Android System Partition to Traffic Data?

Brittany Byrd, Bing Zhou, and Qingzhong Liu

***Abstract*—The familiarity and prevalence of mobile devices inflates their use as instruments of crime. Law enforcement personnel and mobile forensics investigators, are constantly battling to gain the upper-hand at developing a standardized system able to comprehensively identify and resolve the vulnerabilities present within the mobile device platform. The Android mobile platform can be perceived as an antagonist to this objective, as its open nature provides attackers direct insight into the internalization and security features of the most popular platform presently in the consumer market. This paper identifies and demonstrates the system partition in an Android smartphone as a viable attack vector for covert data trafficking. An implementation strategy (comprised of four experimental phases) is developed to exploit the internal memory of a non-activated rooted Android HTC Desire 510 4g smartphone. A set of mobile forensics tools: AccessData Mobile Phone Examiner Plus (MPE+ v5.5.6), Oxygen Forensic Suite 2015 Standard, and Google Android Debug Bridge adb were used for the extraction and analysis process. The data analysis found the proposed approach to be a persistent and minimally detectable method to exchange data.**

***Index Terms*—Android forensics, factory reset, system partition, AccessData MPE+, oxygen forensic standard suite, android debug bridge.**

I. Introduction

It is estimated that roughly more than 1.08 million of the world's population use smartphones [1]. And Android is now the most popular operating system platform in the world. Hence, Android forensics (a subset of mobile forensics) has been introduced to mitigate these challenges and offer the digital forensics community support – with the key areas of concern listed as the how, what, and wheres of data storage [2]. There exists work that demonstrates and classifies anti-forensics tools and techniques (e.g., data destruction, data hiding, counterfeiting data, and more) against the capabilities of digital forensics tools [3]-[10].

Android smartphones use NAND flash memory to store persistent data on the device (e.g., the operating system, applications, and user data). Investigators are painstakingly aware that the internal flash memory of a mobile device can be extracted and examined with the use of forensics tools, and that despite its non-volatile nature this data can be arbitrarily erased at the discretion of the device user via the factory reset feature. However, not all areas of the internal flash memory (e.g., the boot and system partitions) are affected by the application of the factory reset on the device, and provide fundamental operations mandatory to function the smartphone... so why not target one of these core partitions for manipulation?

The purpose of this work sought to demonstrate that legitimate and marginally traceable means can be used by attackers to exchange data (nefarious or otherwise) through the openness offered by the Android mobile platform.

The following contributions were made:

- An implementation strategy was proposed that exploits the system partition of an Android smartphone, to offer a persistent and undetectable method to exchange data.
- Highlighted a potential attack vector inherent to the openness presented within the Android platform, concerning superuser privileges.
- Identified the need for mobile forensic tools to better manage the vast amount of data stored on Android smartphones, to enable data detection anomalies with precision.

The remainder of this paper is organized as follows:

Section II focuses on the broad topic of android forensics, Android structure and internalization, extraction tools, Android security and more. Section III outlines the implementation strategy created to undermine the removal and recovery of the data intentionally written to the system partition. Significant findings are presented in this section. Lastly, Section IV offers a brief conclusion.

II. Overview of Android Forensics

A. Android Forensics and Android Structure

"Android is a fast growing, feature-rich, and exciting mobile platform. The combination of features, connectivity, and popularity naturally lead to a growing need for Android forensics" [2]. Android forensics is a subset of mobile forensics that focuses predominantly on challenges concerning data storage and acquisition methods specific to the Android mobile device platform [2], [11].

The Android architecture primarily includes the operating system, a middleware, and set of applications. The Android mobile platform is comprised of core components that are consistent, regardless of device type or version that support features common among Android devices, and enable users to store data externally via SD card or internally using NAND flash [2].

1) NAND flash

NAND flash will be discussed in detail, as gaining superuser privileges to the Android device permits users to access the internal memory of the device and exploit the system partition.

Manuscript received August 2017; revised November 2017. This work was supported in part by the Sam Houston State University (SHSU) Office of Research and Sponsored Programs, the SHSU College of Sciences and the SHSU Department of Computer Science.

The authors are with the SHSU Department of Computer Science, Huntsville, Texas, USA (e-mail: bas050@shsu.edu, bxz003@shsu.edu, liu@shsu.edu).

doi: 10.18178/ijke.2017.3.2.084

An Android device is comprised of two primary types of memory: random-access memory (RAM) and NAND flash memory that are built into a single component referred to as the multi-chip package (MCP) to manage data stored in the device [12]. The RAM is volatile memory so its state cannot be sustained following a power lost or reboot, making it a poor location to insert data that requires persistence. Whereas, NAND flash memory is nonvolatile so its data is saved after the device has been powered off or rebooted, supporting persistence. NAND flash is responsible for storing the bootloader, the operating system and system files, and large portions of user-generated data [2].

2) Memory Technology Device (MTD)

The MTD provides Android with the necessary Flash Transition Layer (FTL) interacting with the NAND flash. The MTD is represented by partitions [2], [12]:

- boot – the boot partition includes the Android kernel and ramdisk, without the boot partition the device will not boot.
- system – the system partition includes the entire Android operating system: Android GUI and all the core pre-installed system applications (excluding the kernel and ramdisk).
- recovery – the primary purpose of this partition is to perform backup operations. An alternate for the boot partition, the recovery partition allows the device to boot into a recovery mode to perform advanced recovery and maintenance operations.

The partitions outlined below are all affected by the factory reset function and upon its execution all user-generated data and subsequent modifications thereof should be erased from each partition to regain the original factory settings [12], [13]:

- data – referred to as the userdata partition, the data partition includes all user created data (e.g., contacts, SMS settings, installed applications).
- cache – the cache partition stores frequently accessed data and application components. The data stored in this partition is acquired from common used of the device.
- misc – this partition includes miscellaneous system settings in the form of on/off switches.

Mobile forensics tools specifically target the data stored in the NAND flash memory of the mobile device.

B. Extraction Tools

There are a number of mobile forensics extraction tools available in the forensics market – the majority are commercial tools, which can be difficult to obtain given cost and security considerations, the remainder are open source tools. Commercial mobile forensics tools can provide for logical acquisition of thousands of mobile device platforms, while physical acquisition is only offered on a limited basis for specific device platforms [14].

Android forensics techniques are either logical or physical, each with the key goal to avoid altering the original state of the targeted device during the investigative process. Therefore, the majority of Android forensics tools and techniques eliminate the need to detach the flash memory by directly connecting the phone into a forensics hardware tool or a computer running the forensics software, to perform a logical or physical acquisition [2], [11]:

1) Logical acquisition

Logical forensics techniques are limited to the bit-by-bit extraction of allocated (not deleted) data accessible on the file system (e.g., files and directories). A logical acquisition does not require root access, only that USB debugging be enabled. In Android forensics the most common logical technique cannot enable direct access to the file system, but only performs at a more abstract and less-effective level than the traditional logical techniques performed on computers. This limitation prevents logical forensics techniques targeting the Android platform from acquiring *all* allocated data directly from the file system.

2) Physical acquisition

Physical techniques are sub-divided into two main categories: hardware and software, and target the physical storage medium directly, bypassing the file system to gain accessibility to *both* allocated and unallocated (deleted) data. Root access on the targeted device is needed for forensic techniques that acquire physical images, permitting exponentially more data to be extracted from the device [8].

Typically, mobile forensics extraction tools advertise that the following data can be obtained from the mobile device, up to and including [15]: text messages (SMS/MMS), contacts, call logs, e-mail messages, GPS coordinates, photos/videos, web browser history, and calendar appointments.

C. Android Security

The Android platform uses a multilayered security structure that supports the open platform of the Android architecture and is designed to protect the device user. Mostly, Android security mechanisms are predominantly concerned with the installation and permissions setting of applications on the device [16]. Additional security mechanisms concerning data storage exists that both internally and externally limit where on a device, data can be written and sustain persistence against a factory reset – which is defined as the primary method of securely returning the device to its original factory settings, by erasing all device settings, user data, and third party applications [14], [16], [17].

III. Implementation Strategy

The implementation strategy undermines the open nature of the Android market, by gaining superuser permission via root access to write data into the system partition to serve as a covert data trafficking method – see [18] and [19] on how to root a HTC Desire 510.

The strategy consists of four phases. In summary, the objective of Phases One and Two were to identify and gain access to the attack vector located within the targeted device. Phase Three describes how the data was inserted and modified to sustain persistence and minimize detectability. Lastly, Phase Four discusses the purpose in executing the factory reset against the device to further undermine suspicion and gain legality (if desired).

The case study conducted in this paper used the commercial tools: adb application via Android SDK and AccessData MPE+ to provide images that contain allocated

and unallocated data, and the Oxygen Forensics Standard Suite mobile forensics tool to provide a logical image, from the targeted Android smartphone. Provided below is a brief summary of each tool:

Android Debug Bridge (adb) [2], [15]

The Google Android Operating System provides a Software Development Kit (SDK) to assist application developers in communicating with an Android device via USB. The Android Debug Bridge adb tool is one of the available proprietary tools in the SDK. The adb application tool is a client-server executable – a daemon runs in the background per device instance. The adb tool can execute several commands on the Android device.

Oxygen Forensic Suite 2015 Standard [14], [15]

A standard license for personal use of the Oxygen Forensic Suite can be freely obtained upon request from the company. The standard suite can support over 10,350 different mobile devices and platforms. The standard suite installs a client application, Oxygen Agent into the external memory of the phone to extract data from the device onto the computer. The forensics software is equipped with a built-in HEX, text, and multimedia viewer for files. Data parsing, unicode support and data integrity verification are also provided.

AccessData Mobile Phone Examiner Plus (MPE+)

The following MPE+ overview was provided by the vendor [20]: Mobile Phone Examiner Plus (MPE+) is a stand-alone mobile device investigation solution that includes enhanced smart device acquisition and analysis capabilities. MPE+ offers support for over 10,000 mobile devices, platforms and operating systems and provides the following features: deleted data recovery, device password bypass, data carving, and more.

A controlled environment was created using an Android HTC Desire 510 4g smartphone. The smartphone lacked a contract and was not activated during its use as a data trafficking device to avoid stipulations from a telecommunications provider, provide anonymity to the communicating parties, and offer an easy means to both acquire and discharge the device without concern. The TeamWin Recovery Project utility was used to flash the SuperSu.zip binary file as a custom ROM, permitting superuser privileges to access the entire file system on the device. The application of the proposed technique is evaluated in a case study and summarized.

For simplicity purposes, only one trial experiment will be discussed in detail: The "NanumGothic-**Bold**.ttf" trial since the results provided are reflective of all the trials conducted, as shown in Tables III and IV and outlined in the Significant Findings.

A. Phase One — Device Exploration

In this phase a cursory overview of the Android device is completed. Using Oxygen and MPE+ mobile forensics tools, and the adb application a viable attack vector on the device is identified to enable data insertion and hiding. Note: USB debugging must be enabled prior to using any of the referenced tools – see [21].

Exploration of the device via the execution of varied UNIX commands within the adb shell application, identify the system partition – specifically, the */system/fonts* folder as an appealing location to write the hidden message text file, due to its unsuspecting and convoluted nature since more than 50 different font files are observable in the directory.

See [22] for a list of *NIX commands and their functions.

A compilation of the mobile forensics tools: Oxygen Forensic Suite 2015 Standard and MPE+, in conjunction with the adb application were used to extract a logical image of the device and evaluate any limitations concerning the tools data retrieval capabilities as shown in Table I. Use of adb requires prior installation of SDK Manager and adb drivers [23], [24].

TABLE I: MOBILE FORENSICS TOOLS LOGICAL DATA RETRIEVAL CAPABILITIES

	Oxygen	MPE+	adb
Device name	yes	yes	yes
Pre-installed packages	no	yes	yes
/system partition	no	no	yes
List device root status	yes	no	no

B. Phase Two — Device Rooting

In this phase the bootloader is unlocked and root access is gained on the Android smartphone – see [18], [19] if using a HTC Desire 510. Once, superuser privileges have been acquired, manual access to the directories and files on the device is permissible. The system partition can now be remounted from *read only* to *read-write*, and a physical acquisition can now be performed on the device. The mobile forensics tools: Oxygen Forensic Suite 2015 Standard and MPE+, in conjunction with the adb application were used to extract a physical image of the device and evaluate any limitations concerning the tools data retrieval capabilities as shown in Table II.

TABLE II: MOBILE FORENSICS TOOLS PHYSICAL DATA RETRIEVAL CAPABILITIES

	Oxygen	MPE+	adb
Device name	yes	yes	yes
Pre-installed packages	no	yes	yes
/system partition	no	yes	yes
List device root status	yes	no	no
Timestamp	yes	yes	yes
Index ID	no	no	yes
HEX view display	yes	yes	no *adb pull

Note: Use the adb pull command to retrieve data from the device.

C. Phase Three — Data Insertion and Hiding

Fifteen trial experiments were performed that evaluated the application of varied techniques involving data insertion, hiding, and manipulation. MPE+ and adb tools were used to acquire physical images of the system partition to enable forensic analysis and validate the effectiveness of the strategy presented in this paper.

1) Write, size and rename

Data insertion via the adb push command is used to write the 53 byte text file named "secret.txt" generated on the workstation onto the Android device, specifically into the */system/fonts* folder (the phone must be in 'Recovery' mode to

complete this task). A physical acquisition using MPE+ and adb reveal that in comparison to "secret.txt" – files preinstalled into the */system/fonts* directory were larger and had a **.ttf** extension. Hence, modifications were made to "secret.txt" as follows:

- "secret.txt" is padded with X's to increase the file size from 53 bytes to approximately 83KB.
- "secret.txt" was renamed to "NanumGothic-**Bold**.ttf" – a non-existent font file name (reflective of the name of an existing preinstalled font file "NanumGothic.ttf") is created.

Following the modifications, "NanumGothic-Bold.ttf" is written into the */system/fonts* folder via the adb push command.

2) Index ID

Execution of the ls -li command in the adb shell revealed that the preinstalled font files were assigned sequential index ID numbers. In comparison, "NanumGothic-Bold.ttf" failed to satisfy this consistency and was assigned an outlier index ID number. Note: no technique was identified that is able to modify the index ID number and MPE+ does not display index ID numbers.

3) Timestamp

TABLE III: PROBABILITY OF DETECTION — HIGH/MED/LOW/NONE

Trial #	Write	Size	Rename	Index ID	Timestamp *via touch
1	Low-None	None	None	Med-Low	Med-Low
2	Low-None	None	None	Med-Low	Med-Low
3	Low-None	None	None	Med-Low	Med-Low
4	Low-None	None	None	Med-Low	Med-Low
5	Low-None	None	None	Med-Low	Med-Low
6	Low-None	None	None	Med-Low	Med-Low
7	Low-None	None	None	Med-Low	Med-Low
8	Low-None	None	None	Med-Low	Med-Low
9	Low-None	None	None	Med-Low	Med-Low
10	Low-None	None	None	Med-Low	Med-Low
11	Low-None	None	None	Med-Low	Med-Low
12	Low-None	None	None	Med-Low	Med-Low
13	Low-None	None	None	Med-Low	Med-Low
14	Low-None	None	None	Med-Low	Med-Low
15	Low-None	None	None	Med-Low	Med-Low

**Manual examination* consists of actively searching through the targeted device to observe any noticeable discrepancies.

***High** = high probability of detection (90% or greater) when using any forensics tool capable *or* performing a manual examination that displays the disparate data.

***Med** = medium probability of detection (between 11-89%) *only if* using a tool capable of displaying the disparate data; a manual examination did *not* display any disparate data.

***Low** = detection is low to minimal (10% or less) when using a forensics tool *and/or* manual examination.

***None** = zero probability of detection (0%) when using *the provided* forensics tool *and/or* manual examination.

Typically, a file has three main timestamps: 'creation date', 'date modified', and 'last accessed'. A physical acquisition via MPE+ of the */system/fonts* folder after data insertion revealed that data written to the system partition have a 'creation date' – whereas, the preinstalled files do not. MPE+ is able to display all three main timestamps. However, adb only displays one timestamp (either the 'date modified' or 'last accessed'). No technique was identified that is able to modify the 'creation date' timestamp of a file. However, the touch command can change the 'date modified' and 'last accessed' timestamps. The applied modifications were as follows:

The following touch commands were used to change the "NanumGothic-Bold.ttf" text file timestamp from the current date to reflect the timestamp identical to that of the preinstalled font files, respectively:

#touch -m -t 201408131149.00 /system/fonts/NanumGothic-Bold.ttf

#touch -a -t 201408131149 .00 /system/fonts/NanumGothic-Bold.ttf

Table III demonstrates the likelihood of detection, in using each described technique to insert, hide, or manipulate data.

The application of Phase Four serves to provide legality, ensure anonymity of the communicating parties, and sustain persistence through the execution of the factory reset in the Android device. For the purposes of this paper, this phase is included to substantiate the findings demonstrated in Table IV.

TABLE IV: LEVEL OF PERSISTENCE — HIGH/MED/LOW

Trial #	Write	Size	Rename	Index ID	Timestamp
1	High	High	High	N/A	High
2	High	High	High	N/A	High
3	High	High	High	N/A	High
4	High	High	High	N/A	High
5	High	High	High	N/A	High
6	High	High	High	N/A	High
7	High	High	High	N/A	High
8	High	High	High	N/A	High
9	High	High	High	N/A	High
10	High	High	High	N/A	High
11	High	High	High	N/A	High
12	High	High	High	N/A	High
13	High	High	High	N/A	High
14	High	High	High	N/A	High
15	High	High	High	N/A	High

***High** = none of the inserted file content was erased after reset

***Med** = the inserted file content was partially erased after reset

***Low** = all of the inserted file content was erased after reset

D. Phase Four — Factory Reset

The factory reset is applied to evaluate persistence. An attacker seeking to further evade detection and add a layer of legality could simply write the message to the system partition, issue a factory reset on the device, then sell it to a company that recycles electronics. The intended recipient purchases the device, retrieves the message, and destroys the smartphone – and no one would be the wiser, since the smartphone never has to be activated – see [25] on how to execute a factory reset on an HTC Desire 510 smartphone.

In summary, it is favorably presumed that in the event of an investigation, the initial placement of a hidden file in the unsuspecting */system/fonts* directory will substantially minimize detection, but there is no guarantee. The proposed technique presented in this paper, serves to provide a persistent and marginally detectable means to covertly exploit the system partition for use as a message delivery system. In addition, legality by way of selling the smartphone to an electronics recycle/resell company can be established to further undermine the efforts of forensics examiners. The findings support that the techniques used to both establish sustenance and undermine detection (as demonstrated within the case study) have an estimated high success rate.

To sum it up, the detection via a compilation of mobile forensic tools extraction and analysis efforts is possible, yet limited:

- Execution of the touch command is proven to be persistent following a factory reset.
- Modification of a file's size written into the system partition is not detectable.
- Standard edition of the Oxygen Forensic Suite failed to provide a physical image of the rooted device. *May be a direct result of the trial version provided, and not applicable of the forensic tool as a whole.
- Execution of the adb push command to write a file into the system partition is detectable, if a forensics tool displays sequential index ID numbers for the preinstalled files.
- Renaming modification of an inserted file to copy the file extension of a preinstalled file in the */system/fonts* directory is not detectable.
- Application of a factory reset on the Android device did not affect the intentional changes made to the system partition – adb pull command verified that the "secret.txt" content was not altered either.

IV. Conclusion

In conclusion, the implementation strategy proposed in this paper was successful in demonstrating how the system partition in an Android smartphone can be exploited to provide a persistent and mildly detectable message delivery system. The tools and techniques used are readily available and require minimal prior technological knowledge to successfully execute.

The findings identified in this work serve to bring attention to future implications concerning viable means for terrorists to stealthily and successfully traffic data, as a direct consequence of an inability to develop more advanced tools and techniques capable of detecting and analyzing data disparities among the vast amount of data stored within mobile devices.

Acknowledgment

The authors are grateful to Dr. Karen Murff, Ms. Rachel Anderson, Ms. Susan Korn, and Mr. Lee Reiber and Oxygen Forensics for providing the trial versions to substantiate the findings.

References

[1] Smartphone users around the world—Statistics and facts [Infographic]. [Online]. Available: http://www.go-gulf.com/blog/smartphone/

[2] A. Hoog, *Android Forensics: Investigation, Analysis and Mobile Security for Google Android. Waltham,* MA: Elsevier, 2011

[3] I. Sporea, "On the availability of anti-forensic tools for smartphones," *International Journal of Security,* vol. 6, no. 4, pp. 58-64, 2012.

[4] A. Distefano, "Android anti-forensics through a local paradigm," in *Proc. the 10th Annual Digital Forensic Research Workshop (DFRWS '10).*

[5] P. Albano, "On the construction of a false digital alibi on the android OS," in *Proc. the 2011 Third International Conference on Intelligent Networking and Collaborative Systems.* Washington DC.

[6] P. Albano, A. Castiglione, G. Cattaneo, and A. De Santis, "A novel anti-forensics technique for the android OS," in *Proc. BWCCA. IEEE Computer Society,* 2011.

[7] R. Schwamm, "Effectiveness of factory reset on a mobile device," M.S. thesis, Comp. Science Dept., Naval Postgraduate School, California, [Online]. Available: http://calhoun.nps.edu/bitstream/handle/10945/41441/14Mar_Schwamm_Riqui.pdf?sequence=1

[8] G. S. Cardwell, "Residual network data structures in android devices," M.S. thesis, Comp. Science Dept., Naval Postgraduate School, California, 2011. [Online]. Available: http://faculty.nps.edu/cdprince/mwc/docs/THESIS/2011-08_thesisCardwell.pdf

[9] L. Simon and R. Anderson, "Security analysis of android factory resets," in *Proc. 4th Mobile Security Technologies Workshop (MoST).* [Online]. Available: http://www.cl.cam.ac.uk/~rja14/Papers/fr_most15.pdf

[10] K. Munro, Android scraping: Accessing personal data on mobile devices. *Network Security.* [Online]. pp. 5-9. Available: http://www.sciencedirect.com.ezproxy.shsu.edu/science/article/pii/S1353485814701114

[11] J. Lessard and G. Kessler. "Android forensics: Simplifying cell phone examinations," *Small Scale Digital Device Forensics Journal,* vol. 4, no. 1, 2010.

[12] A. Folloder, Digital forensics and file carving on the android platform. [Online]. Available: http://thehebrew.net/Digital%20Forensics%20and%20File%20Carving%20on%20the%20Android%20Platform.pdf

[13] Android partition details. [Online]. Available: http://techblogon.com/wp-content/uploads/2013/02/partition-size-in-android-device1.jpg

[14] V. Vijayan, "Android forensic capability and evaluation of extraction tools," M.S. thesis, Advanced Security & Digital Forensics, Edinburgh Napier University, 2012.

[15] Compare oxygen forensic suite editions. [Online]. Available: http://www.oxygen-forensic.com/en/compare/

[16] S. Hobarth and R. Mayrhofer, "A framework for on-device privilege escalation exploit execution on Android," in *Proc. IWSSI/SPMU 2011: 3rd International Workshop on Security and Privacy in Spontaneous Interaction and Mobile Phone Use, Colocated with Pervasive 2011.*

[17] Android security overview. [Online]. Available: https://source.android.com/devices/tech/security/

[18] Brian, How to unlock the bootloader for the htc desire 510. [Online]. Available: http://briansprojects.net/2014/10/26/how-to-the-unlock-bootloader-for-the-htc-desire-510/

[19] Brian. How to root the HTC desire 510. [Online]. Available: http://briansprojects.net/2014/12/21/how-to-root-the-htc-desire-510/

[20] Mobile phone examiner plus. [Online]. Available: http://accessdata.com/solutions/digital-forensics/mpe

[21] D. Cogen, How to set up ADB/USB drivers for android drivers (updated 7/15/2014). [Online]. Available: http://theunlockr.com/2009/10/06/how-to-set-up-adb-usb-drivers-for-android-devices/

[22] Brian, How to install android fastboot and ADB on windows. [Online]. Available: http://briansprojects.net/2015/06/24/how-to-install-android-fastboot-and-adb-on-windows/

[23] A. Bednarz. HTC desire 510 enable or turn on developer options development & usb debugging. [Online]. Available: https://www.youtube.com/watch?v=2IUCbXRpFB0

[24] Basic UNIX commands. [Online]. Available: http://mally.stanford.edu/~sr/computing/basic-unix.html

[25] Resetting HTC desire 510 (Hard reset). [Online]. Available: http://www.htc.com/us/support/htc-desire-510-cricket/howto/550842.html

Brittany Byrd graduated from the Department of Computer Science of Sam Houston State University with a master's degree in digital forensics. Her research interests include digital forensics and cyber security.

Bing Zhou is an assistant professor in the Department of Computer Science, Sam Houston State University, USA. She received her Ph.D from University of Regina, Canada. Her publications cover various topics on pattern recognition, intelligent security data analysis, soft computing, rough set theory, data mining, and machine learning. She has served as a program committee member of many international conferences and workshops. She is a reviewer of many reputable international journals and an editorial board member of several books.

Qingzhong Liu is currently an associate professor in computer science at the Sam Houston State University. His research interests include multimedia forensics, information assurance, data mining, bioinformatics, and intelligent computing applications.