metric system where a full rotation is defined as an integer, 48. In Figure 10, the highlighted region, which is 51 bytes long, stands for a single minutia point. The meaning of each field constituting this minutia is summarized in Table 1.

We were able to restore each and every minutia point in the original fingerprint template from the encrypted file, `csfp.tpl`, by using our PoC code. This result is shown in Figure 11. In the left-hand part of this figure, we enumerated restored points. These points can be graphically expressed using the convention in the literature. The result is presented in the right-hand part of Figure 11.

The restored template may be used to reconstruct the original fingerprint image [9, 20]. The reconstructed image can be used to forge a fake fingerprint to fool an image sensor. Moreover, the reconstructed image can be used for the same purposes as those of the fingerprint disclosure attack in Section 3.2, for example, to unlock a door. In some sense, however, the reconstruction through a template restoration attack may be a more severe threat because there is no need to execute a malicious application and to wait until the user scans his/her fingerprint.

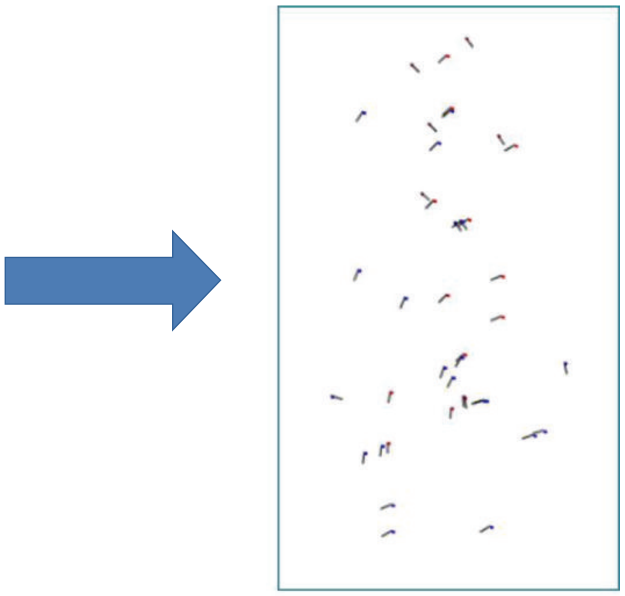
4. Discussion on Countermeasures

The reason why our fingerprint disclosure attack was successful was that `com.pantech.app.fingerscan` has a component which provides the client application (the malicious application, in our case) with the bitmap image of a fingerprint. Because fingerprint recognition service should be available to client applications, it seems inevitable to allow these applications to call some component providing fingerprint recognition functionalities. However, this component should be designed so that it returns only the result of fingerprint authentication, not the original image itself. Therefore, we suggest that the `BTPService` component should be replaced with such a new component. We also remark that a customized permission `com.pantech.fingerprint.security` for fingerscan recognition has already been defined in the manifest in VEGA Secret Note. If an external application obtains this permission, it can access the fingerprint recognition service including the original fingerprint image, whose property was used by our PoC program. Therefore, we suggest that the permissions for the fingerprint recognition service should be subdivided



```
x: 95, y: 23, angle: 105, type: 0
x: 82, y: 35, angle: 277, type: 0
x: 57, y: 41, angle: 90, type: 0
x: 85, y: 71, angle: 277, type: 0
x: 85, y: 73, angle: 285, type: 1
x: 24, y: 74, angle: 247, type: 1
x: 69, y: 82, angle: 90, type: 0
x: 117, y: 90, angle: 112, type: 0
x: 76, y: 95, angle: 270, type: 1
x: 129, y: 97, angle: 300, type: 0
x: 64, y: 130, angle: 82, type: 0
x: 73, y: 135, angle: 270, type: 0
x: 97, y: 148, angle: 292, type: 0
x: 91, y: 149, angle: 105, type: 0
x: 92, y: 149, angle: 285, type: 1
x: 87, y: 150, angle: 105, type: 1
x: 21, y: 183, angle: 225, type: 1
x: 120, y: 187, angle: 315, type: 0
x: 82, y: 200, angle: 270, type: 0
x: 53, y: 202, angle: 225, type: 1
x: 120, y: 215, angle: 315, type: 0
x: 94, y: 241, angle: 277, type: 0
x: 92, y: 243, angle: 240, type: 1
x: 163, y: 247, angle: 157, type: 1
x: 80, y: 250, angle: 217, type: 1
x: 86, y: 257, angle: 232, type: 1
```

(a)



(b)

Figure 11: Restored minutiae points and their graphical representation using x, y, and θ (reproduction of Figure 2 in [19]).

into several ones and the permission to access the original fingerprint image should be selectively given.

Regarding the template replacement attack, using the same key and IV vector on all devices is not recommended because an attacker can thwart the authentication test by overwriting the template file in the target device with that extracted from another device. Therefore, we suggest that a distinct key and a distinct IV should be used for each device. Then, even the same template will be transformed into a distinct ciphertext in each device, which implies that simply overwriting a template file does not work. For automatic generation of these device-dependent values, hardware characteristics such as a processor identifier could be used. For example, ARM Cortex-A series including Cortex-A7 processor embedded in Qualcomm Snapdragon CPU of VEGA Secret Note provide the device ID number in the `Primary part number` field in the `Main ID register (MIDR)` [21]. However, this patch does not completely solve the problem, given that the key generation and encryption procedures are easily recognizable by reverse engineering the library file. Thus, an attacker can still mount a template restoration attack that we demonstrated in Section 3.3, even if she/he cannot use the same template file. That is, a forged `csfp.tpl` file encrypted using the key of the target device and the minutiae points from a source device will let the owner of the source device be authenticated by the target device. Therefore, it would be desirable to design a fingerprint recognition procedure so that an extracted template should be useless for other devices even after being properly decrypted. To achieve this goal, cancelable fingerprints [10, 22] may be adopted. By using a noninvertible transform based on device-dependent parameters, an original template may be transformed into a new template before encryption. This transform can be viewed as a kind of error-tolerant one-way hash. Then, the comparison of templates for fingerprint verification is done over a transformed domain as in the case of traditional password authentication where passwords are compared over the hashed domain. Even after an attacker extracts the transformed template, the information about the original minutiae points is protected thanks to the one-way property of the noninvertible transform. This approach prevents a template restoration attack, though it cannot prevent a fake template synthesized according to the rules reverse-engineered from the target device. Therefore, the logic to generate a key and an IV vector should be obfuscated and made hard to be analyzed. For this purpose, we may use the well-known off-the-shelf obfuscation tools.

We may consider a more essential solution including hardware-based isolation technologies such as ARM TrustZone [23]. These techniques might be adopted for secure storage of fingerprint data and isolated execution of fingerprint recognition service.

We finally remark that there is an automated tool to analyze the cryptographic misuse in Android mobile applications [24]. This system only supports the analysis of Dalvik bytecodes, but applications such as our target application that invoke cryptographic primitives from native code cannot be analyzed.

5. Conclusion

By reverse engineering a fingerprint recognition service application, we have identified a few vulnerabilities in the fingerprint recognition service of VEGA Secret Note and demonstrated actual attacks against this service. The technical flaws we have found in this device are a common trap that developers may fall into. To mitigate these vulnerabilities, we suggested possible countermeasures which may be implemented using well-known techniques in the literature. We expect that the findings we obtained through our analysis may be used as a general guideline to design a secure biometric verification service on smartphones. However, the proposed countermeasures cannot prevent all attacks, for example, a fake template synthesized using the reverse-engineered rules and keys of the target device. Therefore, it would be an important future research issue to develop a more robust countermeasure. In addition, it would be a good research issue to verify whether other smartphones such as Galaxy series and iPhones equipped with fingerprint recognition service are vulnerable or not to the attacks described in this paper.

Disclosure

Most of this research was done when Young-Hoo Jo was a student in Inha University. A preliminary version of this paper was presented at FutureTech 2015 [19].

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (Grant no. 2014R1A1A2058514).

References

[1] PayPal, "Pay faster with your fingerprint," 2014, https://www.paypal-pages.com/samsunggalaxys5/us/index.html.

[2] Pantech, "Pantech unveils VEGA LTE-A, world's first LTE-A with fingerprint recognition and rear touch," 2013, http://www.pantech.co.kr/en/board/reportBoardView.do?seq=5870&bbsID=report&ulcd=KO.

[3] M. Bishop, *Computer Security: Art and Science*, Addison-Wesley, Boston, Mass, USA, 2003.

[4] N. K. Ratha, J. H. Connell, and R. M. Bolle, "Enhancing security and privacy in biometrics-based authentication systems," *IBM Systems Journal*, vol. 40, no. 3, pp. 614–634, 2001.

[5] A. K. Jain, K. Nandakumar, and A. Nagar, "Biometric template security," *EURASIP Journal on Advances in Signal Processing*, vol. 2008, Article ID 579416, 17 pages, 2008.

[6] A. K. Jain, "Technology: biometric recognition," *Nature*, vol. 449, no. 7158, pp. 38–40, 2007.

[7] ISO/IEC, "Information technology—biometric data interchange formats—part 2: finger minutiae data," ISO/IEC International Standard 19794-2, 2011.

[8] ANSI and INCITS, "American National Statandard for information technology—finger minutiae format for data interchange," ANSI INCITS 378-2009, 2009.

[9] R. Cappelli, A. Lumini, D. Maio, and D. Maltoni, "Fingerprint image reconstruction from standard templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1489–1503, 2007.

[10] N. K. Ratha, S. Chikkerur, J. H. Connell, and R. M. Bolle, "Generating cancelable fingerprint templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 4, pp. 561–572, 2007.

[11] T.-Y. Jea and V. Govindaraju, "A minutia-based partial fingerprint recognition system," *Pattern Recognition*, vol. 38, no. 10, pp. 1672–1684, 2005.

[12] Android Developers, android.os.Messenger, 2014, http://developer.android.com/reference/android/os/Messenger.html.

[13] Android Developers, android.content.Intent, 2014, http://developer.android.com/reference/android/content/Intent.html.

[14] Android Debug Bridge, 2014, http://developer.android.com/tools/help/adb.html.

[15] Pantech VEGA Service, 2014, http://www.pantechservice.co.kr.

[16] National Institute of Standards and Technology, *Recommendation for Block Cipher Modes of Operation*, NIST Special Publication 800-38A, National Institute of Standards and Technology, Gaithersburg, Md, USA, 2001.

[17] NIST Federal Information Processing Standards Publication 197, *Advanced Encryption Standard (AES)*, 2001.

[18] OpenSSL, "The Open Source Toolkit for SSL/TLS," 2014, http://www.openssl.org/.

[19] Y.-H. Jo, S.-Y. Jeon, J.-H. Im, and M.-K. Lee, "Vulnerability analysis on smartphone fingerprint templates," in *Advanced Multimedia and Ubiquitous Engineering*, vol. 354 of *Lecture Notes in Electrical Engineering*, pp. 71–77, Springer, Berlin, Germany, 2016.

[20] J. Feng and A. K. Jain, "Fingerprint reconstruction: from minutiae to phase," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 2, pp. 209–223, 2011.

[21] ARM, *Cortex-A7 MPCore Technical Reference Manual*, Revision r0p3, 2012.

[22] D. Moon, J.-H. Yoo, and M.-K. Lee, "Improved cancelable fingerprint templates using minutiae-based functional transform," *Security and Communication Networks*, vol. 7, no. 10, pp. 1543–1551, 2014.

[23] ARM, "TrustZone," 2014, http://www.arm.com/products/processors/technologies/trustzone/index.php.

[24] M. Egele, D. Brumley, Y. Fratantonio, and C. Kruegel, "An empirical study of cryptographic misuse in Android applications," in *Proceedings of the ACM SIGSAC Conference on Computer & Communications Security (CCS '13)*, pp. 73–83, ACM, Berlin, Germany, November 2013.

Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 6707524, 11 pages
http://dx.doi.org/10.1155/2016/6707524

Research Article

Function-Oriented Mobile Malware Analysis as First Aid

Jae-wook Jang and Huy Kang Kim

Graduate School of Information Security, Korea University, Seoul 136-713, Republic of Korea

Correspondence should be addressed to Huy Kang Kim; cenda@korea.ac.kr

Received 3 November 2015; Accepted 2 February 2016

Academic Editor: Seung Yang

Recently, highly well-crafted mobile malware has arisen as mobile devices manage highly valuable and sensitive information. Currently, it is impossible to detect and prevent all malware because the amount of new malware continues to increase exponentially; malware detection methods need to improve in order to respond quickly and effectively to malware. For the quick response, revealing the main purpose or functions of captured malware is important; however, only few recent works have attempted to find malware's main purpose. Our approach is designed to help with efficient and effective incident responses or countermeasure development by analyzing the main functions of malicious behavior. In this paper, we propose a novel method for function-oriented malware analysis approach based on analysis of suspicious API call patterns. Instead of extracting API call patterns for malware in each family, we focus on extracting such patterns for certain malicious functionalities. Our proposed method dumps memory sections where an application is allocated and extracts suspicious API sequences from bytecode by comparing with predefined suspicious API lists. By matching API call patterns with our functionality database, our method determines whether they are malicious. The experiment results demonstrate that our method performs well in detecting malware with high accuracy.

1. Introduction

As mobile devices become widely spread, highly well-crafted mobile malware has become one of the most dangerous threats to the mobile computing environment. According to a report by McAfee [1], the total amount of mobile malware has continued its steady climb as it broke 6 million in the 4th quarter of 2014 and increased by 14% over the 3rd quarter of the same year. Moreover, approximately 0.8 million new malware families and variants were reported to appear in the same quarter. Antivirus (AV) vendors analyze a large amount of malware samples daily, and prevent them from spreading widely; in other words, offenders and defenders have been waging an endless battle.

Despite the efforts of AV vendors, detecting and preventing all malware become difficult because malware tends to evolve over time. In particular, the number of malware samples that embed antimalware analysis methods, such as Android packer (e.g., APK Protect [2] and Bangcle [3]), dynamic loading, and `dex` encryption, has increased steadily [4]. Android packers enable encrypting an original `dex` bytecode, decrypting the `dex` bytecode on memory section at runtime, and then executing the `dex` bytecode via `DexClassLoader`. Under these circumstances, it is impossible to detect and prevent all malware and its variants, especially malware that embeds antianalysis techniques.

As shown in Figure 1, based on their purpose or object, we categorize malware analysis approaches into three types: autopsy-oriented approach, function-oriented approach, and symptom-oriented approach. The autopsy-oriented approach aims to represent how malware conducts malicious behavior and consists of traditional static and dynamic analysis. Static analysis aims to examine the binary code in order to determine its characteristics without executing it. Many static analysis approaches in previous works have failed to parse meaningful footprints from malware samples that embed these antianalysis techniques [5–9]; the malware that embeds antianalysis techniques leads malware analysts to problems. On the other hand, dynamic analysis aims to provide methods for extracting the unique behavioral patterns of malware. Dynamic analysis addresses obfuscation, packing, and encryption methods because all methods are eliminated during execution of a malicious application [10–13]. However, dynamic analysis is performed only on a part of the executed

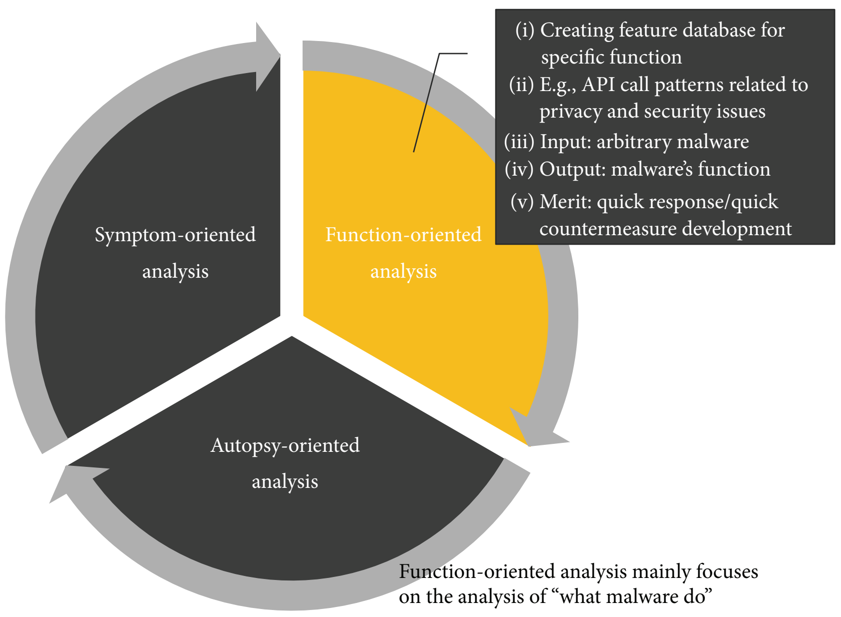

(i) Creating feature database for specific function
(ii) E.g., API call patterns related to privacy and security issues
(iii) Input: arbitrary malware
(iv) Output: malware's function
(v) Merit: quick response/quick countermeasure development
Symptom-oriented analysis
Function-oriented analysis
Autopsy-oriented analysis
Function-oriented analysis mainly focuses on the analysis of "what malware do"

Figure 1: Categorization of malware analysis approaches.

application, and malicious behavior should be executed during analysis time; malware analysts might believe that malware conducts malicious behavior during analysis time. Most of the previous works have presented the traditional autopsy-oriented approach. However, the autopsy-oriented approach has a limitation in estimating malware's destructive power or influence on our lives because it is only dedicated to classifying malware samples into the most similar groups.

On the other hand, the function-oriented approach creates a feature database for specific functionalities. This approach allows efficient and effective incident responses or countermeasure development by analyzing the functionalities related to malicious behavior. Moreover, the function-oriented approach has evolved over time to become the symptom-oriented approach. The symptom-oriented approach, known as behavior analysis or reputation-based analysis, allows malware analysts to utilize the threat intelligence gathered by security communities. This approach prevents widespread dissemination of computer crime, although relevant signatures do not yet exist. When all static analysis and dynamic analysis methods fail, behavior observation can be useful for estimating a given application's maliciousness.

Tam et al. [14] manually inspected various malware samples from repositories such as contagion and Android Malware Genome Project. They classified malicious behaviors into six groups according to behavior patterns: calling, sending SMS, network access, retrieving personal information, altering filesystem, and executing external applications. We also manually examined malware samples we had and the extracted API call patterns for certain malicious functionality; since some behavior patterns are not found in our dataset, we leave them out. Few recent works focus on the fact that malware in different categories can have common API call sequences when the main purpose or functionalities are the same [15]. These approaches help detect malware efficiently by analyzing the core behavior functionalities of malware samples. Because modern malware has complex functionalities, traditional similarity matching-based malware classification can miss the malware's purpose if such malware has a critical functionality part that is much smaller than its noncritical functionalities. The critical functionality we define is the main target of malware attack, whereas the noncritical functionalities we define are additional malicious behaviors for concealing the critical functionality. For example, malware authors can mostly reuse known malware code and insert small pieces of code to cause intentionally analysts to lose focus. In addition, an automated analysis tool is based on similarity matching only reports that the given malware is similar to known malware. Therefore, in this case, the malware's critical functionality (e.g., deleting MBR) cannot be detected quickly. Without determining the malware authors' intent, AV vendors can fail to analyze significant malicious footprints or behavior patterns; they only concentrate on extracting fragmental signatures for malware detection or classification (Section 3). If excluding the noncritical functionalities purposely, AV vendors are limited to capture influentially malicious functionalities and they are prone to fail occasionally to detect malware.

To overcome the drawbacks of the traditional autopsy-oriented approach, we propose the function-oriented approach to more efficiently determine what a given malware wants to do. For this, malware analysts need to know what the malware authors want to obtain and how they trigger malicious behavior. Such intent of the malware authors can provide malware analysts with vital clues for identifying malware, and malware analysts can determine the malware authors' attack patterns from them and leverage these patterns as detection rules. To achieve the end goals, we introduce a hybrid antimalware system that combines static analysis and dynamic analysis and focuses on extracting malicious functionalities related to the malware authors' intent. Our approach is not an exhaustive search for parsing meaningful footprints, but a fast primary screening.

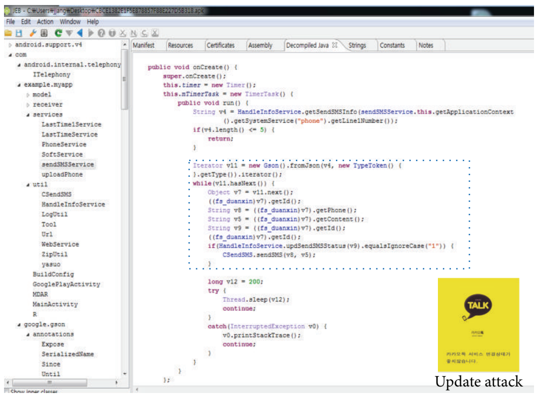
Despite malware that embeds various antianalysis methods, the ndroid platform decrypts (or unpacks) all malware on the memory section. Our system catches the moment when the `odex` bytecode is loaded onto the memory section [16] and leverages artifacts for malware analysis. In particular, our proposed system runs a malicious application on an emulator, dumps the meaningful volatile memory section (`odex` bytecode) where a target application is allocated, and extracts suspicious APIs from the `odex` bytecode in order to overcome challenging issues (i.e., addressing malware embedding antianalysis methods). Instead of extracting all API call patterns for malware in each family, as most previous works have done, we focus on extracting API call patterns for certain malicious functionality. As comparing these with the predefined suspicious API list in detection ruleset, our system enables the effective and efficient malware detection.

To this end, the contributions of this work are as follows. First, as first aid, our system can respond quickly to many species of malware samples by analyzing the functionalities related to malicious behavior. Second, we propose a hybrid malware detection method coupled with a memory acquisition method to enhance detection accuracy. Finally, with a recent real-world malware dataset, we empirically study whether the proposed approach generates superior results.

The rest of this paper is organized as follows. In Section 2, the related work is reviewed. In Section 3, we discuss the limitations of the autopsy-oriented approach. In Section 4, we present our methodology and experiment. In Section 5, we discuss the limitations of our proposed method, and we conclude with Section 6.

2. Related Work

Most of the previous works falls into two broad types: static and dynamic analysis. They have mainly presented the traditional autopsy-oriented approach. In static analysis, code patterns of malware samples through reverse engineering are used for a signature for malware detection. On the other hand, dynamic analysis aims to extract unique behavior patterns of each malware sample based on its behavior. Enck et al. [10] proposed the Kirin security service, a lightweight antimalware system, to mitigate malware at installation time. Kirin inspected the requested permissions in a manifest file and determined whether malicious activities were executed by comparing them with predefined rules. Pearce et al. [12] proposed a privacy information antileakage system, AdDroid, that separated permissions related to advertisement from other permissions. However, AdDroid was limited to responding to the majority of malware that does not spread via advertisements. Peng et al. [6] proposed probabilistic models to represent risk scoring, ranging from the simple Naïve Bayes to advanced mixture models. Their proposed system computed a risk score for each application based on the requested permissions in a manifest file and determined whether malicious activities were executed. Wang et al. [7] proposed DroidRisk, a quantitative risk assessment model of permissions. By quantifying the risk level of each application, they presented a reliable risk indicator that could be generated to warn of potential malicious activities. However, application developers tended to declare an unnecessary number of requested permissions in a manifest file, despite the application not requiring them at all; therefore, these methods were prone to high false positives. Recently, these methods have also considered the requested permission for using each API method in reality by analyzing an API call graph. However, these detection methods are not efficient for classifying benign applications as benign because the relevant rule sets merely focus on detecting malware. Zhang et al. [13] proposed VetDroid, a dynamic analysis approach for extracting the sensitive behavior of a given application. By leveraging the API-related permission table, VetDroid completely identified all possible API-related permission uses. However, their methods are not effective for identifying benign applications because the relevant rules only focus on detecting malware, thus causing large false alarms. Arp et al. [5] proposed DREBIN, which considered permissions and sensitive APIs as features. DREBIN extracted features from both a manifest file and bytecode and then identified malware automatically. Yang et al. [8] proposed DroidMiner, which automatically mined malicious behavior from a behavioral graph. DroidMiner considered the frequency of APIs and the dependencies between multiple sensitive API functions. Zhang et al. [9] proposed DroidSIFT for malware classification based on a weighted contextual API dependency graph. To prevent malware from spreading, DroidSIFT exploited graph similarity metrics for malware detection. Zhou et al. [17] proposed DroidRanger, a permission and heuristic-based method. Wu et al. [18] proposed DroidMat, a feature-based malware detection method. DroidMat chose the requested permissions, intent, and API calls as feature vectors and characterized the malicious behavior of each application. DroidMat used K-NN and K-means classifiers to determine whether a given application was malicious. Zheng et al. [19] introduced a signature-based analytic system, DroidAnalytics, to automatically collect malware, generate signatures for malicious application, and identify malicious logics to the opcode level. Moreover, DroidAnalytics easily discovered repackaged applications by estimating the similarity score. Deshotels et al. [20] proposed DroidLegacy, an automated method for extracting each malware signature. DroidLegacy identified repackaged malware by constructing signatures using API calls. DroidLegacy partitioned an APK into loosely coupled modules and compared the API calls of each module with the signature of each family. The API calls invoked by each malicious module were transformed into a signature for malware analysis. Lee et al. [21] proposed a malware detection method by leveraging signature entries or exactly matched binary patterns to estimate the similarity score. For detecting malware variants, they used the signature entries that consisted of class/method name, character string, and method body in the dex bytecode. However, these approaches have a limitation in parsing meaningful code patterns from an application that embeds antimalware analysis techniques (e.g., packing, dynamic loading, and bytecode encryption).



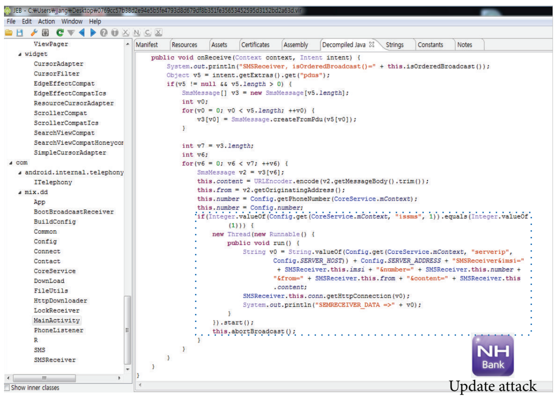
```
public void onCreate() {
    super.onCreate();
    this.timer = new Timer();
    this.mTimerTask = new TimerTask() {
        public void run() {
            String v4 = HandleInfoService.getSendSMSInfo(sendSMSService.this.getApplicationContext
                    ().getSystemService("phone").getLine1Number());
            if(v4.length() <= 5) {
                return;
            }

            Iterator v11 = new Gson().fromJson(v4, new TypeToken() {
            }.getType()).iterator();
            while(v11.hasNext()) {
                Object v7 = v11.next();
                ((fs_duanxin)v7).getId();
                String v8 = ((fs_duanxin)v7).getPhone();
                String v5 = ((fs_duanxin)v7).getContent();
                String v9 = ((fs_duanxin)v7).getId();
                ((fs_duanxin)v7).getId();
                if(HandleInfoService.updSendSMSStatus(v9).equalsIgnoreCase("1")) {
                    CSendSMS.sendSMS(v8, v5);
                }

                long v12 = 200;
                try {
                    Thread.sleep(v12);
                    continue;
                }
                catch(InterruptedException v0) {
                    v0.printStackTrace();
                    continue;
                }
            }
        }
    };
```

SHA256: d5928a3c9bf517daf7f96dd49024d2fe3c243551dc256ed4bb07763ce68bd806

(a) `Fakeinst`



```
public void onReceive(Context context, Intent intent) {
    System.out.println("SMSReceiver, isOrderedBroadcast()=" + this.isOrderedBroadcast());
    Object v5 = intent.getExtras().get("pdus");
    if(v5 != null && v5.length > 0) {
        SmsMessage[] v3 = new SmsMessage[v5.length];
        int v0;
        for(v0 = 0; v0 < v5.length; ++v0) {
            v3[v0] = SmsMessage.createFromPdu(v5[v0]);
        }

        int v7 = v3.length;
        int v6;
        for(v6 = 0; v6 < v7; ++v6) {
            SmsMessage v2 = v3[v6];
            this.content = URLEncoder.encode(v2.getMessageBody().trim());
            this.from = v2.getOriginatingAddress();
            this.number = Config.getPhoneNumber(CoreService.mContext);
            this.number = Config.number;
            if(Integer.valueOf(Config.get(CoreService.mContext, "issms", 1)).equals(Integer.valueOf
                    (1))) {
                new Thread(new Runnable() {
                    public void run() {
                        String v0 = String.valueOf(Config.get(CoreService.mContext, "serverip",
                                Config.SERVER_HOST)) + Config.SERVER_ADDRESS + "SMSReceiver&imsi="
                                 + SMSReceiver.this.imsi + "&number=" + SMSReceiver.this.number +
                                "&from=" + SMSReceiver.this.from + "&content=" + SMSReceiver.this
                                .content;
                        SMSReceiver.this.conn.getHttpConnection(v0);
                        System.out.println("SEMRECEIVER_DATA =>" + v0);
                    }
                }).start();
                this.abortBroadcast();
            }
        }
    }
}
```

SHA256: 0769cc57b38d2e94e5b5fe4793d8d679df8b351fe35653452595d3152bd2a63d

(b) `SMSSend`

FIGURE 2: In-depth analysis results of smishing samples. `Fakeinst` sequentially sends the same SMS message to recipients found in the contacts provider (marked by a dotted blue line box in (a)). `SMSSend` intercepts incoming SMS messages and sends that information to a remote server (marked by a dotted blue line box in (b)). However, F-Secure's descriptions fail to represent the critical behavior of the malware; these textual descriptions only indicate that the malware appears to be an application installer, but it sends SMS messages to premium rate numbers.

The previous works (autopsy-oriented approach) only take whole analysis time for detecting and classifying malware into the most similar groups by comparing a malware sample with the signatures of each malware group. These signatures do not represent malicious functionalities, and when circumventing the signatures of malware groups, malware easily achieves the goal of the malware author. In order to prevent missing vital clues when identifying malware, in this paper, we propose the function-oriented approach based on malware attack patterns that represent the malware attacker's intent. In the following section, we first study the aforementioned limitations of the traditional malware analysis and then review the proposed method.

3. Case Study: Limitations of Autopsy-Oriented Approach

Many previous works on API call sequence analysis have classified malware based on the textual description produced by AV vendors. However, such textual description often fails to capture all malicious behavior. Recently, we found elaborate malware of type Smishing (SMS phishing) and conducted in-depth analysis of the malicious application. As shown in Figure 2(a), the malware `Fakeinst` conducts an update attack: it disguises itself as the most popular chatting application (KaKaoTalk) in Korea. `Fakeinst` copies and sends the same SMS messages to recipients found in the contacts provider. In addition, `Fakeinst` captures incoming SMS messages and steals sensitive information, such as call history and contact information, in order to send that information to a remote server. As shown in Figure 2(b), the malware `SMSSend` conducts an update attack: it disguises itself as a well-known banking application in Korea. Similar to `Fakeinst`, `SMSSend` captures incoming SMS messages and steals sensitive information in order to send that information to a remote server. It also copies and sends the same SMS messages to recipients found in contacts provider.

However, the traditional malware analysis (autopsy-oriented) approach has a limitation in explaining the destructive power or influence on malware on our lives because its concern is on whether a suspicious application is malicious by leveraging known signatures; traditional malware analysis does not explain all the malicious behavioral patterns. Textual descriptions produced by F-Secure only indicate that `Fakeinst` and `SMSSend` appear to be application installers, but such malware sends SMS messages to premium rate numbers [22]. F-Secure's description fails to capture all the malicious behavior of the malware because it only explains fragmentary and broad malicious behavior. By excluding the premium-rate SMS function, the malware sample can cause confusion to AV vendors. Given that AV vendors do not focus on capturing influentially malicious functionalities, they are prone to occasionally fail to detect malware that circumvents their signature. In the aforementioned example, the malware author might evade AV vendors by hiding the function of sending premium-rate SMS. By analyzing the core behavior functionalities of malware samples, malware analysts can detect malware efficiently. In this case study, we only mention the F-Secure [23] case, but this is a common error of all mobile AV vendors.

4. Materials and Methods

4.1. Environment Setup. We performed all experiments in a virtualization environment (VMWare ESX$_i$; http://www.vmware.com/). As shown in Figure 3, our system consists of an analyzer and repository. The analyzer and repository are installed on the server, which runs an Ubuntu 12.04 LTS (64-bit) operating system. We implemented the analyzer using the Python programming language (as scripts) with an Android

Table 1: Malware and benign samples for experiments.

Category	Family	Number of samples	Family	Number of samples
Malware (906)	Smforw	130	None	43
	FakeBank	141	Gidix	40
	FakeKRBank	123	Recal	25
	WroBa	117	SmsSend	25
	Fakeinst	88	TelMan	21
	SmsSpy	79	Helir	12
	MisoSMS	52	Fakeguard	10
Benign		1,776		

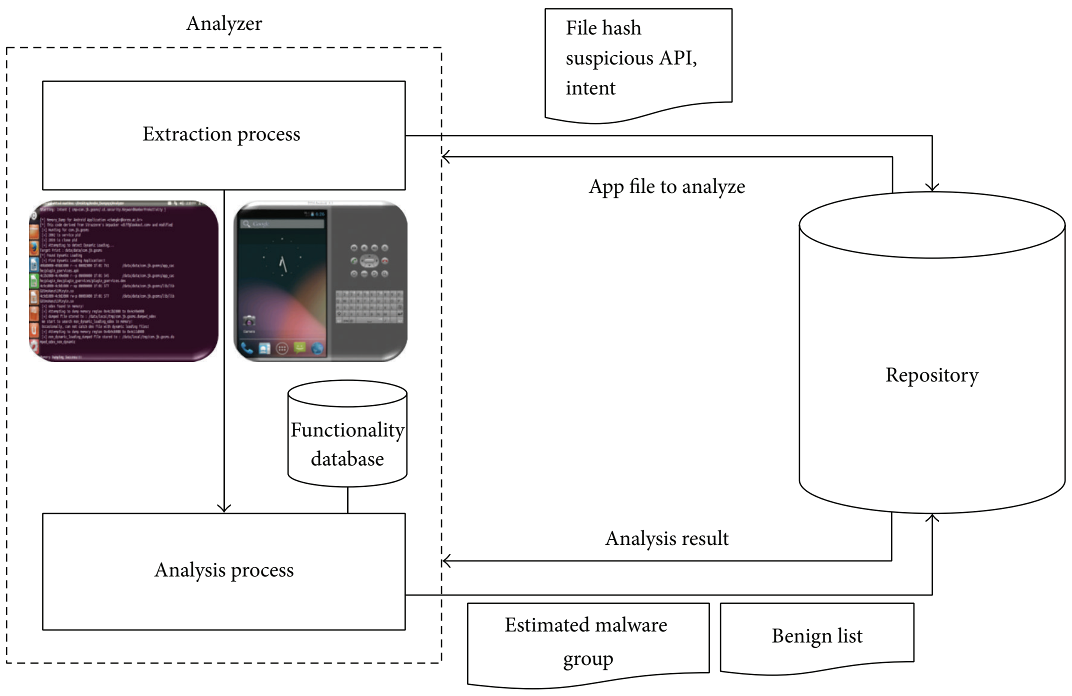


Figure 3: System architecture.

emulator that runs on Android 4.1.2 (level 16). The analyzer is composed of an extraction process for volatile memory acquisition and analysis process for malware detection. The analyzer restores the emulator to the initial state when a new application enters its queue.

4.2. Dataset Preparation. As shown in Table 1, we chose 906 malware samples composed of 13 malware families provided by the Korea Internet Security Agency (KISA) from March to December 2014. These malware samples are the latest smishing and spy applications reported by the Ministry of National Defense and mobile telecommunication companies in South Korea. To sanitize benign samples, we downloaded popular free applications with high rankings from Google Play in March 2015 and excluded the samples diagnosed by at least one AV vendor at VirusTotal; 1,776 benign samples were used for the experiments. Duplicate benign samples were excluded according to SHA256, and duplicate malware samples were excluded according to SHA256 and the package name. Our entire dataset is accessible at http://ocslab.hksecurity.net/sapimmds.

4.3. API Categorization. The Application Programming Interface (API) is a set of functions provided conveniently to control the main actions on a given platform, for example, the Android platform. For instance, we can easily implement the "sending an SMS message" functionality by calling some APIs, such as sendTextMessage() or sendMultimediaMessage(). Seo et al. [24] analyzed a variety of malware and benign samples and studied the distribution of APIs requested by each dataset, listing suspicious APIs.

Although various malicious behaviors seem to be similar to each other, the APIs found in malware vary based on the malware authors. To resolve this problem, we updated the suspicious API list by examining all the APIs that work similarly to those suspicious APIs listed by [24]. These APIs are involved in gathering private or system information, sending and deleting SMS messages, recording voice, and accessing

TABLE 2: Examples of malicious functionalities and their suspicious API call patterns.

Category	Suspicious API call pattern	Additional information
Hiding SMS notification	{getOriginatingAddress() $\vee$ getMessageBody() $\vee$ getDisplayMessageBody()} $\wedge^*$ abortBroadcast()	android.provider. Telephony.SMS RECEIVED with high priority
Hiding shortcut	setComponent EnabledSetting() $\wedge^*$ abortBroadcast()	
SMS message hijacking	{query() $\vee$ parse()} $\wedge^*$ getExternalStorageDirectory() $\wedge$ getExternalStorageState() $\wedge^*$ Transmission APIs[a]	content://sms $\wedge^*$ {Hiding SMS notification $\vee$ Hiding shortcut}
Contacts content hijacking	getContentResolver() $\wedge^*$ query() $\wedge^*$ getLine1Number() $\wedge^*$ Transmission APIs	{Phone.CONTENT_URI $\vee$ Contacts.CONTENT_URI} $\wedge^*$ {Hiding SMS notification $\vee$ Hiding shortcut}
Bookmark hijacking	getContentResolver() $\wedge^*$ Transmission APIs	BOOKMARKS_URI $\wedge^*$ {Hiding SMS notification $\vee$ Hiding shortcut}
Location information content hijacking	getLastKnownLocation() $\wedge^*$ Transmission APIs	{Hiding SMS notification $\vee$ Hiding shortcut}
Hijacking certificate for financial transaction	{getExternalStorageDirectory() $\vee$ getExternalStorageState()} $\wedge^*$ FileOutputStream $\wedge$ ZipOutputStream.close() $\wedge^*$ Transmission APIs	/npki $\wedge^*$ {Hiding SMS notification $\vee$ Hiding shortcut}

[a]The APIs of AQuery.ajax() or HttpClient() or DefaultHttpClient() or URLConnection() or HttpURLConnection() class.
A $\wedge$ B denotes that malware calls functions A and B successively (other API calls cannot be executed between A and B).
A $\wedge^*$ B denotes that malware calls functions A and B, but not necessarily successively (other API calls can be executed between A and B).
A $\vee$ B denotes that malware calls function A or B.

the content provider and web services. Details of the suspicious APIs that we defined are listed in Appendix.

4.4. Suspicious API Call Patterns of Known Malicious Behavior. In order to extract suspicious API call patterns, we reviewed previous our work [25]. As a result of this analysis, we found that most malware hijacks sensitive information without the victim's consent; occasionally, the malware is controlled by the adversary's C&C server via SMS messages. In addition, we found that the malware causes the victim to subscribe to premium services without his/her notice by removing the SMS confirmation message. To achieve its goal, malware requests the highest priority for received SMS messages and calls `abortBroadcast()` to hide a notification of SMS messages. Similar to premium-rate SMS, smishing applications receive SMS messages for C&C from the remote server and send hijacked sensitive information (e.g., contacts, SMS content, call logs, and certificate for financial transactions in South Korea). In this case, the smishing applications hide the received SMS message after the malicious behavior is completed. Thus, we have concluded that malware has distinct malicious behavior patterns and characteristics, and the malicious behavior is influenced by the invoked API call patterns. Using a feature database for those API call patterns, we can identify more efficiently whether the application is malicious. To this end, we analyze the relationship between malicious behavior and API call patterns. Table 2 lists the samples of suspicious API call patterns that we found, and the following descriptions explain how suspicious API call patterns cause malicious behavior.

(1) Hiding SMS notification: with high priority, malware successively calls APIs related to incoming SMS message handling, and then such malware calls `abortBroadcast()` to cancel the broadcasting of SMS-related actions. This malicious behavior pattern is indispensable for conducting additional malicious behavior without the victim's consent.

(2) Hiding shortcut icon: because malware hides its shortcut icon, the user cannot be aware of installing such malware. This malicious behavior pattern is indispensable for conducting additional malicious behavior without the victim's awareness.

(3) SMS message hijacking: when given the URI for managing SMS messages and APIs for querying such URI, malware can access the SMS content provider. Moreover, when given the APIs for accessing external storage and APIs for transmitting data, malware can access that storage in order to save the victim's SMS messages and transmit them to a remote server. This type of malicious behavior always requires some APIs to communicate with the remote server. The functions

for hiding SMS or shortcut icons are necessary for this malicious behavior.

(4) Contacts hijacking: when given the URI to manage contact contents and the APIs to query that URI, malware can access the contacts provider. Moreover, when given the APIs to transmit data, malware can transmit contact contents to a remote server along with the device's identifier information. This malicious behavior always requires some APIs to communicate with the remote server. The functions for hiding SMS or shortcut icons are necessary for this malicious behavior.

(5) Bookmark or location information hijacking and hijacking certificates for financial transactions: similar to SMS message hijacking, malware transmits this information to a remote server. In the case of hijacking certificates for financial transactions, given that the unique string "npki" represents the folder name of a given financial certificate in Korea, malware can compress that folder into a zip archive in the external storage and then transmit this information to the remote server. This malicious behavior always requires some APIs to communicate with the remote server. The functions for hiding SMS or shortcut icons are necessary this for malicious behavior.

4.5. API Call Pattern Matching. In order to compare the API call patterns of the application with the feature database for suspicious API call patterns, we adopted the longest common subsequence (LCS) algorithm [26] using dynamic programming. The LCS algorithm finds the longest common subsequence of two strings, where a subsequence is the sequence that maintains the same relative order, although not necessarily continuous. Since problem finding the LCS is one of the most famous problems in computer science and the basis of data comparison, the LCS is leveraged in malware analysis domain [15, 27]. For example, the sequences "abcd" and "abbdccccbd" have an LCS of "abcd." If the result of LCS is found in the feature database for suspicious API call patterns, we determine that the application conducts malicious behavior.

4.6. Overall Process. In the extraction process, our method dumps the meaningful volatile memory section where a given application is allocated on an Android emulator, as shown in Figure 3. In order to dump that memory section, we modified Strazzere's Native Development Kit code (https://github.com/strazzere/android-unpacker) that allows unpacking elaborate packing methods without depending on `gdb`. We enhanced this tool to dump the volatile memory section of the application that embeds the dynamic loading method with bytecode encryption. Memory acquisition then proceeds in the following order. First, our system retrieves a process ID (PID) of the application and a cloned process ID (CID), according to the package name. Next, our method retrieves the memory boundaries of the running CID using `/proc/CID/maps` by checking whether the packing and dynamic loading methods are adopted. If the application adopts packing and dynamic loading methods, the `odex` bytecode is allocated in the memory section after unpacking and dynamic loading are completed. Finally, our method attaches that process to `PTRACE` and copies the memory section whose signature is "`dey`" from `/proc/CID/mem`. In the extraction process, our method dumps the meaningful volatile memory section where a given application is allocated on an Android emulator, as shown in Figure 3.

In the analysis process, our method extracts the package name as an identifier, component name, and intent-specific information in `AndroidManifest.xml` from the `apk` file and disassembles the `odex` bytecode into `smali` code. After sorting the parsed components in ascending order, our method searches only `smali` files and the folder with the same component name. In the case of the application that embeds the dynamic loading method, our method retrieves all `smali` files because we believe that the dynamic loaded code can hide malicious codes. Dynamic loading method provides flexible memory allocation and extends the dynamic functionality during runtime. An application developer leverages dynamic loaded code for intellectual property protection. However, when utilized in malware, it is difficult for malware analyst to analyze malware embedding dynamic loaded codes; the number of malware families adopting antianalysis techniques has increased rapidly [4]. Furthermore, our method extracts suspicious APIs from `smali` files and creates a suspicious API sequence by comparing with the predefined suspicious API list. Our method determines whether the target application is malicious by checking predefined malicious functions or records of the functionality database in the suspicious API sequence of that application. After completing the analysis, the process stores the result in the repository.

4.7. Accuracy Test. We demonstrate that our method provides an effective metric for detecting packed malicious application and identifying malicious application as malware. We demonstrate that our proposed method provides an effective metric for distinguishing malware samples from those benign samples. Table 3 is a confusion matrix that shows the performance of the detection algorithm. As a result, eight benign samples were detected as malware (false positives), and 147

TABLE 3: Confusion matrix for malware detection.

Category	Actual class	
	Malware	Benign
Estimated class		
Malware	759	8
Benign	147	1,768

TABLE 4: Detection results for packed malware.

Packing method	Number of samples	Detected	Missed
APK Protect[a]	6	6	0
Bangcle	4	4	0
None	896	749	147
Total	906	759	147

[a]We use APK Protect professional version.

Table 5: Suspicious API list that we defined.

Category	Class	Method

Retrieving system information	TelephonyManager	getDeviceId()
		getLine1Number()
		getNetworkOperator()
		getSimOperatorName()
		getSimSerialNumber()
		getSubscriberId()
		getCallState()

	UUID	toString()
	WifiInfo	getMacAddress()

	WifiManager	getConnectionInfo()
		getWifiState()

Retrieving personal information	LocationManager	getLastKnownLocation()
		requestLocationUpdates()

	ContentResolver	query(), delete()
	Audio.Media	getContentUriForPath()
	Images.Media	getContentUri()
	Video.Media	getContentUri()
	—	getContentResolver()
	Uri	parse()

Sending or receiving SMS	SmsManager	getDefault()
		sendTextMessage()
		createFromPdu()
		getDisplayMessageBody()
		getMessageBody()
		getOriginatingAddress()
		getUserData()

	gsm.SmsManager	sendTextMessage()
		createFromPdu()
		getDisplayMessageBody()

Calling	telephony.ITelephony	endCall()
Recoding	AudioRecord	startRecording()
	MediaRecorder	start(), stop()
Data transmission	HttpURLConnection	getOutputStream()

	URLConnection	getInputStream()
		getOutputStream()

	ssl.HttpsURLConnection	getOutputStream()
	client.HttpClient	execute()
Data transmission	client.DefaultHttpClient	execute()
	JSONObject	put()
	AQuery	ajax()
Device policy management	DevicePolicyManager	isAdminActive(), lockNow()
	DeviceAdminReceiver	—
Dynamic loading	AssetManager	getAssets()
	DexClassLoader	loadClass()
	SecureClassLoader	—
	URLClassLoader	—
	Runtime	exec(), getRuntime()
	VMRuntime	getRuntime()
	System	load(), loadLibrary()

Table 5: Continued.

Category	Class	Method
Encryption	crypto.Cipher	doFinal() getInstance()
	crypto.KeyGenerator	generateKey()
	SecretKeySpec	—
Reflection	Class	getDeclaredMethod()
	reflect.AccessibleObject	setAccessible()
ETC	PendingIntent	getBroadcast()
	—	abortBroadcast
	FileOutputStream	—
	ZipOutputStream	close()
	PackageManager	setComponentEnabledSetting()
	Environment	getExternalStorageDirectory() getExternalStorageState()
	String	equalsIgnoreCase(), split()
	ActivityManager	restartPackage()
	AudioManager	setVibrateSetting() setRingerMode()
	Context	getSystemService()

malware samples that correspond to 16.23% of all malware samples were detected as benign (false negatives). To summarize, precision (precision = true positive/(true positive + false positive)), recall (recall = true positive/(true positive + false negative)), accuracy (accuracy = (true positive + true negative)/(true positive + false positive + false negative + true negative)), and the F-Score (F-Score = 2TP/(2TP + FP + FN) where TP is true positive, TN is true negative, FP is false positive, and FN is false negative) are 0.990, 0.838, 0.942, and 0.907, respectively. This implies that our proposed malware detection method is highly reliable. Moreover, the proposed method allows detecting all packed malware samples as malware, as shown in Table 4.

We demonstrate that our method provides an effective metric for identifying zero-day malware. We define a zero-day malware as an application with malicious behavior undetected by AV vendors. We leveraged ten of the malware samples offered by KISA and then checked the scanning results of various AV vendors, such as F-Secure, Avast, TrendMicro, Symantec, Kaspersky, McAfee, ClamAV, Sophos, and nProtect. The results of our experiment show that our methods can detect all zero-day malware samples.

Comparing Different Methods. To the best of our knowledge, the closest approaches in the literature to our approach are Wu et al. [18], Zheng et al. [19], Deshotels et al. [20], and Lee et al.'s work [21]. Wu et al.'s method [18] distinguished malware from benign applications with 97.87% accuracy. Zheng et al.'s method [19] detected 2,494 malware samples and 342 zero-day malware samples. Deshotels et al.'s method [20] detected malware with 94% accuracy and approximately 95% precision/recall. Lee et al.'s method [21] had accuracy and recall rates of more than 97%. For completeness of our approach, we need to compare ours with these approaches. However, they are not available for public use. Moreover, these approaches need more improvement for addressing such antimalware analysis techniques as packing, dynamic loading, and bytecode encryption. In contrast with these, our approach allows analyzing malware that embeds antimalware analysis techniques. Thus, we expect to detect more malware that embeds these techniques.

5. Limitation & Future Works

Our prototype has not yet implemented a module for malware classification. For such classification, a similarity comparison between suspicious API sequences enables calculating through sequence alignment algorithms, such as the Needleman-Wunsch and Smith-Waterman algorithms. In order to compare suspicious API sequences with each other, we transform the API method into ASCII code. The converted letters, for example, the ASCII code sequence, represent the behavior patterns of each application. Moreover, our proposed method extracts `odex` bytecode through the volatile memory acquisition process and employs static analysis to capture malicious behavior footprints. Similar to other dynamic analysis-based methods, ours has a limitation when the given malware is obfuscated by more sophisticated packing methods. In this case, our method fails to attach `PTRACE` in order to dump the meaningful bytecode in the memory section, thus yielding a less meaningful analysis. However, this drawback is common in dynamic analysis. To overcome such limitation, we will study the deobfuscation method to enhance our system in future work.

6. Conclusion

Most malware detection systems focus on analysis based on the signature or similarity matching of malware families, and they are not suitable for quick analysis intended to find the main purpose of such malware. For quick response against malware, we adopted the function-oriented approach based on suspicious API call patterns. Our proposed method dumps the meaningful volatile memory section where a target application is allocated and extracts suspicious APIs from `odex` bytecode by comparing with a self-defined suspicious API list. By matching API call patterns with our functionality database, our method can detect the functionalities implemented in a given mobile malware. Our experiments demonstrated that the proposed method performs well in detecting malware with high accuracy. We believe that our proposed method can be useful for responding to ever evolving malware.

Appendix

See Table 5.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was also supported by the ICT R&D Program of MSIP/IITP. [14-912-06-002 and the Development of Script-Based Cyber Attack Protection Technology]. In addition, this research is also supported by a Korea University Grant.

References

[1] McAfee, "McAfee Labs Threats Report," February 2015, http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q4-2014.pdf.

[2] Apkprotect, http://www.apkprotect.com/.

[3] Bangcle, http://www.bangcle.com/.

[4] R. Yu, "Android Packer Facing the Challenges, Building Solutions," https://www.virusbtn.com/pdf/conference_slides/2014/Yu-VB2014.pdf.

[5] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "DREBIN: effective and explainable detection of android malware in your pocket," in *Proceedings of the 21th Annual Network and Distributed System Security Symposium (NDSS '14)*, pp. 1–15, 2014.

[6] H. Peng, C. Gates, B. Sarma et al., "Using probabilistic generative models for ranking risks of Android apps," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS '12)*, pp. 241–252, Raleigh, NC, USA, October 2012.

[7] Y. Wang, J. Zheng, C. Sun, and S. Mukkamala, "Quantitative security risk assessment of android permissions and applications," in *Data and Applications Security and Privacy XXVII*, Lecture Notes in Computer Science, pp. 226–241, Springer, 2013.

[8] C. Yang, Z. Xu, G. Gu, V. Yegneswaran, and P. Porras, "DroidMiner: automated mining and characterization of fine-grained malicious behaviors in android applications," in *Computer Security—ESORICS 2014 :19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7–11, 2014. Proceedings, Part I*, vol. 8712 of *Lecture Notes in Computer Science*, pp. 163–182, Springer, Berlin, Germany, 2014.

[9] M. Zhang, Y. Duan, H. Yin, and Z. Zhao, "Semantics-aware Android malware classification using weighted contextual API dependency graphs," in *Proceedings of the 21st ACM Conference on Computer and Communications Security (CCS '14)*, pp. 1105–1116, ACM, Scottsdale, Ariz, USA, November 2014.

[10] W. Enck, M. Ongtang, and P. McDaniel, "On lightweight mobile phone application certification," in *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09)*, pp. 235–245, ACM, November 2009.

[11] J.-W. Jang, J. Yun, J. Woo, and H. K. Kim, "Andro-profiler: anti-malware system based on behavior profiling of mobile malware," in *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion (WWW Companion '14)*, pp. 737–738, 2014.

[12] P. Pearce, A. P. Felt, G. Nunez, and D. Wagner, "AdDroid: privilege separation for applications and advertisers in Android," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security (ASIACCS '12)*, pp. 71–72, Seoul, Republic of Korea, May 2012.

[13] Y. Zhang, M. Yang, B. Xu et al., "Vetting undesirable behaviors in Android apps with permission use analysis," in *Proceedings of the ACM SIGSAC Conference on Computer & Communications Security (CCS '13)*, pp. 611–622, ACM, Berlin, Germany, November 2013.

[14] K. Tam, S. J. Khan, A. Fattori, and L. Cavallaro, "CopperDroid: automatic reconstruction of Android malware behaviors," in *Proceedings of the 22nd Annual Network and Distributed System Security Symposium (NDSS '15)*, San Diego, Calif, USA, February 2015.

[15] Y. Ki, E. Kim, and H. K. Kim, "A novel approach to detect malware based on API call sequence analysis," *International Journal of Distributed Sensor Networks*, vol. 2015, Article ID 659101, 9 pages, 2015.

[16] D. Kim, J. Kwak, and J. Ryou, "DWroidDump: executable code extraction from android applications for malware analysis," *International Journal of Distributed Sensor Networks*, vol. 2015, Article ID 379682, 9 pages, 2015.

[17] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang, "Hey, you, get off of my market: detecting malicious apps in official and alternative android markets," in *Proceedings of the 19th Annual Network and Distributed System Security Symposium (NDSS '12)*, San Diego, Calif, USA, February 2012.

[18] D.-J. Wu, C.-H. Mao, T.-E. Wei, H.-M. Lee, and K.-P. Wu, "DroidMat: android malware detection through manifest and API calls tracing," in *Proceedings of the Seventh Asia Joint Conference on Information Security (Asia JCIS '12)*, pp. 62–69, Tokyo, Japan, August 2012.

[19] M. Zheng, M. Sun, and J. C. S. Lui, "Droid analytics: a signature based analytic system to collect, extract, analyze and associate android malware," in *Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom '13)*, pp. 163–171, IEEE, Melbourne, Australia, July 2013.

[20] L. Deshotels, V. Notani, and A. Lakhotia, "DroidLegacy: automated familial classification of Android malware," in *Proceedings of ACM SIGPLAN on Program Protection and Reverse Engineering Workshop (PPREW '14)*, ACM, January 2014.

[21] J. Lee, S. Lee, and H. Lee, "Screening smartphone applications using malware family signatures," *Computers & Security*, 2015.

[22] F-Secure, "Threat Report H1 2014," https://www.f-secure.com/documents/996508/1030743/Threat_Report_H1_2014.pdf.

[23] F-Secure, “F-Secure, 25 Years of the Best Protection in the World,” 2013, http://www.fsecure.com/en/web/labs_global/.

[24] S.-H. Seo, A. Gupta, A. M. Sallam, E. Bertino, and K. Yim, “Detecting mobile malware threats to homeland security through static analysis,” *Journal of Network and Computer Applications*, vol. 38, no. 1, pp. 43–53, 2014.

[25] J.-W. Jang, H. Kang, J. Woo, A. Mohaisen, and H. K. Kim, “Andro-AutoPsy: anti-malware system based on similarity matching of malware and malware creator-centric information,” *Digital Investigation*, vol. 14, pp. 17–35, 2015.

[26] L. Bergroth, H. Hakonen, and T. Raita, “A survey of longest common subsequence algorithms,” in *Proceedings of the 7th International Symposium on String Processing and Information Retrieval (SPIRE ’00)*, pp. 39–48, IEEE, A Coruña, Spain, 2000.

[27] Androguard, “Reverse Engineering, Malware and Goodware Analysis of Android Applications,” 2014, https://code.google.com/p/androguard/.

Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 3610157, 12 pages
http://dx.doi.org/10.1155/2016/3610157

Research Article

A Novel Iterative and Dynamic Trust Computing Model for Large Scaled P2P Networks

Zhenhua Tan,[1] Xingwei Wang,[1,2] and Xueyi Wang[1]

[1] *Software College, Northeastern University, Shenyang 110819, China*
[2] *College of Information Science and Engineering, Northeastern University, Shenyang 110819, China*

Correspondence should be addressed to Zhenhua Tan; tanzh@mail.neu.edu.cn

Received 30 October 2015; Accepted 11 January 2016

Academic Editor: Seung Yang

Trust management has been emerging as an essential complementary part to security mechanisms of P2P systems, and trustworthiness is one of the most important concepts driving decision making and establishing reliable relationships. Collusion attack is a main challenge to distributed P2P trust model. Large scaled P2P systems have typical features, such as large scaled data with rapid speed, and this paper presented an iterative and dynamic trust computation model named IDTrust (Iterative and Dynamic Trust model) according to these properties. First of all, a three-layered distributed trust communication architecture was presented in IDTrust so as to separate evidence collector and trust decision from P2P service. Then an iterative and dynamic trust computation method was presented to improve efficiency, where only latest evidences were enrolled during one iterative computation. On the basis of these, direct trust model, indirect trust model, and global trust model were presented with both explicit and implicit evidences. We consider multifactors in IDTrust model according to different malicious behaviors, such as similarity, successful transaction rate, and time decay factors. Simulations and analysis proved the rightness and efficiency of IDTrust against attacks with quick respond and sensitiveness during trust decision.

1. Introduction

Large scaled P2P network is developed rapidly in recent years because of its openness, anonymity, and being self-organized. P2P networking traffics always occupy Internet and P2P architecture is an important part of future Internet [1]. Various P2P applications, such as Mobile P2P, P2P lending (e.g., Zopa.com), and P2P e-commerce (e.g., eBay), have been welcomed by people and demonstrate high-capacity, variety, and rapid response [2–4].

Trust management has been emerging as an essential complementary part to security mechanisms of P2P systems, and trustworthiness is one of the most important concepts driving decision making and establishing reliable relationships. A well-defined trust model can provide meaningful decision support and help customers reduce possible risks during an Internet transaction. Like trust and reputation in social networks, trust evaluation in P2P systems is based on transaction histories.

Threat and attack always exist in P2P networks. Some common attacks against trust management remain in P2P network, such as Sybil Attack, Newcomer Attack, Betrayal Attack, Inconsistency Attack, Bad-Mouthing/Ballot Stuffing Attack, and Collusion Attack [5–8]. According to the behavioral characteristics of malicious nodes, they can be divided into individual and collusion malicious. These malicious behaviors destroy users' trust in P2P systems. Individual malicious nodes tend to act alone and make fault evaluation to any other nodes, while collusion malicious nodes act as a team causing more serious consequence.

In large scaled P2P networks, how to quickly calculate the mass trust evidences is a challenge to the efficiency of trust computing, and how to defend against collusion malicious behaviors is also a big challenge hindering the effectiveness of trust management. The main objective of this paper is to propose an iterative and dynamic trust computing model, named IDTrust for short, to effectively evaluate nodes trust degree against both individual and collusive malicious behaviors in large scaled P2P systems. First of all, explicit and implicit evidences are defined in IDTrust and then direct trust, indirect trust, global trust, and decision trust are modeled in mathematical expressions based on evidences. This paper has the following innovative features.

(1) How to obtain large number of trust evidences and make metric decision iteratively and dynamically is the key in trust computing within an environment of large scale P2P Internet. As a result, we propose a kind of computing architecture with hierarchical distribution trust, which layers these three parts independently: evidence collection, trust computing, and P2P communication, leading to a three-layered P2P topology and enhancing the distribution computing of P2P commercial communication.

(2) Traditionally, each trust computing involves all the trust evidence, which contradicts the characteristic of P2P rapid and large scaled data. Therefore, in this paper, we propose an iterative and dynamic trust computing model. Within this model, trust computing is iterative and only requires the newest trust evidence. The performance of trust computing is better.

(3) This paper models and measures the trust evidence from two aspects, explicit feedback and implicit feedback of customer, to ensure maximum accuracy and integrity. We design direct trust, indirect trust, global trust, and decision trust in IDTrust against individual malicious and collusion node behaviors, to make trust management more effective.

In the remainder of the paper, we introduce some related works in the next section. Section 3 describes the trust computation architecture (named IDTrust-Arch) which provides iterative and dynamic trust computation in P2P network for IDTrust. Trust evidence modeling in IDTrust is proposed in Section 4, and trust measurement of IDTrust is listed in Section 5. The simulations and analysis follow in Section 6, with conclusions afterwards in the last section.

2. Related Work

Researchers have done lots of work around trust computing in decades. Most of these achievements are about trust issues in P2P network, distributed ad hoc, sensor network, Internet, and so forth. Trust computing includes three modules, namely, trust evidence acquisition, trust evaluation, and trust inference. We conclude current related work into four kinds in this section.

(1) Trust Computing Based on Local Transaction Evidences. Direct trust or local trust expresses to what extent a trustor believes a trustee based on the trustor's own local transaction history with the trustee. Buchegger and Le Boudec proposed CONFIDANT protocol based on local information [9], and the model processes secondhand information directly through friends in P2P network or ad hoc network. This method could defend against single malicious behaviors but could not avoid betrayal and collusion. Damiani et al. present local trust model named XRep [10]; they believe that nodes from same IP cluster are collusion group in P2P network. XRep has good convergence to recognize collusion behaviors, but it has limitation and some honest nodes may also be included in the collusion IP cluster. Jia et al. [11] optimize direct trust by power law and design forgetting factor based on time. Their model makes trust decision through a process of asking the neighbors' feedback of the target node. This model is effective but not versatile because it assumes that the nodes must have trusted neighbors. Direct trust has certain cognitive abilities on single malicious node, but it has limitations on node collusion and malicious tampering evidence.

(2) Trust Computing Based on Global Historical Information. Global trust is computed from other nodes recommendations to measure how trustworthy the target node is. To compute global trust, a trustor needs to aggregate all the trustee's trust evidences.

There are two major ways to calculate global trust. The first way is based on central server calculation, such as eBay which requires the peer nodes to deal with each other on the central platform. The other way is fully distributed, such as EigenTrust [12] which is proposed by Kamvar et al. EigenTrust reputation system can infer a unique global trust in a very distributed way by history. Such a global model does not need an administration center and difficultly guarantees a fast and secure convergence when computing the global trust. Nevertheless, it inspires our works. Dou et al. [13] improved the EigenTrust system in computing convergence and model security. However, there remains an efficiency problem and its security mechanism is only from punishment and certification. Xiong and Liu [14] proposed a PeerTrust model with three basic trust parameters and two adaptive factors and then defined a general trust metric to combine them efficiently. Jøsang et al. [15] proposed a trust inference method for simplifying a complex network to express it in a series of parallel networks. This solution may lead to the loss of trust information. They proposed an edge splitting method in the further works [16] to address this problem. Nevertheless, this method is valid only on a simple trust network and invalid on complex trust networks. Wang and Nakao propose Poisoned Water [17] and Shaikh et al. propose GTMS [18]; they measure trust based on global group to improve the convergence rate of the trust calculation and global trust accuracy. Song et al. propose global trust based on fuzzy logic inference rules [19]. This method has higher detection rate of malicious nodes, but it can only avoid simple malicious behaviors and cannot defend against various attacks on trust mechanism. Gan et al. [20] present a new method for trust recommendation. This method uses confidence factor to comprehend direct and recommendation trust and establishes reward and punishment model to encourage fair global recommendation.

(3) Trust Computing Based on Global and Historical Information Correlation. Correlation trust is based on global trust with similarity or correlation factors. Li et al. [21] propose a global trust SWRTrust with cosine similarity, which can identify collaborative cheating malicious behaviors. Das and Islam [22] propose an excellent dynamic trust computation

model SecuredTrust to cope with the strategically altering behavior of malicious agents and to distribute workload as evenly as possible among service providers. Qiao et al. [23] propose a novel network group behavioral model based on trust by exploring behavior similarity, aiming at perception and response of malicious network incidents. The proposed model establishes trust relationship between nodes using large scaled network topology and uses relevant trust concept to increase trust value between weak correlations. Other models like [24, 25] also propose global trust with correlation computation to improve trust decision performance based on large scaled transaction histories.

(4) Trust Computing Based on Inference Network with Multifactors. Another kind of trust computing model is based on inference network with probability or multidimensional factors. Kuter and Golbeck [26] propose a trust inference model SUNNY based on trust network by evaluating trust and confidence with inference probability. It has good effectiveness to discover trust recommender paths. Vu and Aberer [27] use trust model to monitor dishonest behaviors via filtering unfair ratings in trust network. Wang and Singh [28] propose trust model based on evidences conflict probability. And Can and Bhargava [29] propose a distributed trust computing algorithm SORT, where nodes create their own trust network according to local trust information and infer the trustworthiness of target nodes according to history interaction and recommendation information. Liu et al. [30] infer trust relationships by small world theory and apply the proposed model in service network. Tan et al. [31] propose a novel trust inference model based on probabilistic model and balance theory for P2P network, and the proposed algorithm could discover more valuable trust evidences paths for inferring the target's trust. They applied the proposed model in [32] and get a relatively effective inference performance. Gradually, researchers begin to infer trust degree with multidimensional evidence factors. Wang and Wu [33] proposed a multidimensional evidence-based trust management system with multitrusted paths (MeTrust for short) to conduct trust computation on any arbitrarily complex trusted graph. The trust computation in MeTrust has three tiers, namely, the node tier, the path tier, and the graph tier. It is an excellent trust model. However, it does not provide distributed storage structure for P2P system. Jiang and Li presented a novel reputation-based trust mechanism for P2P e-commerce systems [34]. In this mechanism, one peer has two kinds of reputations, local reputations and global reputations. To compute the local and global reputations precisely and to obtain stronger resistibility to attacks as well, they use many comprehensive factors in computing trust value in the mechanism. Basically, this model is a comprehensive mechanism. However, its time factor is only linear and there is no clear method to resist team malicious behaviors. Tan et al. [35] presented a global trust model with correlation factor based on communication history and improved the time factor with exponential equation. It shows a rational history vector and presents three trust models with multidimensional trust factors.

Inspired by the above four kinds of trust computing model, in this paper, the proposed IDTrust has direct trust, indirect trust, global trust, and decision trust. Meanwhile, the IDTrust aims to solve trust issues in large scaled P2P which have dynamic and rapid evidence and to improve trust computing efficiency in iterative and dynamic computation based on evidences with multifactors. Some definitions and parameters have been discussed in our former work [24, 31, 32, 35], and IDTrust integrates methods and improves computation architecture.

3. IDTrust-Arch

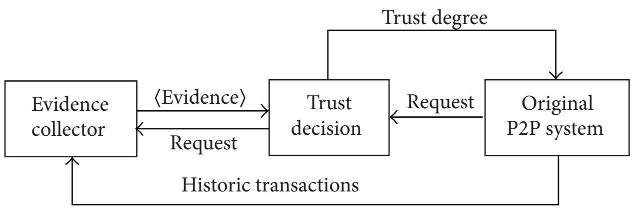
Trust measurement must be based on transaction or communication histories. Large scaled P2P systems have high frequency in generating data, strong dynamics in transactions and large scaled in transaction histories. As a result, trust evidences also have such same characteristics as high generating speed and large scaled histories. How to compute these dynamic and large scaled trust evidences efficiently becomes a challenge to trust model.

In IDTrust, we firstly design a distributed iterative computing architecture named IDTrust-Arch as shown in Figure 1(a), including *evidence collector* (EC) and *trust decision* (TD) which are designed extendedly based on P2P system architecture. In IDTrust-Arch, EC collects historic transactions from P2P system and feeds back evidence vector to TD, while TD computes the trustworthiness of objective nodes according to evidence vector and feeds back trust degree to P2P system according to request instructions. Relations of EC, TD, and original *P2P service* in IDTrust-Arch have the following features.

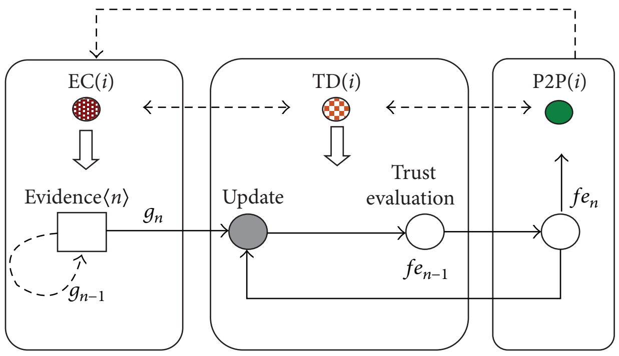
(1) Independent and Extended Functions. Within large scaled P2P system, distributed trust computing may cause huge communication traffic likely influencing the original P2P service quality. We mentioned this situation and it is under consideration in IDTrust-Arch. Different from traditional P2P trust model which treats trust computation as an inner-binding service, the proposed *IDTrust-Arch* is independent from P2P service. Firstly, EC, TD, and *P2P service* have separate communication ports, and EC and TD do not occupy P2P information channel. For any peer node, EC and TD are external extended functions but not internal binding service. Secondly, at any time, EC, TD, and *P2P service* can be deployed in different physical machines or different processes separately.

(2) Distributed but Topology Consistency. As a result of the independent functional design, each node i in P2P system can be extended into three nodes as node(i), EC(i), and TD(i). EC (and also TD) in all peer nodes can constitute a P2P network independently, using same look-up routing algorithm as the original P2P system. Figure 1(d) shows the topology relations between EC, TD, and the original P2P system. As the figure shows, the three will have the same topologies and communicate with each other.

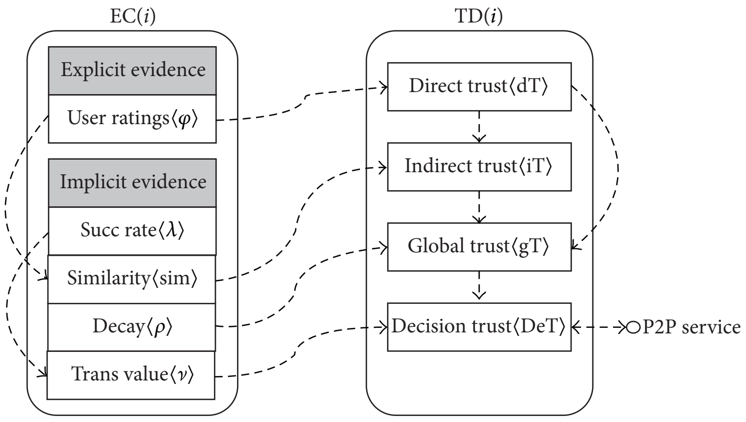
(3) Iterative Computing. IDTrust-Arch designs an iterative computing model for large scaled evidences in TD and EC, which only computes latest evidence vector each time based



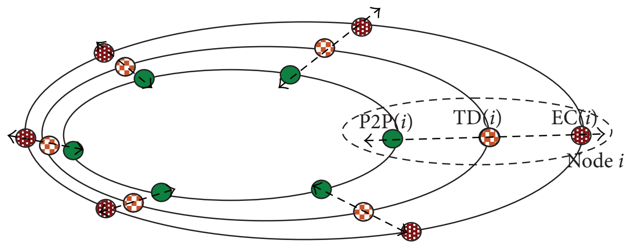
(a) Relations between EC, TD, and original P2P topology in IDTrust-Arch



(b) Iterative computing model in IDTrust-Arch



(c) Relations between evidences and trust model in IDTrust



(d) Topology

FIGURE 1: IDTrust-Arch.

on former result. Figure 1(b) shows the iterative trust computing processes. We can see that each computation is using the result of last computing. This kind of new method can greatly improve the trust computing efficiency and contains all the information of evidence by calculation of iterative accumulation, comparing it to the traditional one, which involves all the evidence every time. When large numbers of nodes in P2P network are online dynamically, for example, sometimes online, sometimes offline, traditional computing model results in difference of trust evaluation due to the inconsistence of set of historic evidences (e.g., there are 100 evidences of activity last time but may be only 30 next time). The iterative model in IDTrust-Arch can make use of evidences more efficiently by keeping evaluation consistency all the time. This

Table 1: Example: cumulative evaluation value of φ_n (1, 2).

Iteration number	$\varphi^{t_{\text{now}}}(1,2)$	Time stamp	α	$\varphi_n(1,2)$
Initial	Ratings from N1 to N2	$T1 = 10$	1	0
1	0.95	$T2 = 20$	1	0.95
2	0.9	$T3 = 30$	0.75	0.9125
3	0.7	$T4 = 80$	0.918367347	0.717346939
4	0.8	$T5 = 100$	0.395061728	0.75

kind of trust computation of IDTrust-Arch can adapt to high speed and large scaled P2P evidences flexibly. The following equation is the formalized formula of updating function:

$$fe_n(p,q) = \text{update}(fe_{n-1}(p,q), g_n(p,q)),$$
$$g_n(p,q) = \text{update}(g_{n-1}(p,q), \text{evidence}(p,q)). \quad (1)$$

Here, we assume that node p (called trustor) needs to calculate the trust degree of node q (trustee). And $fe_n(p,q)$ denotes the nth trust decision computing in node p while $g_n(p,q)$ denotes the nth evidence modeling. Appropriate initialized data will be set for the very first iterative computing. The following IDTrust modeling is based on IDTrust-Arch and (1). It has been noted that (1) is only a formalized iterative computation method to guide modeling in IDTrust, and actual model symbols and parameters depend on model's requisitions.

4. Trust Evidence Modeling for IDTrust

Trust evidences in IDTrust come from EC, including explicit feedback evidence and implicit feedback evidence. The explicit feedback evidence is from customer nodes active evaluation, while implicit feedback evidence should be excavated from transactions. We formalize trust evidence vector in IDTrust as Evidence(p,q) to represent evidences that node p has upon node q. That is modeled by

$$\text{Evidence}(p,q) = \langle\langle\varphi\rangle, \langle\lambda, \nu, \rho, \text{sim}\rangle\rangle. \quad (2)$$

Explicit evidence vector in IDTrust is mainly consisting of user ratings φ, and of course it could be further expanded to multidimensional evidence vector in specific P2P systems. Implicit evidence vector in IDTrust mainly consists of transaction success rate λ, transaction value ν (based on transaction price), trust decay factor ρ, and nodes similarity sim. The following will model both explicit evidence vector $\langle\varphi\rangle$ and implicit evidence vector $\langle\lambda, \nu, \rho, \text{sim}\rangle$ in iterative style according to (1).

4.1. Explicit Evidence.

4.1. Explicit Evidence. Usually explicit evidence is fed back by user ratings or satisfactions. In practice, the quantitative or qualitative feedback can be by scores (5 points or 100 points) or by satisfaction (satisfied, mostly satisfied, not satisfied, etc.). In this paper, we use *user evaluation* by range $[0,1]$ to unify users' ratings or satisfactions.

Definition 1. User evaluation is denoted by $\varphi^t(p,q) \in [0,1]$ and represents that user p evaluated user q with a $[0,1]$ rating after transaction with q at the time t. $\varphi^t(p,q) = 0$ represents that node p is fully unsatisfied with node q's service, while $\varphi^t(p,q) = 1$ represents being completely satisfied. Let $\varphi^{t_{\text{now}}}(p,q)$ mean the latest user evaluation at the time t_{now}, while $t = t_1$ means the very beginning evaluation time.

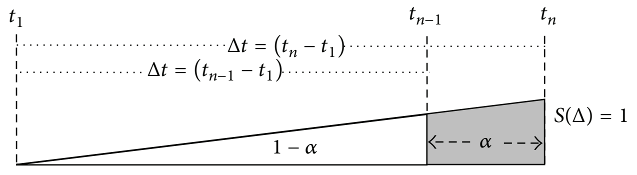


Figure 2: Dynamic factor of current evaluation.

Definition 2. Using $\varphi_n(p,q)$ to denote the current *cumulative evaluation value* by node p upon node q, based on the evaluation of last time according to iterative model, assume $\varphi_0(p,q) = 0$; that is,

$$\varphi_n(p,q) = \alpha \cdot \varphi^{t_{\text{now}}}(p,q) + (1-\alpha) \cdot \varphi_{n-1}(p,q), \quad (3)$$

where the weight α is the dynamic factor of current evaluation $\varphi^{t_{\text{now}}}(p,q)$. In IDTrust, α will be dynamically changed by the distance between t_{n-1} and t_n, and larger distance will bring bigger α, as shown in Figure 2. In other words, the current evaluation $\varphi^{t_{\text{now}}}(p,q)$ will be more important if user p did not transact with q after a longer time.

In Figure 2, we calculate α by the method of right triangle where area is 1 (unit area) and the base is global time difference (from t_1 to t_{now}) of interaction between p and q. Therefore, assume $\alpha = 1$ at the very beginning ($t = t_1$); the weight α of current evaluation is the area of the triangle from time t_{n-1} to t_n. That is,

$$\alpha = \begin{cases} 1 - \left(\dfrac{t_{n-1} - t_1}{t_n - t_1}\right)^2, & \text{if } t_n > t_1 \\ 1, & \text{if } t_n = t_1. \end{cases} \quad (4)$$

For example, assume there are four transactions that happened between node $N1$ and node $N2$, just as shown in Table 1. Then the cumulative evaluation value would be computed according to (3) based on evaluations from $N1$ to $N2$, while α changed based on transaction time stamps. After four iterations, the value of $\varphi_n(1,2)$ is 0.75. This value is less than the average of transaction rations from $N1$ to $N2$ because of the dynamic changes of α.

4.2. Implicit Evidences

4.2.1. Successful Rate. Successful transaction is a positive stimulus for nodes, while failure of transaction (or other situations such as complaint) would result in a negative growth to nodes trust.

Definition 3. $\lambda_n(p,q)$ denotes the percentage of *cumulative successful transaction* of node p to node q. Assume $\lambda_0(p,q) = 0$. That is,

$$\lambda_n(p,q) = \frac{(n-1)\lambda_{n-1}(p,q) + \lambda^{t_{\text{now}}}(p,q)}{n}, \tag{5}$$

where $\lambda^{t_{\text{now}}}(p,q) = \{0, \text{if successful}; 1, \text{if failed}\}$ represents whether the current nth transaction is successful or not.

4.2.2. Transaction Value. Transaction amount (in real or virtual currency) is typical implicit trust evidence. People usually think that the service provider with larger transaction amount is more valuable to trust in intuition. However, people are also worried about the malicious or dishonest block trade which may be only baits for honest users. Thus, we model the transaction amount as *transaction value* which is based on both transaction amount and successful rate.

Definition 4. Use $\nu^{t_{\text{now}}}(p,q) = \omega^{t_{\text{now}}}(p,q) \cdot \lambda_n(p,q)$ to denote the *current transaction value* of the current user transaction amount $\omega^{t_{\text{now}}}(p,q)$.

For instance, if the current transaction amount $\omega^{t_{\text{now}}}(p,q)$ is 100 but the current successful rate $\lambda_n(p,q)$ is only 0.25, then we take the current transaction value to be only $100 * 0.25 = 25$. This engagement can stimulate honest users to improve their transaction successful rate λ and also suppress the malicious behaviors which strategically raise the trustworthiness by transaction amounts meanwhile.

Definition 5. Using $\nu_n(p,q) = \nu^{t_{\text{now}}}(p,q) + \nu_{n-1}(p,q)$, to denote *cumulative transaction value* of the transaction amount, assume $\nu_0(p,q) = 0$.

4.2.3. Time Decay Factor for Trustworthiness. In the process of trust evaluation, the more recent the trust evidence is, the more important it is. If a user node takes a long time without transacting with other nodes, its trust will be degraded gradually. We apply exponential function to design the decay factor of direct trustworthiness of nodes, which will be used in direct trust in IDTrust. That is,

$$\rho = e^{-\kappa\Delta t}. \tag{6}$$

Here, k ($k > 0$) is the decay speed, and $\Delta t = t_n - t_{n-1}$ denotes the time difference for a certain user node. Figure 3 is the illustration of decay curves of four different decay speeds. Apparently, the lager k is, the faster the decay would be.

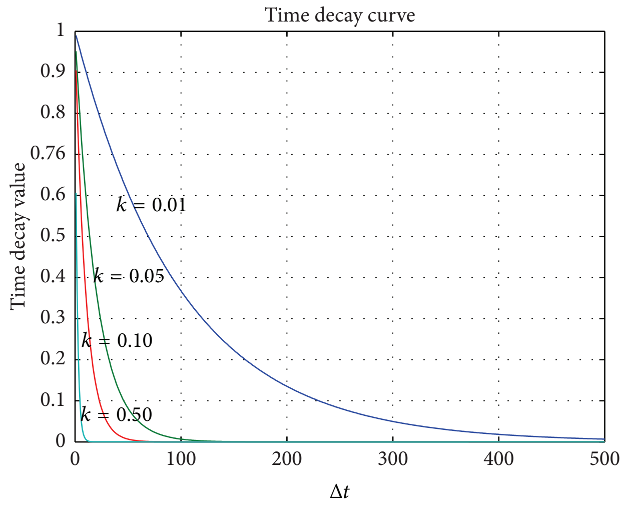


FIGURE 3: Time decay curves.

4.2.4. Similarity of User Evaluations. Similarity is one kind of important implicit evidence which indicates the extent to which two nodes are alike, especially when there is no direct transaction between two strange nodes. In order to measure the similarity between user nodes p and q, we choose the third set of nodes that have interaction with both of them to be a computing object. For example, node A did not know B, but both share common friends $\{C, D\}$, and then A can judge similar correlation with B via its friends $\{C, D\}$. If the communication history among $A \sim \{C, D\}$ is similar to the history among $B \sim \{C, D\}$, then we think A and B have very similar correlation.

Let I_x denote the set of nodes which node x has ever interacted with. Let $I_{pq} = I_p \cap I_q$ denote the common third-part set which interacts with both users p and q. The similarity computation is based on *user-user* matrix $R(n \times n)$ whose element r_{ij} is

$$r_{ij} = \begin{cases} \varphi_n(i,j), & \text{if } i \neq j \\ 1, & \text{if } i = j. \end{cases} \tag{7}$$

If no user interaction happened between user nodes i and j, we assume $r_{ij} = 0$. Then,

$$R = (r_{ij})_{n\times n} = \begin{bmatrix} 1 & r_{12} & \cdots & r_{1n} \\ r_{21} & 1 & \cdots & r_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ r_{n1} & r_{n2} & \cdots & 1 \end{bmatrix}. \tag{8}$$

Each row i in R denotes evaluation to all other user nodes from node i. There are many methods to calculate the similarity between two items, such as cosine similarity, correlation similarity, and adjusted cosine similarity. Cosine similarity is a general method to compute the similarity between two user nodes. However, pure cosine formula cannot distinguish the subjective difference of evaluation criteria that different users have for the same object. For instance, for a certain user x, user p evaluated x with value 0.9 after a transaction, and user q evaluated it with value 0.6 in another transaction. The evaluated value is quite different; however, it is probable that

0.9 and 0.6 both stand for "*satisfied*" separately in perspective of p and q, and the difference may only be because user q has more strict evaluation criteria. In this situation, ordinary cosine formula is not appropriate. Thus, we apply the adjusted cosine formula that is subtracted by average value to compute the similarity based on matrix R. That is,

$$c\,\mathrm{sim}_n(p,q) = \frac{\sum_{c\in I_{pq}}\left(\varphi_n(p,c)-\overline{\varphi_n(c)}\right)\cdot\left(\varphi_n(q,c)-\overline{\varphi_n(c)}\right)}{\sqrt{\sum_{c\in I_{pq}}\left(\varphi_n(p,c)-\overline{\varphi_n(c)}\right)^2}\cdot\sqrt{\sum_{c\in I_{pq}}\left(\varphi_n(q,c)-\overline{\varphi_n(c)}\right)^2}}, \quad (9)$$

where $\overline{\varphi_n(c)}$ denotes the average evaluation value of node $c \in I_{pq}$.

In the above similarity equation, three questions need to be further considered.

(1) Sparse Common Nodes. It is a challenge when $|I_{pq}|$ is zero or very small. This phenomenon may even result in similarity beyond computing. Thus, in IDTrust, we assume that similarity should be computed by cosine only when $|I_{pq}|$ reaches threshold τ (natural number that is larger than 1).

(2) Result Adjusting. Although the range of user evaluation φ is within $[0,1]$ in IDTrust, due to inner product of deviation, the return value of adjusted cosine similarity by (9) is within $[-1,1]$, where range $[-1,0]$ denotes the evaluation vectors angle of p and q is between degrees $[90,180]$ which means the two vectors are completely irrelevant (orthogonal) or even opposite. Thus, we assume it is completely dissimilarity in this case, and the similarity is set to zero when $c\,\mathrm{sim}_n(p,q)$ return $[-1,0]$ in IDTrust. As a result, the range of similarity will be adjusted to $[0,1]$, which is easier to control.

(3) Default Similarity. At the very beginning (or when similarity is beyond computing), we need to initialize a default similarity. In IDTrust, the default similarity is 0.5.

Hence, (9) is modified to be

$$\mathrm{sim}_n(p,q) = \begin{cases} 0, & \text{if } \left(\left|I_{pq}\right| > \tau\right) \wedge \left(c\,\mathrm{sim}_n(p,q) \in [-1,0)\right) \\ c\,\mathrm{sim}_n(p,q), & \text{elseif } \left(\left|I_{pq}\right| > \tau\right) \wedge \left(c\,\mathrm{sim}_n(p,q) \in [0,1]\right) \\ 0.5, & \text{otherwise.} \end{cases} \quad (10)$$

Also, we assume the similarity of $\mathrm{sim}_n(x,x) = 1$, which means the similarity among nodes is reflexive. Note that similarity among nodes is symmetrical; that is, $\mathrm{sim}_n(p,q) = \mathrm{sim}_n(q,p)$. Therefore, similarity among nodes denotes the compatibility relationship. It indicates that we can get a similar set in the whole set by compatibility relationship, which may be a new way to get complete coverage for similar nodes in the whole set in our further research. But in IDTrust now we have not applied this feature.

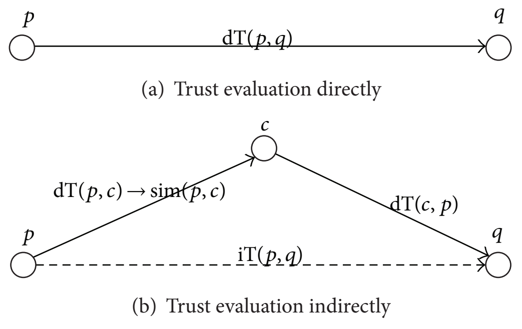

p
q
dT(p, q)
(a) Trust evaluation directly
c
p
q
dT(p, c) → sim(p, c)
dT(c, p)
iT(p, q)
(b) Trust evaluation indirectly

Figure 4: Direct/indirect trust computing.

5. Iterative Trust Measurement Models in IDTrust

Based on the above iterative computing architecture and evidence modeling, we design direct trust model, indirect trust model, global trust model, and decision trust model for IDTrust, against malicious behaviors especially collusion behaviors. Figure 1(c) shows relations between trust evidences and trust measurement models.

5.1. Direct Trust. Direct trust (also called local trust) is based on the direct transaction experiences between the trustor and trustee, as Figure 4(a) shows. We define the direct trustworthiness of node p to node q from user evaluation which is from the direct transaction between the two nodes. Let $\mathrm{dT}_n(p,q) \in [0,1]$ denote the direct trust of node p to node q. That is,

$$\mathrm{dT}_n(p,q) = \varphi_n(p,q). \quad (11)$$

As you can see, the better the service that node q provided, the higher the direct trust degree that node q obtains from node p.

5.2. Indirect Trustworthiness. To compute indirect trust by traditional trust models, the trustor needs to aggregate all of the trustee's transaction evaluations from recommenders and then compute or infer indirect trust degree of target node by those recommenders' evidences. Figure 4(b) illustrates that node p calculates indirect trust degree of node q through recommender node c.

As we know, we can obtain the direct trust $\mathrm{dT}(c,q)$ from recommender node c and direct trust $\mathrm{dT}(p,c)$ from p. Some related indirect trust models use these two kinds of direct trust to calculate indirect trust of the target node, such as $\mathrm{dT}(p,c) * \mathrm{dT}(c,q)$ or other similar forms. However, since the direct trust is usually based on user evaluation and the user evaluation is subjective in a certain extent, different users have different evaluation criteria, just as we discussed in Section 4.2.4 about similarity of user evaluations above. Thus, we consider user similarity between nodes p and c instead of direct trust of p and c, for the similarity computation in IDTrust is also based on user evaluation (which is also equivalent to direct trust). Using similarity can also avoid collusion

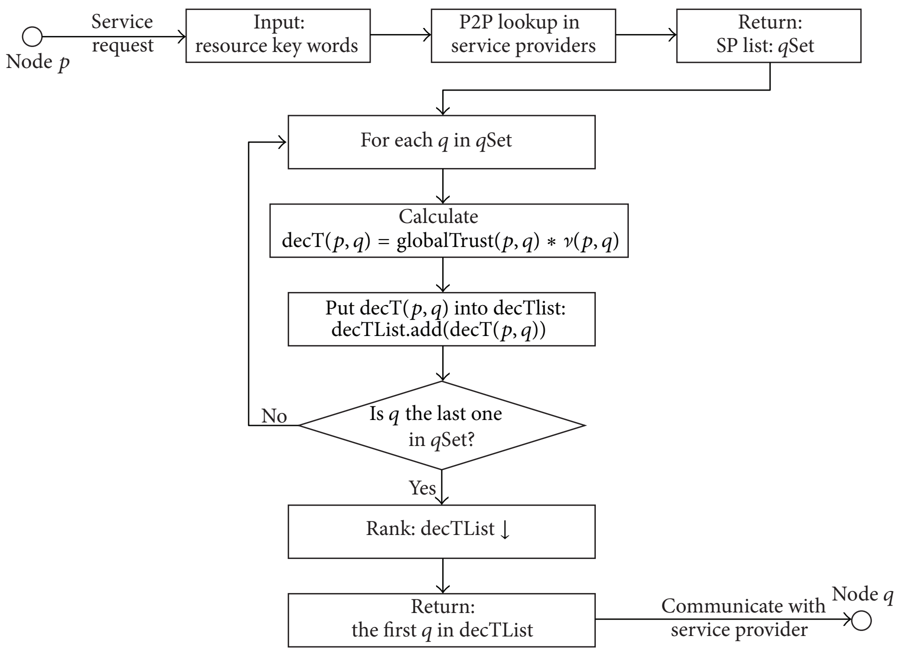

Node p
Service request
Input: resource key words
P2P lookup in service providers
Return: SP list: qSet
For each q in qSet
Calculate decT(p, q) = globalTrust(p, q) ∗ ν(p, q)
Put decT(p, q) into decTlist: decTList.add(decT(p, q))
Is q the last one in qSet?
No
Yes
Rank: decTList ↓
Return: the first q in decTList
Communicate with service provider
Node q

Figure 5: Trust decision process.

malicious recommenders, since malicious evaluation would quite differ from the honest evaluation.

Let $\mathrm{iT}_n(p,q) \in [0,1]$ denote indirect trust of node p to node q, which is calculated by direct trust of recommender set cSet to node q, and use the similarity between node p and cSet as another indirect parameter. That is,

$$\mathrm{iT}_n(p,q) = \begin{cases} \sqrt{\dfrac{\sum_{c\in c\mathrm{Set}} \mathrm{sim}_n(p,c)\cdot \mathrm{dT}_n(c,q)}{|c\mathrm{Set}|}}, & \text{if } |c\mathrm{Set}|>0 \\ 0, & \text{otherwise,} \end{cases} \tag{12}$$

where if recommender nodes set cSet is empty, the indirect trust return is zero. In order to read the result clearly, the above formula amplifies the value with square root, since both similarity and direct trust ranged within $[0,1]$.

5.3. Global Trust. Now, we define the global trust as the weighted sum of direct trust and indirect trust, where the general weight is the decay factor ρ:

$$\mathrm{gT}_n(p,q) = \rho\cdot(\zeta\cdot \mathrm{dT}_n(p,q) + (1-\zeta)\cdot \mathrm{iT}_n(p,q)) + (1-\rho)\cdot \mathrm{gT}_{n-1}(p,q). \tag{13}$$

Here, $\zeta \in [0,1]$ is the weight of the direct trust with default value 0.5 which could be adjusted by users according to the requirement.

5.4. Decision Trust. Decision trust is an integrated value built from iterative global trust and implicit evidence of transaction value, to represent a comprehensive expression for user nodes' trust.

Let $\mathrm{decT}_n(p,q) = \mathrm{gT}_n(p,q)\cdot\nu_n(p,q)$ denote decision trust which is based on global trust and implicit evidence transaction value. For example, if $\mathrm{gT}_{1010}(1,3) = 0.65$ and transaction value $\nu_{1010}(1,3) = 100$, we can get $\mathrm{decT}_{1010}(1,3) = 65$ indicating the integrated trust of node 3 is 65 from the perspective of node 1. As we can see, the $\mathrm{gT}_n(p,q)$ is a global factor while $\nu_n(p,q)$ is a relatively local factor; we apply these two factors to help in decision making.

Hereby, we also design a trust decision algorithm as shown in Figure 5. In the algorithm process, as a trustor, user node p initiates the trust decision in order to find the most trustworthy nodes. At the beginning, node p inputs resource keywords to request P2P services, and then P2P network returns a service provider list (qSet) specifying who can provide related resource services. Each decision trust (decT) will be calculated through EC and TD, and all the decision trust will be ranked in descending order. At last, the node q that has the highest decision trust value will be recommended to node p.

6. Simulations and Analysis

In order to prove the rightness and efficiency of IDTrust, we design a simulation system based on IDTrust-Arch and our former work. We will separately verify IDTrust performance against dynamic individual malicious and collusion malicious environment and verify the iterative computation performance comparing with traditional model.

6.1. Simulation Environment. Firstly, we design a P2P communication architecture as in Figure 6 based on IDTrust-Arch introduced above. In the system, service provider (SP)

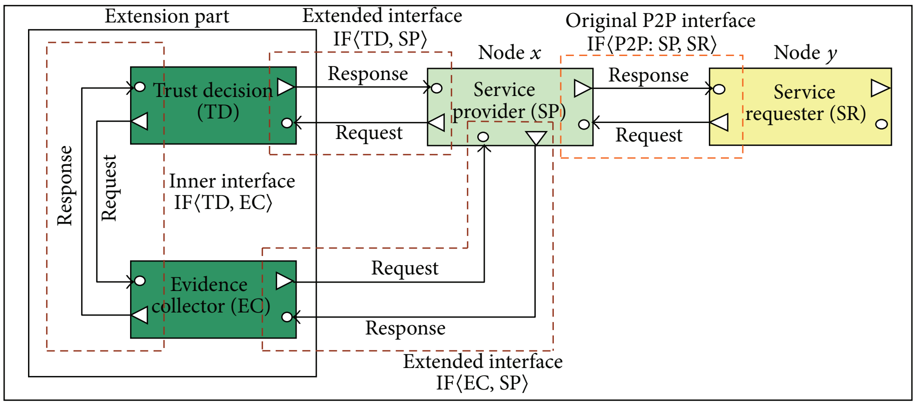


FIGURE 6: Communication architecture of simulation system.

is the node that provides services while service requester (SR) needs the resource services. EC is to collect and integrate the historic data and obtain evidence vector from P2P system. TD is to compute the trustworthiness of objective nodes according to evidence vector provided by EC. The original P2P interface IF⟨P2P: SP, SR⟩ manages the communication between node x and node y. The extended interface IF⟨TD, SP⟩ manages the communication between SP and TD. And interface IF⟨EC, SP⟩ manages the communication between SP and EC. The inner interface IF⟨TD, EC⟩ manages the communication between TD and EC.

We implement the above simulation system based on MS.NET Framework 4.5 and MS P2P Networking platform with *c#* programming language, thread pool technique, and distributed communication. In topology, we use simplified *Chord* to be the P2P routing algorithm.

We simulate three kinds of P2P nodes similar to [24, 35].

(1) Node of Class A. It is the normal node in P2P system, which provides normal service and evaluates service almost equal to simulated resource value.

(2) Node of Class B. It is an individual malicious node that provides false service and makes fault evaluation for any other nodes at random independently.

(3) Node of Class C. It belongs to some malicious team nodes, providing false service, and evaluates normal nodes with fault feedback. It should be noted that it overstates team members' services with high score to cheat other nodes out of their team.

We simulated 1000 nodes and each node has 50~100 resources abstract and designed 4 kinds of experiment and almost 1,000,000 simulated transactions happened between nodes. Table 2 shows the nodes proportion allocation for different experiments.

Table 3 initializes different parameters' value. Most of these parameters will be dynamically changed with the iterative computation going on.

TABLE 2: Nodes proportion setup in different experiments.

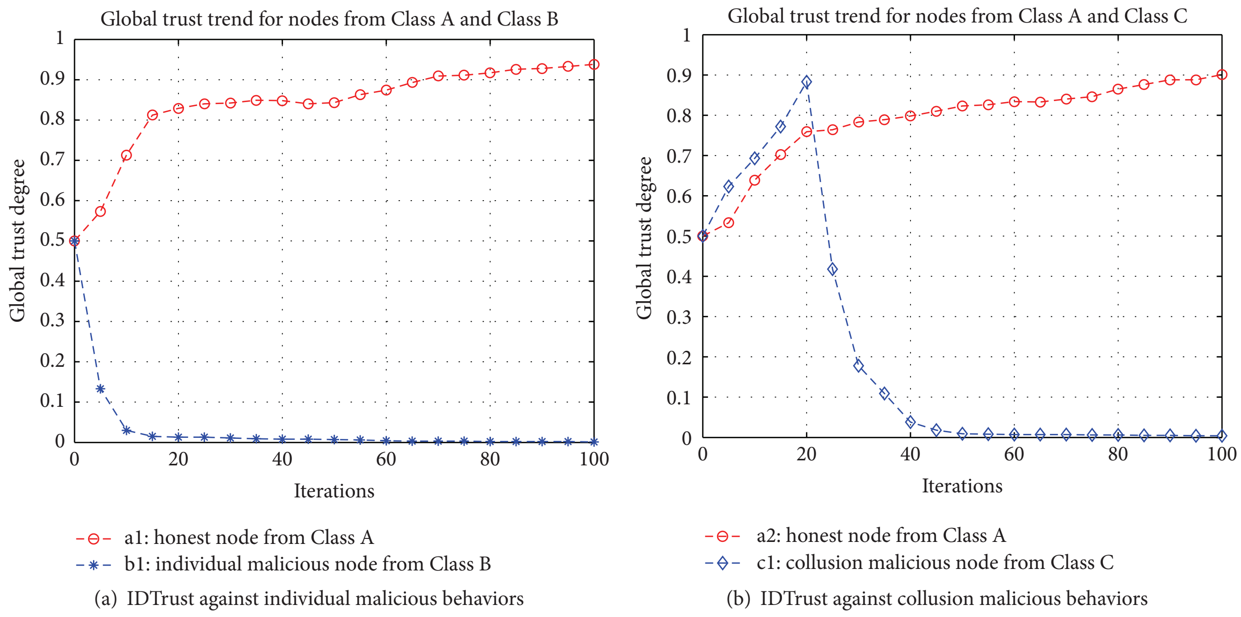
Experiment	Parameter	Allocation
Experiment 1 IDTrust against individual malicious behaviors	A node B node C node	75% 25% 0%
Experiment 2 IDTrust against collusion malicious behaviors	A node B node C node	75% 0% 25%
Experiment 3 Iterative performance *Experiment 4* Trust decision performance	A node B node C node	70% 15% 15%

TABLE 3: Initialized parameters.

Parameter	Initialized value	Description
N	1000	The number of simulated nodes
φ_0	0	Initialized value of user evaluation
λ_0	0	Initialized value of successful transaction rate
ν_0	0	Initialized value of transaction value
sim_0	0.5	Initialized value of similarity
iT_0	0	Initialized value of indirect trust value
gT_0	0.5	Initialized value of global trust value
α	1	Initialized value of user's current evaluation
k	0.01	Decay speed
τ	10	Threshold of $\lvert I_{pq}\rvert$
ζ	0.5	Weight of direct trust which could be adjusted by users according to requirement

6.2. Simulation and Analysis

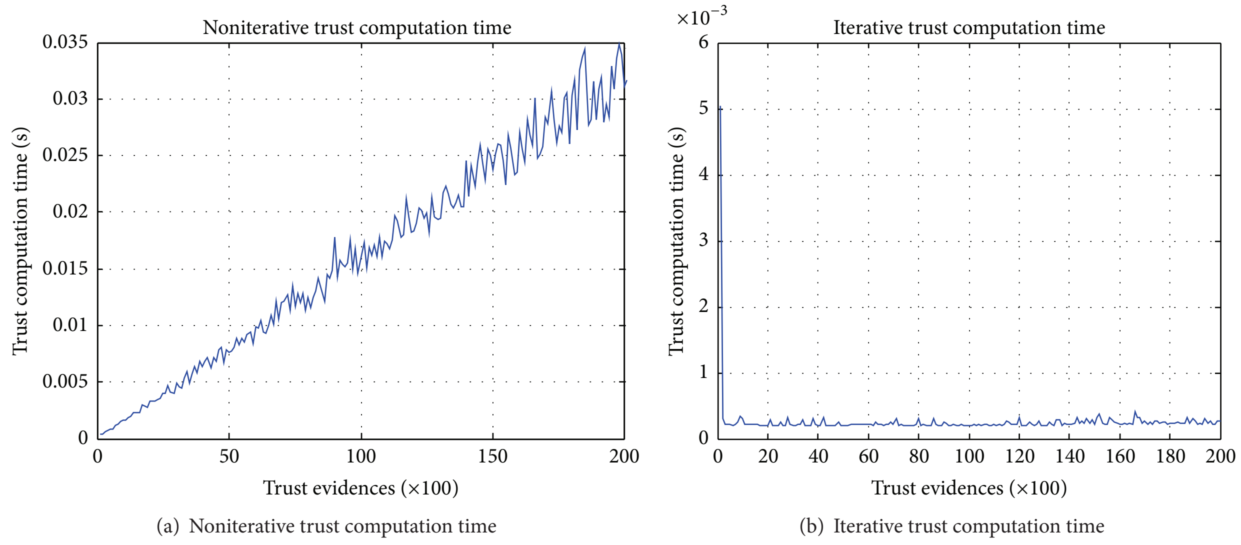
Experiment 1 (IDTrust against individual malicious behaviors). The first adversary model for IDTrust is individual malicious node. We set up almost 25% of individual malicious



(a) IDTrust against individual malicious behaviors

(b) IDTrust against collusion malicious behaviors

Figure 7: Simulation for defending against malicious attacks.



(a) Noniterative trust computation time

(b) Iterative trust computation time

Figure 8: Iterative computation performance comparing with noniterative computation.

nodes of Class B and 75% of normal nodes. From observation of IDTrust, global trust value of designed honest node of Class A and individual malicious node of Class B during simulation experiment, we can get result as shown in Figure 7(a). With increasing of iterative time, global trust value of observed individual malicious node decays very fast; in contrast, trust value of honest node increases all the way. This means that defense to individual malicious node is effective.

Experiment 2 (IDTrust against collusion malicious behaviors). In this experiment, we set up 25% of collusion malicious nodes of Class C and 75% of normal nodes. We want to see whether IDTrust can defend against collusion malicious behaviors. Figure 7(b) shows results. At the beginning, IDTrust cannot distinguish and recognize collusion malicious node due to historic set of public transactions. With iterative calculation increasing, the trust value of collusion node degraded drastically to almost zero. It indicates the similarity in IDTrust works well and IDTrust is effective to defend collusion behaviors.

Experiment 3 (performance of iterative calculation). In order to verify the efficiency of iterative calculation in IDTrust, we count the computation time cost of decision trust computation statistically in both iterative and noniterative calculations separately. In this experiment, we set up 15% of collusion malicious nodes, 15% of individual malicious nodes, and 70% of normal nodes. Figure 8(a) shows the result of noniterative

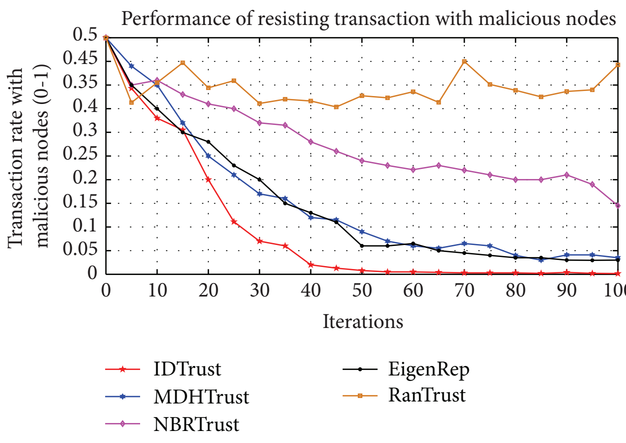


Figure 9: Trust decision performance comparisons.

calculation time cost while Figure 8(b) is about iterative calculation in IDTrust. As we can see from the comparison, the IDTrust with iterative calculation improves computation performance greatly, and its time cost is much smaller than that of noniterative one. The experiment indicates that architecture of IDTrust is highly efficient.

Nevertheless, this iterative performance still occupied CPU and memory utilization. During the simulation, the CPU (Intel Core i7-4980HQ@2.8 GHz) utilization is almost 60% and the memory (4 G DDR3 RAM) utilization is 73% or so. But this disadvantage would be helpful for our further study.

Experiment 4 (performance of trust decision). Finally, in this paper, we compare performance of trust decision of IDTrust with EigenTrust [12], MDHTrust [35], NBRTrust [24], and random model with same nodes proportion as Experiment 3. In this experiment, we count the number of malicious nodes of Class B and Class C which are selected to be trustees by normal nodes of Class A. Also, we define rejectRate to denote to what extent the normal nodes reject services against malicious nodes:

$$\text{rejectRate} = \frac{\text{Count of transacted malicious nodes}}{\text{Count of all transacted nodes}}. \quad (14)$$

Experiment result is shown in Figure 9. As we can see, IDTrust has rapid convergence to reject transactions with malicious nodes since it has strict trust decision processes as described in Section 5.4. The random trust model has relatively lowest performance for it has no trust decision definition. This experiment indicates that IDTrust is very sensitive to malicious node and has an advantage of recognizing malicious behavior.

7. Conclusions

In this paper, we propose a novel dynamic trust model with iterative computation for P2P systems, according to dynamics and fast responding characteristics in large scaled P2P network, named IDTrust.

Firstly, we propose a distributed computing architecture named IDTrust-Arch to obtain trust evidences and make trust decision, including three computing modules, EC, TC, and original P2P module, which are independent but communicate with each other, to get higher computation performance. This architecture separates the trust computing task from P2P system, while traditional trust computing is a binding service and based on universal set of evidence facts to obtain global trust degree of a given node.

Secondly, we propose an iterative computation method to model trust evidence vector and calculate trust in IDTrust. This design degrades the trust computation cost and improves the computation performance. Based on the above, IDTrust is proposed. Trust evidence vector in IDTrust includes both explicit and implicit trust evidences to improve the evidence integrity. Direct trust, indirect trust, global trust, and decision trust are designed in IDTrust based on explicit transaction evaluation and implicit evidence like transaction value, successful rate, decay factor, and similarity factor.

Finally, we design a simulation system based on IDTrust-Arch. Simulations against both individual and collusion malicious nodes prove that the IDTrust is right and efficient against the two adversary models. Simulations about iterative computation and trust decision prove that IDTrust has high computation performance and more efficient trust decision performance.

Conflict of Interests

The authors of this paper declare that they have no conflict of interests.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant no. 61402097 and no. 61572123; the National Science Foundation for Distinguished Young Scholars of China under Grant no. 61225012 and no. 71325002; the Specialized Research Fund of the Doctoral Program of Higher Education for the Priority Development Areas under Grant no. 20120042130003. The authors also acknowledge Ayebazibwe Lorretta's help in doing the grammatical revision.

References

[1] H. Esaki, "A consideration on R&D direction for future Internet architecture," *International Journal of Communication Systems*, vol. 23, no. 6-7, pp. 694–707, 2010.

[2] J. Duarte, S. Siegel, and L. Young, "Trust and credit: the role of appearance in peer-to-peer lending," *Review of Financial Studies*, vol. 25, no. 8, pp. 2455–2484, 2012.

[3] R. Emekter, Y. Tu, B. Jirasakuldech, and M. Lu, "Evaluating credit risk and loan performance in online Peer-to-Peer (P2P) lending," *Applied Economics*, vol. 47, no. 1, pp. 54–70, 2014.

[4] E. Lee and B. Lee, "Herding behavior in online P2P lending: an empirical investigation," *Electronic Commerce Research and Applications*, vol. 11, no. 5, pp. 495–503, 2012.

[5] Z. Jie, "A survey on trust management for VANETs," in *Proceedings of the 25th International Conference on Advanced Information Networking and Applications*, pp. 105–112, Singapore, March 2011.

[6] L. Mekouar, Y. Iraqi, and R. Boutaba, "Peer-to-peer's most wanted: malicious peers," *Computer Networks*, vol. 50, no. 4, pp. 545–562, 2006.

[7] H. Q. Lin, Z. T. Li, and Q. F. Huang, "Multifactor hierarchical fuzzy trust evaluation on peer-to-peer networks," *Peer-to-Peer Networking and Applications*, vol. 4, no. 4, pp. 376–390, 2011.

[8] C. Selvaraj and S. Anand, "A survey on security issues of reputation management systems for peer-to-peer networks," *Computer Science Review*, vol. 6, no. 4, pp. 145–160, 2012.

[9] S. Buchegger and J. Y. Le Boudec, "Performance analysis of the CONFIDANT protocol," in *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc '02)*, pp. 226–236, New York, NY, USA, June 2002.

[10] E. Damiani, S. D. C. Vimercati, and S. Paraboschi, "A reputation-based approach for choosing reliable resources in peer-to-peer networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 207–216, ACM Press, Washington, DC, USA, November 2002.

[11] C. Jia, L. Xie, X. Gan, W. Liu, and Z. Han, "A trust and reputation model considering overall peer consulting distribution," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 42, no. 1, pp. 164–177, 2012.

[12] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The Eigentrust algorithm for reputation management in P2P networks," in *Proceedings of the 12th International Conference on World Wide Web (WWW '03)*, pp. 640–651, ACM, May 2003.

[13] W. Dou, H. M. Wang, and Y. Jia, "A recommendation-based peer-2-peer trust model," *Journal of Software*, vol. 15, no. 4, pp. 571–583, 2004.

[14] L. Xiong and L. Liu, "PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 843–857, 2004.

[15] A. Jøsang, R. Hayward, and S. Pope, "Trust network analysis with subjective logic," in *Proceedings of the 29th Australasian Computer Science Conference (ACSC '06)*, vol. 48, pp. 85–94, Hobart, Australia, January 2006.

[16] A. Jøsang and T. Bhuiyan, "Optimal trust network analysis with subjective logic," in *Proceedings of the 2nd International Conference on Emerging Security Information, Systems and Technologies (SECURWARE '08)*, pp. 179–184, Cap Esterel, France, August 2008.

[17] Y. Wang and A. Nakao, "Poisonedwater: an improved approach for accurate reputation ranking in P2P networks," *Future Generation Computer Systems*, vol. 26, no. 8, pp. 1317–1326, 2010.

[18] R. A. Shaikh, H. Jameel, B. J. d'Auriol, H. Lee, S. Lee, and Y.-J. Song, "Group-based trust management scheme for clustered wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 11, pp. 1698–1712, 2009.

[19] S. Song, K. Hwang, R. F. Zhou, and Y.-K. Kwok, "Trusted P2P transactions with fuzzy reputation aggregation," *IEEE Internet Computing*, vol. 9, no. 6, pp. 24–34, 2005.

[20] Z.-B. Gan, Q. Ding, K. Li, and G.-Q. Xiao, "Reputation-based multi-dimensional trust algorithm," *Journal of Software*, vol. 22, no. 10, pp. 2401–2411, 2011.

[21] J.-T. Li, Y.-N. Jing, X.-C. Xiao, X.-P. Wang, and G.-D. Zhang, "A trust model based on similarity-weighted recommendation for P2P environments," *Journal of Software*, vol. 18, no. 1, pp. 157–167, 2007.

[22] A. Das and M. M. Islam, "SecuredTrust: a dynamic trust computation model for secured communication in multiagent systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 2, pp. 261–274, 2012.

[23] L. Qiao, H. Hui, F. Bingxing, Z. Hongli, and W. Yashan, "Awareness of the network group anomalous behaviors based on network trust," *Chinese Journal of Computers*, vol. 37, no. 1, pp. 1–14, 2014.

[24] Z. Tan, H. Wang, W. Cheng, and G. Chang, "A distributed trust model for P2P overlay networks based on correlativity of communication history," *Journal of Northeastern University (Natural Science)*, vol. 30, no. 9, pp. 1245–1248, 2009.

[25] J. Yin, Z.-S. Wang, Q. Li, and W.-J. Su, "Personalized recommendation based on large-scale implicit feedback," *Journal of Software*, vol. 25, no. 9, pp. 1953–1966, 2014.

[26] U. Kuter and J. Golbeck, "Using probabilistic confidence models for trust inference in web-based social networks," *ACM Transactions on Internet Technology*, vol. 10, no. 2, article 8, 23 pages, 2010.

[27] L. Vu and K. Aberer, "Effective usage of computational trust models in rational environments," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 6, no. 4, article 24, 25 pages, 2011.

[28] Y. Wang and M. P. Singh, "Evidence-based trust: a mathematical model geared for multi agent systems," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 5, no. 4, article 14, 28 pages, 2010.

[29] A. B. Can and B. Bhargava, "SORT: a self-organizing trust model for peer-to-peer systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 1, pp. 14–27, 2013.

[30] F. M. Liu, L. Wang, L. Gao, H. X. Li, H. F. Zhao, and S. K. Men, "A Web Service trust evaluation model based on small-world networks," *Knowledge-Based Systems*, vol. 57, pp. 161–167, 2014.

[31] Z. Tan, L. Zhang, and G. Yang, "BPTrust: a novel trust inference probabilistic model based on balance theory for peer-to-peer networks," *Elektronika ir Elektrotechnika*, vol. 18, no. 10, pp. 77–80, 2012.

[32] Z. H. Tan, G. M. Yang, and W. Cheng, "Distributed trust inference model based on probability and balance theory for peer-to-peer systems," *International Journal on Smart Sensing and Intelligent Systems*, vol. 5, no. 4, pp. 1063–1080, 2012.

[33] G. Wang and J. Wu, "Multi-dimensional evidence-based trust management with multi-trusted paths," *Future Generation Computer Systems*, vol. 27, no. 5, pp. 529–538, 2011.

[34] S.-X. Jiang and J.-Z. Li, "A reputation-based trust mechanism for P2P e-commerce systems," *Journal of Software*, vol. 18, no. 10, pp. 2551–2563, 2007.

[35] Z.-H. Tan, X.-W. Wang, W. Cheng, G.-R. Chang, and Z.-L. Zhu, "A distributed trust model for peer-to-peer networks based on multi-dimension-history vector," *Chinese Journal of Computers*, vol. 33, no. 9, pp. 1725–1735, 2010.