University of South Australia

Division of Information Technology, Engineering and the Environment

School of Information Technology & Mathematical Sciences

Remote Wiping in Android

Ming Di Leom

Bachelor in Computing (Hons).

A thesis submitted to

University of South Australia

in partial fulfilment of the requirements for the degree of

Master of Science (Cyber Security and Forensic Computing)

Supervisor: Dr. Kim-Kwang Raymond Choo

University of South Australia
Division of Information Technology, Engineering and the Environment
School of Information Technology & Mathematical Sciences

Remote Wiping in Android

By

Ming Di Leom

1 June 2015

Information Assurance Research Group
Building F, Mawson Lakes Campus
University of South Australia
Mawson Lakes, South Australia 5095

© 2015 Ming Di Leom

Chapter Guide

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ADM	Android Device Manager
API	Application Programming Interface
app	Mobile application
eMMC	Embedded MultiMediaCard
Exif	Exchangeable image file format
FTL	Flash translation layer
GB	Gingerbrea, Android version 2.3
ICS	Ice Cream Sandwich, Android 4.0
JB	Jellybean, Android version 4.1-4.3
KK	KitKat, Android version 4.4
NAND	Not AND logic gate, flash memory device storage
OOB	out-of-band, spare area of flash page
RAM	Random Access Memory
ROM	Android OS ramdisk image, not Read Only Memory
USB	Universal Serial Bus

Declaration

I declare that this thesis does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge it does not contain any materials previously published or written by another person except where due reference is made in the text.

Ming Di Leom

1 June 2015

Abstract

Remote wiping allows the device owner to send a remote command to wipe the contents on the lost or stolen device. Previous studies have shown that remote wiping is ineffective in Android devices which allows user data to be recovered. This thesis expands the scope of previous studies by measuring the effectiveness of remote wiping by third-party app. Seven third-party apps are compared against the built-in remote wiping app, Android Device Manager. The preliminary result suggests that some app has less effective remote wiping than what was claimed by app’s creator. The data analysis shows different implementation by each app and capability of each forensic tool.

The user data remnant left after factory reset or remote wipe could still be useful for forensic purpose. However, an investigator still faces another challenge of recovering fragmented image. Instead of recovering image fragments, “thumbnail”, a smaller version of the original image can be targeted. A thumbnail is marginally smaller than the original image and is also less likely to be fragmented. In addition to benchmark test, this thesis also describes thumbnail forensic recovery process for Android devices.

Acknowledgements

First of all, I would like thank my family for all the support they have given me throughout my study. My mere expression of gratitude to them likewise would not suffice.

This thesis would not have been possible without support of my supervisor, Raymond. Raymond has spent a lot of time reading and reviewing my work. Working with Raymond has provided me a valuable opportunity to improve my academic writing.

I would like to acknowledge members of Information Assurance Research Group for their support. Christian for technical knowledge. Ben and Niken for resources especially the forensic tools.

1 Introduction

Computing power has been doubling on average every 1.5 years since 1975, outperforming Moore's Law (Intel 2006). This trend is partly due to ever increasing computing power including mobile devices at lower cost (Markoff 2007; The Economist 2009). Mobile devices are increasingly ubiquitous and no longer used only for making and receiving phone calls. Example usage include receiving and sending email and instant messages, making VoIP calls, taking and uploading of photos and video clips, and finding one's way around using mapping apps, which results in an increasing amount of data (and metadata such as geolocation) stored and transmitted from such devices. Due to the size of such devices, they can be easily lost or stolen. For example, an estimated 150,000 mobile device were reported lost or stolen every year in Australia (AMTA 2011) , and more than 30,000 mobile device were reportedly stolen in London alone at 2013 (Lynn & Davey 2014) .

With the advent of cloud storage services (e.g. Dropbox[1], Apple's iCloud[2], and Google Drive[3]), mobile device users are able to synchronise the data stored on their devices to their cloud storage accounts. There is, however, potential for information leakage should the account be compromised (e.g. the incident involving the compromising of several celebrities' online accounts (Anderson 2012; Greenberg 2014; Gallagher 2014)) or when the device is lost, stolen or compromised (e.g. malware). Physical theft and loss of devices are among the most common cause of data leakage in organizations according to the 2014 study by Verizon (2014). The cost of the hardware and software due to lost or stolen devices is generally less than the direct and indirect cost resulting from the leakage of the information. In a study by Symantec (2012) where 50 mobile devices were intentionally "lost" and then monitored for any access attempt, for example, 96 percent of these devices were reportedly accessed by the finders of the devices (perhaps due to the inherent curiosity of human nature). The study also highlighted the difficulty for device owner to regain possession of the devices as only 50 percent of the "lost" devices in the study were recovered, although the owner's contact information was clearly shown on the device.

To mitigate the issue of data leakage due to lost or stolen device, remote wiping feature has been introduced to modern mobile devices such as iOS, Android, Windows Phone and

[1] https://www.dropbox.com/

[2] https://www.icloud.com/

[3] https://drive.google.com/

BlackBerry devices. This feature allows the device owner to send a remote command to wipe the contents on the lost or stolen device. Such command has been referred to as "kill pill" (Caldwell 2011, p. 8) or "poison pill" (Hansen 2010, p. 3; Burnett et al. 2011, p. 57) in the literature.

The earliest implementations of remote wiping feature were on Blackberry devices (Punja & Mislan 2008; Evers & Johnston 2005) and the now defunct Microsoft Windows Mobile (succeeded by Windows Phone) (Munro 2008; Microsoft 2009; Microsoft 2005) in 2005. It is not surprising as BlackBerry devices are known for their security features and one of the first devices to be approved by government use (Erlichman & Miller 2014; Harauz & Kaufman 2009; Morrison 2005). Remote wiping was introduced to Apple's iPhones in 2009 through a service known as "Find My iPhone[4]" (Ogg 2009). "Find My iPhone" service was initially only available to now defunct MobileMe (replaced by iCloud and discontinued from June 2012 (Mayers & Lee 2011)) subscription. It was not until the release of iOS 4.2 released in November 2010 (Apple 2010) that "Find My iPhone" service becomes a free service (Aomoth 2010). In August 2013, Google introduced remote wiping feature through Android Device Manager (ADM) to devices running Android 2.2 or above (Poiesz 2013). This has become an official feature of Android devices, which was previously only available to Google Apps customer (Zhang 2012) or via a third-party app (e.g. Cerberus[5], SeekDroid[6]).

Remote wipe functionality can be very useful to prevent information leakage when mobile device is no longer in owner's possession due to theft, robbery, or simply misplaced. Given its usefulness, evaluating the effectiveness can help forensic investigator to understand its implication. Any shortcomings identified also can help product vendor as a room of improvement.

1.1 Research Questions

This thesis aims to answer following research question:

- What is the difference between remote wiping apps in terms of removing user data?

If there is any user data left even after factory reset, it could be utilised by forensic investigator. Therefore the secondary question for this thesis is:

[4] https://www.apple.com/icloud/find-my-iphone.html
[5] https://www.cerberusapp.com/
[6] https://seekdroid.com/

- What is the potential user data that can be easily recovered after factory reset/remote wipe?

1.2 Thesis Structure

Chapter 2 discusses the different forensic approaches in acquiring data from Android device, as well as a survey existing literatures on remote wiping and secure deletion. Material presented in this chapter is based on the following publication:

- Leom, MD & Choo, K-KR & Hunt, R 2016, ‘Remote wiping and secure deletion on mobile devices: a review’, *Journal of Forensic Science*s, pp. 1-20, doi: 10.1111/1556-4029.13203.

Chapter 3 compared the effectiveness of seven third-party remote wiping app against Android built-in’s. Chapter 4 describes forensic collection and analysis of thumbnail. Material presented in this chapter is based on the following publication:

- Leom, MD, D’Orazio, CJ, Deegan, G & Choo, K-KR 2015, ‘Forensic collection and analysis of thumbnails in Android’, *Trustcom/BigDataSE/ISPA*, IEEE, pp. 1059-66, doi: 10.1109/Trustcom.2015.483.

Finally, Chapter 5 concludes this thesis and outlines some future work.

2 Background

Material presented in this chapter is based on the following publication:

- Leom, MD & Choo, K-KR & Hunt, R 2016, ‘Remote wiping and secure deletion on mobile devices: a review’, *Journal of Forensic Science*s, pp. 1-20, doi: 10.1111/1556-4029.13203.

The purpose of this chapter is to provide an in-depth understanding of the current state of play. First the different forensic approaches in acquiring data from Android device are discussed. Then existing literatures on remote wiping and secure deletion are surveyed. Once a mobile device has been remotely wiped, the deleted data should be irrecoverable. Since majority of the mobile device found today use flash storage, also known as NAND flash memory (Shin 2005), we then discuss how secure deletion is addressed on this particular type of storage.

Remote wiping is one of several anti-theft methods for mobile device. Other anti-theft methods include remote tracking, data loss prevention (DLP) system deployed in an enterprise environment, and tools that activate self-destruction upon predetermined conditions. In this survey, discussion is limited to system that wipes itself only when the user triggers it.

One common approach to mitigate data leakage is through storage encryption (i.e. encrypting data-at-rest on the device). In mobile device, storage encryption has been available to BlackBerry version 4.0 or above (BlackBerry 2010), Apple iOS since introduction of iPhone 3GS (Teufl et al. 2013; Götzfried & Müller 2013), Windows Mobile 6.5 (Microsoft 2010) and reintroduced back in Windows Phone 8 (Belfiore 2012; Godfrey 2012). Storage encryption is also available to Android device users since version 3.0 (Honeycomb) (Elenkov 2014a; Götzfried & Müller 2014; Elenkov 2014b). Google initially announced that all devices shipped with version 5.0 (Lollipop) would have storage encryption enabled by default (Google 2014a; Timberg 2014). Due to performance issue (Chester & Ho 2014), it is not enabled on every new Lollipop device (2014), despite the optimisations implemented later (Malchev 2014; Franco 2015; Elenkov 2015). Despite the initial announcement, storage encryption was never mandatory for existing device upgrading to Lollipop. So, the Lollipop device used for the experiment, Moto G is not encrypted by default and not enabled.

Prior to version 4.4 (KitKat), encryption key is stored in flash memory and encrypted with weak key derivation function (KDF), PBKDF2 with 2000 iterations only. This made extraction and bruteforcing the encrypted key fairly trivial. KitKat replaced the KDF with scrypt to make bruteforcing more expensive (Percival 2009). On Lollipop, the key is encrypted with master key that is not in flash memory and cannot be extracted even with root access (Elenkov 2014b). This has made unauthorised access to encrypted flash memory much more difficult.

However, no matter how secure is the key, existing law can compel a person to surrender the key (e.g. Australia *Crimes Act 1914 (Cwlth)*; Ockenden & Sveen 2015). There is also misconception that secure deletion is unnecessary when storage encryption is available since all the data is secured. Secure deletion is a condition where "adversary is given access to system but not able to recover the deleted data from the system" (Reardon et al. 2013a, p. 301). However, cryptography is only effective due to computation cost (Storer et al. 2006) given the best cryptanalysis efforts at that point. Given advances in cryptanalysis and technological advances, there is no guarantee that the data resided even in encrypted form, could not be deciphered by an adversary in the near future. Any future vulnerability(ies) in the device hardware or software may be exploited by an adversary to gain unauthorised access to the decryption key. Thus, secure deletion methods that can ensure deleted data removed from storage is still relevant even data-at-rest are encrypted on the mobile device. Therefore, in this literature review, secure deletion methods are also discussed.

For this literature survey, 15 patents and 18 academic publications published in English between November 1999 and June 2014 are located. On remote wiping topic, materials were located using Google Scholar (including Google Patents) and academic databases such as ScienceDirect, ACM Digital Library, IEEE, and Springer using search terms such as *remote (wipe OR wiping), (sanitize OR sanitization) mobile phone, secure (erase OR delete) (flash OR NAND).*

Past surveys on secure deletion (Reardon et al. 2013a; Diesburg 2012) investigated two common types of non-volatile memory, namely; magnetic hard drive and flash memory. The discussion in Section Secure flash storage deletion, however, focuses on flash memory only. Similar to Reardon et al. (2013a), the discussion on secure deletion does not include information deletion, which encompasses searching and removing all traces of some information. There are two approaches to secure deletion, namely; data overwriting or encryption, and physical destruction (e.g. disk crusher, degaussing, incineration) (DON CIO

Privacy Team 2010). The main difference is that the second approach will render the drive unusable; thus, irrecoverable (Hughes & Coughlin 2006) while the first approach does not. The discussion on this thesis is, therefore, limited to the first approach and does not focus on any specific data type such as database or cloud storage.

This chapter is organised as follows. Section 1 outlines the existing approach in forensic acquisition from Android device. Section 2 reviews existing approaches to remote wiping, while Section 3 on secure deletion techniques. Section 4 discusses the limitation of existing approaches on remote wiping, and Section 5 concludes discussion on this section.

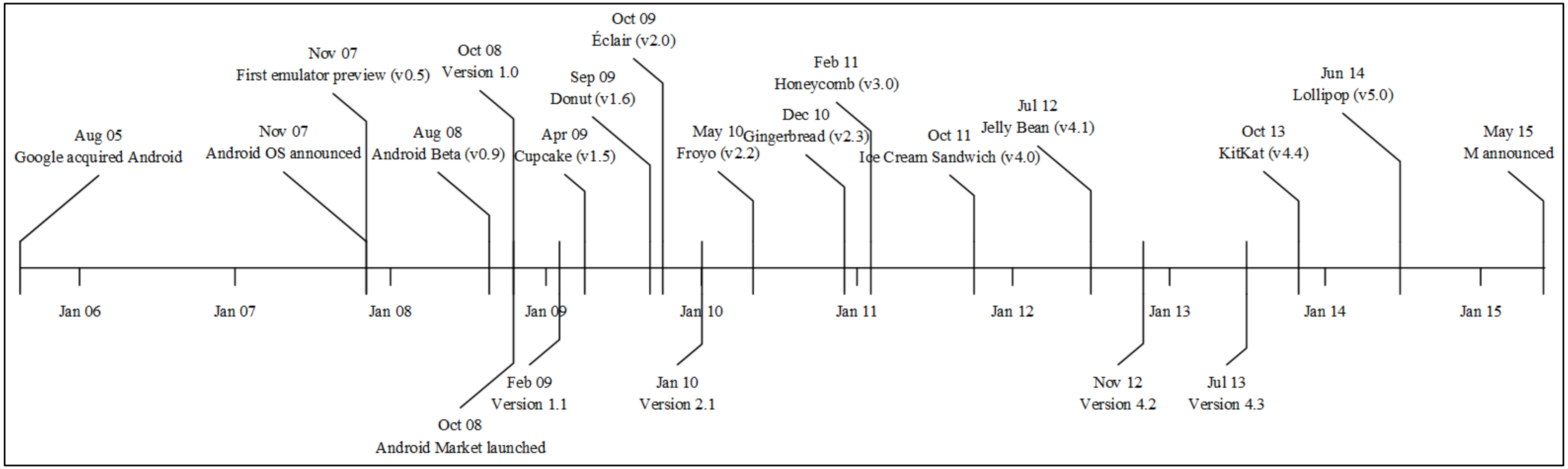


Figure 1: Android version history (adapted from Armadeo, 2014)

2.1 Android Forensics

Name	**Mount point**	**Description**
Recovery	N/A	Recovery mode
Boot	N/A	Linux kernel
System	/system	Operating system files, system apps
Cache	/cache	Cache files
User data	/data	User installed apps
Internal SDcard (Media)	/mnt/sdcard /storage/sdcardX /data/media/X	User-accessible storage to store media files.

Table 1: Partition layout of typical Android device. Adapted from (Vidas, Zhang & Christin 2011).

Android mobile devices typically consist of several partitions (Table 1). Partitioning schemes may differ between Android devices due to vendor customization but there are generally six partitions and each partition stores data specific to a particular function. This information would be useful to a forensic investigator as it allows the forensic investigator to focus only on the relevant partition during the evidence identification process. Majority of user's data is stored in "user data" (/data) partition, internal SDcard, and cache partition.

The recovery partition is used by the Android device to boot into "recovery mode", which is a minimal environment with its own kernel. Recovery mode can be used to wipe user data and update the Android OS since such operations can usually only be performed "offline" due to file locking.

The /boot partition stores the Linux kernel image. This partition is not used during normal system operation, but only during initial boot. The /system partition contains (most of) the Android OS, including built-in apps. This partition is read-only and any new version of the built-in app will be installed in the /data partition. The latter /data partition also stores user personal information (e.g. Google account) and, user-installed apps, and updated version of built-in apps. During a factory reset, both data and cache partitions are formatted and, optionally, the media partition (also known as internal SD card[7]). Generally, the media partition is the largest partition and stores multimedia files (e.g. songs, pictures, and videos), which are accessible from a desktop via a USB connection.

[7] Despite its name, it is not an SD *card*, but a partition instead. Thus, we call it media partition for clarity.

Prior to Android 3.0 (Honeycomb), /data and media are two separate partitions. The media partition generally uses FAT32, which can be mounted and accessed from a host computer through USB Mass Storage (UMS), just like a USB flash drive. The problem with this layout is that /data partition has limited size since majority of storage space is allocated to media partition. This limits the amount and size of apps a user can install. In Android Honeycomb, media becomes a subfolder in /data as */data/media*, not as a separate partition. In this layout, /data partition is allocated with a larger storage space. Since /data partition is in ext4 file system and the file system is not natively supported in Microsoft Windows (requires third-party file system driver, e.g. Ext2Fsd, Explore2fs, Ext2IFS, Ext2Read), a user can no longer mount the storage through UMS. A user needs to access the Android device from a host computer through Media Transfer Protocol (MTP) or Picture Transfer Protocol (PTP). Android 4.2 introduced multi-user support, where each user is assigned a subfolder in /data/media. The default user is assigned to /data/media/0. Each new user is subsequently assigned to /data/media/10, /data/media/11, /data/media/12 and so on.

2.1.1 Physical acquisition

Data acquisition from storage media can be either logical or physical. Logical acquisition copies allocated data that is accessible on the file system. Physical acquisition, on the other hand, copies every single bit of entire storage media or partition including allocated and unallocated data regardless of file system. Typically, whenever a file is deleted, file system simply un-allocate the space used to store the file so it is available to store new data. As physical acquisition through bit-by-bit copy provide access to unallocated space, it is possible to recover deleted data.

There are several techniques for performing physical acquisition from Android devices. The approach is different from traditional desktop environment where the hard disk can easily be removed and connected to forensic workstation for duplication. In mobile device, flash memory is typically soldered onto the printed circuit board (PCB). These techniques as outlined by Hoog (2011c) are divided into two categories:

1. Hardware-based method involves direct access to the hardware component and usually requires some disassembly.
2. Software-based method involves running forensic tools on the device itself.

Hardware-based physical acquisition techniques consist of JTAG and chip-off. Mobile device typically has JTAG test access port (TAP) for manufacturer to test the PCB (Breeuwsma 2006). Flash memory can be directly accessed through JTAG and has been utilised to recover user data from dead or faulty mobile phones (Al-Zarouni 2007). JTAG can be connected through flasher box (e.g. RIFF Box (2014)) which is then connected to workstation to receive data. A major difficulty to JTAG is locating the TAP on the PCB. Connecting to the TAP also requires soldering cable onto the connector PIN. There is also compatibility of flasher box to mobile device because each mobile device has different memory interface in a memory chip (Kim et al. 2007b).

Chip-off is an acquisition technique akin to desktop forensic where physical storage media is removed. In this case, the flash memory chip is de-soldered from the PCB and inserted into memory chip reader, so that the memory content can be extracted. The removal process could damage the memory chip due to high heat of soldering.

Hardware-based technique has the advantage of being least intrusive to the data since no software is running on the mobile device. This property also means root access is irrelevant thus not required compared to software-based. However, there is risk of physically damaging the mobile device, in addition to requiring significant electronic skill and specialised equipment to perform. Interested reader can refer to (Breeuwsma et al. 2007) for more details on the hardware-based techniques discussed.

In software-based technique, the forensic tools must be executed with root privilege in order to access the physical image. By default, user is restricted to reading only allocated data in userdata and media partitions. Data acquisition can be performed either (1) when the system is running (live acquisition) (Aouad & Kechadi 2012; Lessard & Kessler 2010; Lim et al. 2013; Simão, André Morum de Lima et al. 2011) or (2) recovery mode (Vidas, Zhang & Christin 2011; Cannon 2012; Tsai & Yang 2013). Live acquisition is preferred in some situation due to ability to capture volatile data (e.g. network packet and RAM content). Although capturing RAM content is possible in recovery mode, it requires freezing the mobile device and capture the content fast enough to get any valuable data left by OS (Müller & Spreitzenbarth 2013). In addition, RAM acquisition requires loading a custom kernel module that is specific to the Linux kernel version running (Sylve et al. 2012). Every device has different kernel version. OS update could update the kernel as well, so even each device has different kernel version. User also could be installing custom kernel. Although the RAM acquisition is available for Android, but running it is not straightforward.

Rooting is generally considered intrusive due to installation of *su* binary, thus introducing new data. However, in live acquisition, rooting only affects *system* partition, while recovery mode only affects *recovery* partition. Since userdata and media partition are not involved in the preparation, it is unlikely to change user content on the device.

Although user content is not altered during preparation, having a normal system running would use userdata and media partitions. OS can write data to these partitions even without explicit user action. For instance, even when auto-update feature of Play Store has been disabled, OS app like Google Play Service is still updated automatically without user action. This would affect data integrity during acquisition. In contrast, recovery mode is a minimal operating environment that uses only recovery partition and do not depend on other partitions. Thus, when recovery mode is used, data acquisition can be performed with minimal interaction with user data. In forensic cases, live acquisition might run the risk of remote wiping unless the communication can be shielded (i.e. faraday cage). Since any form of wireless communication is not activated in recovery mode, so it is safer in that case.

2.1.2 Custom recovery

The structure of recovery partition is actually similar to boot partition. Both contain bootable image (bootimg) which basically consist of header, Linux kernel, and initial-RAM disk (initrd) (Android Wiki 2013). The header has pointers to locate the kernel and initrd. Once kernel is loaded, the kernel unpack initrd into RAM. initrd is a compressed archive file which contains program (e.g. *init*) and instruction to continue the rest of booting process. Boot partition simply contains different initrd that has instruction to load the Android OS image while recovery partition’s initrd loads a minimal environment, the “recovery mode”. Reader interested in Android boot process may refer to (Björnheden 2009; Hoog 2011a; Jones 2006).

User can customise the recovery unpack the initrd (after unpack from bootimg) and append additional binary programs (e.g. forensic tools) for more functionality. The bootimg is then repacked back to become a *custom recovery*. Instead of creating it ourselves (Vidas, Zhang & Christin 2011; Son et al. 2013), existing custom recovery Team Win Recovery Project (TWRP 2015) is used which is able to serve the purpose of experiment. TWRP has ADB root shell and include BusyBox binary. BusyBox include common Unix utility tools especially *nc* and *dd* that are used in data acquisition.

Custom recovery can be installed (or *flashed*) to these locations:

1. Recovery partition (default)
2. Boot partition
3. RAM

Custom recovery can be flashed to boot partition as well due to aforementioned similarity with recovery partition. Son et al. (2013) preferred this approach because if the custom recovery fail to load, the device will proceed to boot normally which would affect user data partition. On the other hand, failing to boot *boot* partition would simply halt since in this case, the instruction to boot the ROM in normal boot process has been replaced with recovery mode's. This approach is useful when user is not sure the custom recovery can actually load especially when user customise it from scratch. Besides, the device can enter recovery mode straightaway in this case without manual control since the bootloader loads the boot partition by default.

2.1.3 Bootloader

Most bootloaders support *fastboot* mode which allows for flashing the bootimg from user's machine to the Android device's partition. Alternative to fastboot mode is *download* mode, commonly found in Samsung devices. fastboot also allow booting transient bootimg without flashing to partition, essentially "flashing" it directly to the device RAM. Since there is no data written to the flash memory, it can be considered as least intrusive to the *whole* data content. Even so, there is minimal impact on user data when using any of install location of custom recovery.

In order to ensure device integrity, most of the Android devices shipped with *locked* bootloader, including the mobile devices used in case studies and experiments for this thesis (*test devices*). A locked bootloader will only load or flash bootimg that has been signed by the device manufacturer (Elenkov 2014a), similar concept to UEFI's *secure boot*. So, using a custom recovery requires *unlocked* bootloader. Most of the devices allow for unlocking the bootloader, which removes signature checks. Instruction to unlock is easily found online (Moto G – see (Motorola n.d.), Nexus S - see (Nickinson 2010), Nexus 4 - see (Hildenbrand 2012)). Additionally, do note that unlocking bootloader will trigger factory reset (Google n.d.a). This is to prevent unauthorized access to user data as unlocked bootloader can grant more access.

In addition to triggering factory reset, unlocking bootloader might not even possible in some devices. There is another alternative approach to rooting a device that without unlocking bootloader or installing custom recovery, by running root exploit. This approach is similar to iDevice's *jailbreaking*. However this method is not very reliable as not all devices is affected by certain vulnerabilities. Vulnerability also already been patched through software update although it might not be frequent.

2.1.4 Transferring physical image

Aside from data acquisition approaches, there are two methods of saving the acquired physical image:

1. Place an SD card into the device, mount, and save it there. The content is then transfer to workstation through SD card reader. (Aouad & Kechadi 2012; Lessard & Kessler 2010; Simão, André Morum de Lima et al. 2011)
2. Use ADB port forward to create a network between the Android device and the workstation over USB.

SD card method is not applicable to the test devices because none of them has SD card slot. ADB method also has the advantage of transferring physical image directly to workstation. Android Debug Bridge (ADB) consists of software component on the USB-connected Android device (ADB *daemon*) that can communicate with another ADB instance on a workstation (ADB *client*). ADB is usually utilised for debugging purpose. The interactive remote shell and TCP port forwarding functionalities can be utilised for data acquisition purpose. ADB port forwarding essentially connects TCP port on the Android device and workstation. Any input to the device's TCP port will be sent over via USB connection to workstation's TCP port and vice versa.

The overall process of data acquisition using ADB method is as such, a data dumping utility (*dd*) reads binary content of storage media and output to a network utility (*netcat*/*nc*) connected to ADB TCP port on Android device. Another netcat instance connected to ADB TCP port on workstation receives the data and the data is then saved. This process is reversed when restoring data.

In addition to dd, nanddump is another data dumping utility. nanddump is able to capture spare area also known as out-of-band (OOB) area of flash memory which is not reachable by dd. nanddump is only applicable in MTD. There are two methods of handling data access in flash memory: Memory Technology Device (MTD) and Flash Translation Layout (FTL). Before

Android version 3.0 (Honeycomb), file system accesses the flash storage through MTD (Woodhouse 2008). MTD is simplistic and does not offer garbage collection and wear-levelling. Yet Another Flash File System 2 (YAFFS2) is one such file system that has these features (Zimmermann et al. 2012). YAFFS2 identify *dirty* pages[8] (or *garbage*) by writing metadata to OOB. Since dd could not acquire the metadata of YAFFS2, nanddump is necessary to be able to reconstruct the YAFFS2 later from the acquired image (Hoog 2011b; Quick & Alzaabi 2011). However, since Android version 3.0, MTD interface has been replaced by embedded multi-media card (eMMC) and secure digital (SD) (Skillen & Mannan 2013). Both eMMC and SD have integrated FTL implemented in the hardware controller and has built-in garbage collection and wear levelling. This allows common block file system (e.g. ext4 and FAT32) that was not designed for flash memory to operate properly[9]. YAFFS2 becomes obsolete due to built-in features of FTL. Since OOB is not used by EXT4 nor F2FS, dd is sufficient for physical acquisition.

Discussion on Android forensic so far is summarised in following diagram:

[8] A block is made up to multiple pages. Page is the smallest unit of I/O operation in flash memory. OOB is located at the end of each page.

[9] MTD does have software-based FTL to emulate block device so block file system can operate. However, it is not ideal due to aforementioned limitation of MTD.

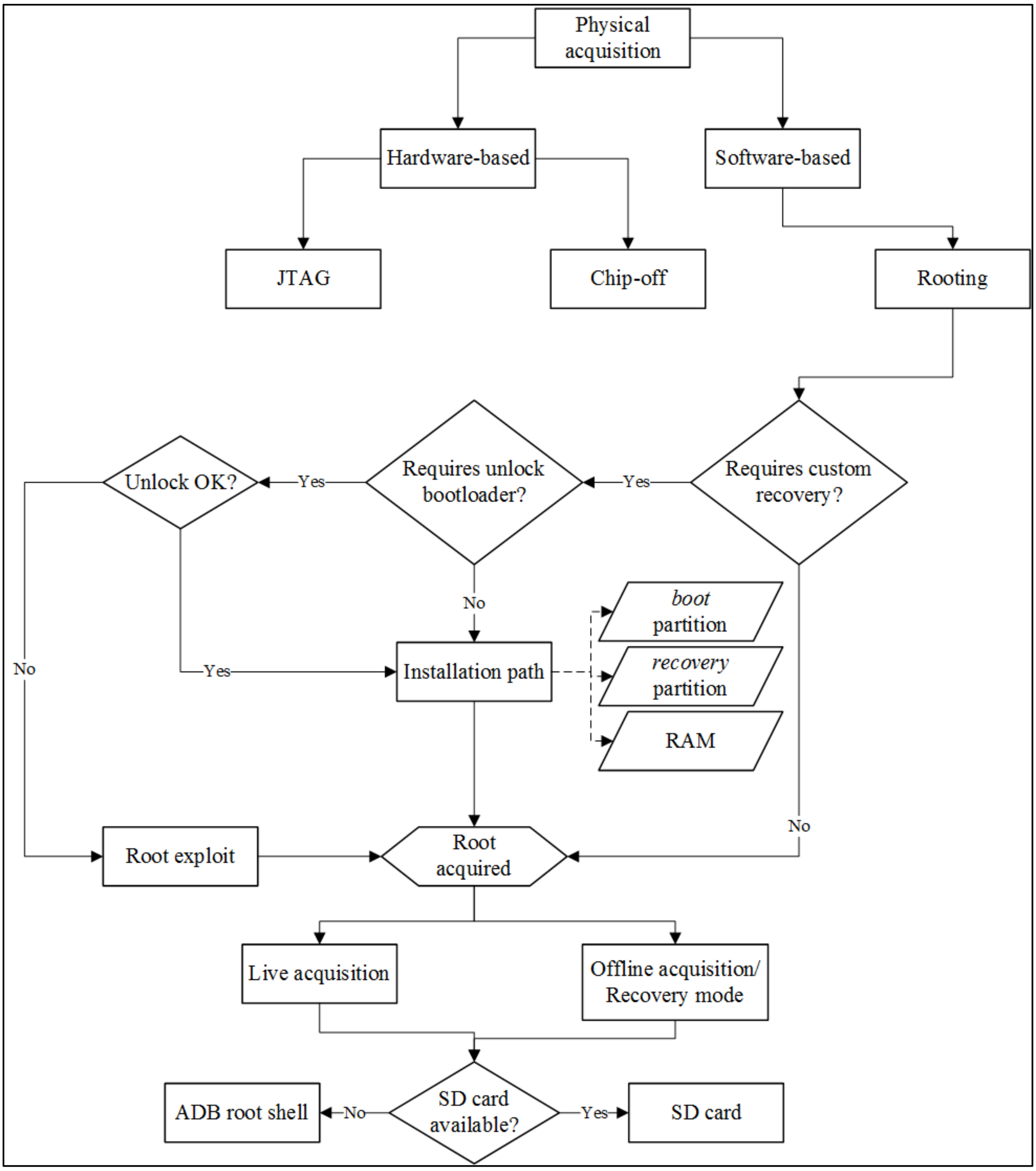


Figure 2: Android physical acquisition approach.

To demonstrate the different approaches to physical acquisition in Android, Chapter 3 would be using *recovery mode*, while Chapter 4 would be using *live acquisition* approach.

2.2 Remote wiping

In this section, located publications on remote wiping – 8 patents and 5 academic publications are discussed.

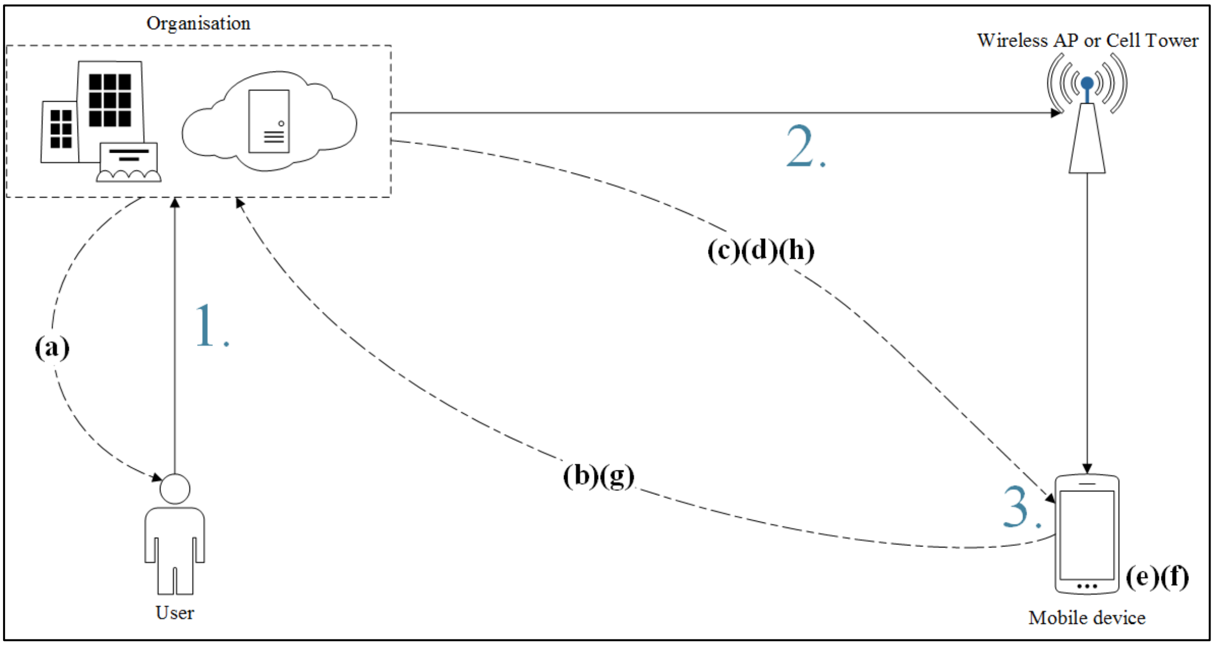


Figure 3: Remote wiping process and security considerations in remote wiping process.
Legend: (a) Authenticate reporter
(b) Authenticate origin of wipe command
(c) Secure wipe command
(d) Transmission channel
(e) Secure delete
(f) Ensure wiping operation is completed
(g) Acknowledge source that wipe is completed
(h) Replay attack mitigation

Remote wiping process (Figure 3) generally can be described as follow:

1. User enrols into the remote wiping system maintained by an organisation. When the mobile device is lost, the user reports it to the same organisation and request for the mobile device to be wiped.
2. The organisation sends the wiping command to the intended mobile device. Depending on the transmission channel, the command may be sent through wireless access point (AP) in Internet connection, or cell tower in telecommunication channel (e.g. 3G/4G).
3. Upon receiving the wipe command, the mobile device erases the data.

Feature	(Angelo et al. 2003)	(Brown et al. 2011)	(Walker & Fyke 2013)	(Kenney 2005)	(Hasebe 1999)	(Sennett & Daly 2013)	(Onyon et al. 2007)	(Gajdos & Kretz 2006)	(Yu et al. 2014)	(Park et al. 2011)	(Kuppusamy et al. 2012)	(Joe & Lee 2011)	(Adusumalli 2014)
Authenticate reporter	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	N
Authenticate origin of wipe command	Y	Y	N	N	N	Y	Y	Y	Y	N	Y	N	N
Secure wipe command	N	Y	N	N	N	N	N	N	N	N	N	N	N
Transmission channel	Internet	Internet	Cellular	Cellular	Internet	Cellular	Internet	Internet	Cellular (Emergency call)	Cellular (SMS)	Cellular (SMS)	Internet	Cellular (SMS)
Secure delete	N	Y	N	N	N	Y	Y	Y	Y	N	N	N	N
Ensure wiping operation is completed	N	Y	N	N	N	N	N	N	Y	N	N	N	N
Acknowledge source that wipe is completed	N	N	N	Y	N	N	N	Y	N	N	N	Y	Y
Replay attack mitigation	Y	N	N	N	N	N	N	N	N	Y	N	Y	N

Table 2: Security considerations mentioned in existing literature (illustrated in Figure 3).

2.2.1 Authenticate reporter

System should verify the identity of the reporter and the device's owner when a mobile device is reported lost or stolen. This can be done either through information known only to the user (e.g. account number, address, and last bill number). Existing authentication methods considered by the literature are summarised in Table 3.

	Biometrics	Certificate	Password	Secret question	Username and password	Identification information
Angelo et al. (2003)	●			●		
Brown et al. (2011)						
Walker & Fyke (2013)			●			
Kenney (2005)				●	●	
Hasebe (1999)			●			
Sennett & Daly (2013)					●	
Gajdos & Kretz (2006)					●	
Yu et al. (2014)			●			●
Park et al. and Kuppusamy et al. (2011; 2012)			●			
Joe & Lee (2011)		●			●	
Adusumalli (2014)			●			

Table 3: Authentication methods

Brown et al. (2011) considered only the authorisation level to determine the data type that the reporter or any person who initiate the remote wipe, allowed to wipe. The proposed scheme did not indicate any means of authenticating the identity of the reporter.

Several high-profile hacks (Anderson 2012; Gallagher 2014; Honan 2012a; Zetter 2008) have shown using secret question or personal identifiable information alone is not sufficient to authenticate a person. The victims' account was compromised because an adversary is able to answer secret question by supplying publicly available information deduced from the Internet. Those incidents could have been prevented with two-factor authentication.

Nearly half of the proposals specified more than one mechanism of authenticating the reporter but they did not consider multi-factor authentication. Of those proposals, only Yu et al. (2014) proposal include two-factor authentication even though it is not explicitly indicated in the original proposal. In Yu et al. (2014), the authentication process involves two steps; first reporter submits personal identifiable information to identify himself to the service provider, then he provides the PIN code to be verified by the mobile device. Thus, it can be considered as two-factor authentication, even though the authentication is performed by two entities; service provider and mobile device.

2.2.2 Authenticate origin of wipe command

When receiving the wipe command from any channel (e.g. cell tower or web server), the mobile device should check whether the source is authorised to send the command or instruction.

	Public key cryptography	Shared code / password	Incoming number
Angelo et al. (2003)	●		
Brown et al. (2011)	●	●	
Sennett & Daly (2013)		●	
Onyon et al. (2007)	●	●	
Gajdos & Kretz (2006)	●	●	
Yu et al. (2014)		●	
Kuppusamy et al. (2012)			●

Table 4: Summary of methods used to authenticate origin of wipe command.

Existing literature generally prefer the use of public key cryptography. Pretty Good Privacy (PGP) is one such popular encryption program - a sender first creates a digital signature by hashing the message and encrypts the hash with its private key to *sign* the message. The sender then encrypts the signed message with the recipient's public key. Only the recipient can decrypt the message with its private key and verify the message using the sender's public key.

Onyon (2007) suggested storing the sender's public key when enrolling into the remote wiping system. Brown et al. (2011) and Gajdos & Kretz (2006) assumed the sender's public key has been stored on the mobile device during manufacturing. Angelo et al. (2003) suggested a signature-based approach whereby the sender encrypts the message with the recipient's private key. The signed message can then be decrypted by the recipient using its own public key.

Authentication can also be established using a shared code between the sender and the recipient. In SSL/TLS, for example, a symmetric key is exchanged using public key cryptography. However, Brown et al. (2011), Onyon et al. (2007) and Gajdos & Kretz (2006) did not consider the need to protect the shared code. Without any protection, the shared code could be exposed allowing an adversary to spoof as a valid sender or recipient.

Park et al. (2011) and Yu et al. (2014) proposed a mechanism to request password from the reporter, which will be sent with the wipe command. The mobile device authenticates the password to determine authenticity of the wipe command. But this mechanism could only authenticate the reporter, since if a correct password is provided, then *any* server can send the command, and the mobile device will simply accept. Thus, this mechanism is considered to be more suitable to authenticate the reporter. Additionally, it is important to consider the response mechanism when dealing with incorrect password. If there is no limit to the number of failed attempts, this will allow online brute-force dictionary attack. Therefore, it is recommended for the system to lock the account should the number of failed attempted exceeds a predetermined limit.

Kuppusamy et al. (2012) proposed a system that checks the incoming telephone number of the received SMS against trusted numbers. This is a failback mechanism when the message is not secured through the authentication method proposed by Park et al. (2011). Instead of using as a failback mechanism, the system should always check the incoming number as origin authentication, with trusted number set beforehand. This mechanism can reduce the possibility of replay attack (see Section Replay attack mitigation) unless the number is spoofed (2004), assuming the telephone number is checked separately and not embedded inside the encrypted or encoded message.

2.2.3 Secure wipe command

The “wipe” command sent as a message or packet should be secured against sniffing to avoid tampering. Surprisingly, only Brown et al. (2011) considered encrypting the wipe command despite the potential for the wipe command to be hijacked and modified.

2.2.4 Secure delete

The wiping process should result in the wiped data being irrecoverable, and secure deletion is discussed further in Section Secure flash storage deletion.

Brown et al. (Brown et al. 2011) and Onyon et al. (Onyon et al. 2007) proposed overwriting with zeroes, ones, or random combination of them. Sennett and Daly (2013) proposed permanent physical self-destruction. Gajdos and Kretz (2006) proposed self-destruction by overwriting the firmware. Yu et al. (2014) only mentioned future possibility of incorporating secure deletion solution. Although the patent by Kenney (2005) does not provide secure deletion, it proposed a method to render data stored on the device inaccessible. The latter is similar to the approach undertaken by Apple where data is ‘wiped’ by rendering all files cryptographically inaccessible when the file system key used to encrypt the files is deleted (Apple 2014).

The secure deletion method described in the patent by Brown et al. (2011), owned by BlackBerry, Inc[10] (formerly known as Research In Motion, RIM), is deployed on BlackBerry devices. The pattern or process differs between the OS version, type of storage, and type of data (BlackBerry 2011; BlackBerry 2014c). The difference of “flash storage” and “user files” is that user files refer to a portion of storage that allow user file-level access. Typical, “user files” is used to store media files such as pictures and videos. “Flash storage” in this case include the rest of user’s data[11] (contacts, SMS, e-mail, calendar entries, etc), but does not include the operating system (OS). This level of wiping is also commonly known as factory reset.

2.2.5 Ensure wiping operation is completed

This feature if implemented ensures that the wiping process is completed successfully, even when it is interrupted by switching off the mobile device as the process will resume once the mobile device is switched back on. When a device is switched off, it would be challenging for an adversary to gain access to the data stored on the device even though the wiping process might be interrupted. Only Brown et al. (2011) and Yu et al. (Yu et al. 2014) considered this aspect in their approaches.

2.2.6 Acknowledge source that wipe is completed

The lost mobile device informs the owner or the system operator that the wiping operation has been completed successfully.

Kenney (2005) suggested using handshake, ACK or NACK (Negative ACK, also known as NAK), or ping. These are commonly found in TCP/IP. Sennett and Daly (2013) suggested broadcasting a signal to indicate that the mobile device has received a valid command and will proceed to execute the command. The latter, however, does not specifically check that the mobile device has successfully completed the task. Adusumalli (2014) and Joe and Lee (2011)

[10] http://www.blackberry.com/

[11] For full list of data type, see Blackberry (2011, p. 45)

suggested using SMS and “service complete code” to confirm successful execution of command respectively.

2.2.7 Replay attack mitigation

If an adversary manages to capture the command, the adversary can send the same command to another device to wipe it. In other words, conduct a “Replay attack” by replaying a previous request made by reporter to wipe the new (replacement) device.

Angelo et al. (2003) proposed using timestamp, non-repeating sequence number, and randomly generated number to mitigate replay attack. Similarly, Park et al. (2011) also suggest using a concatenation of the wipe command and the timestamp using Base64 encoding to mitigate replay attack (referred to as reply attack). This measure is, however, ineffective as an adversary is able to decode the message, modify the timestamp so that the timestamp fulfils the requirement, and re-encode the modified concatenated message. It is recommended that the suggested approaches of Angelo et al. (2003) and Park et al. (2011) be deployed using encryption to avoid modification of the wipe command.

2.2.8 Vendor implementation

2.2.8.1 Android

Remote wiping feature was officially introduced to Android via ADM (Poiesz 2013) which was previously only available via a third-party app. ADM is remotely controlled via Google Cloud Messaging (GCM). GCM is a push messaging service used in Android platform that enables developers of mobile apps deliver data to their apps running on their customers’ mobile device. With push messaging, developer can conveniently push notifications, messages and even commands to the apps, without continuous polling from the apps which consume more resource. Developer operates a server (referred as *app server*) to send data to the app installed in particular user’s device. In push messaging, the app server does not directly initiate the connection to the mobile device. Instead, the data is relayed through the provider of push messaging service (referred as *connection server*). In addition, any message received from connection server is processed by service client (akin to OS service) before passing to the app (Figure 4). Connection server usually restricts the data size, and in this case, the mobile device can be instructed to download the data from an app server. Before an app can receive data through GCM, it needs to register itself to the service with a registration ID and also the app server's sender ID. This allows the cloud server to associate an app server with an app installed on a particular mobile device. Thus, only app server with authorised sender ID can push message to app with related registration ID (Li et al. 2014a).

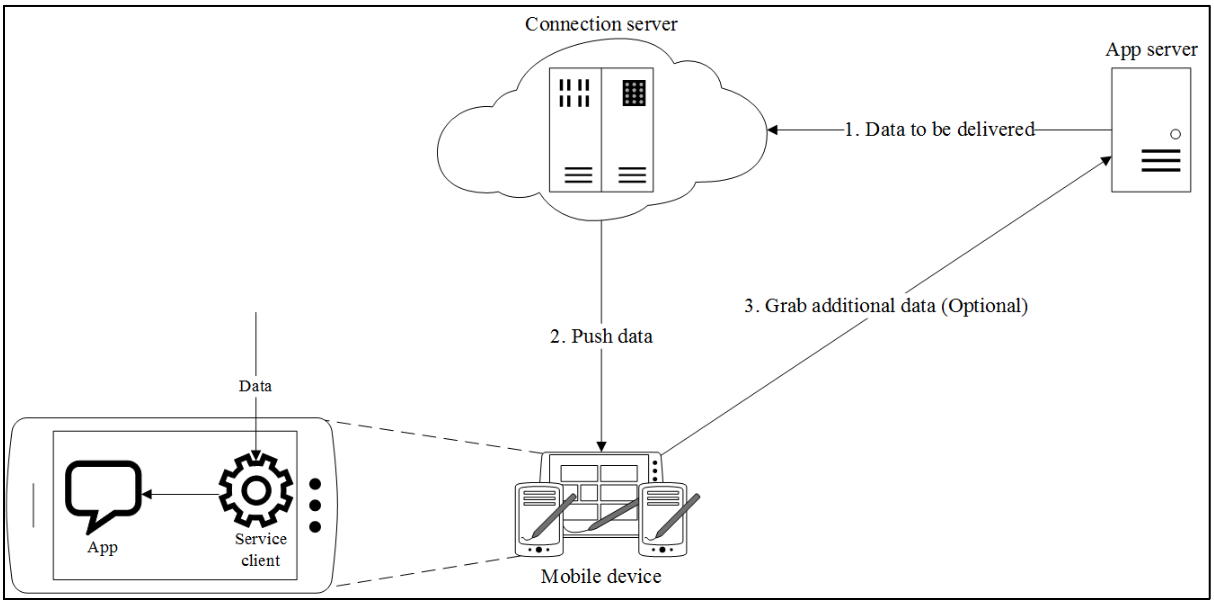
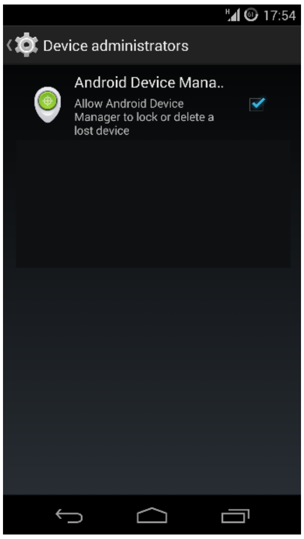


Figure 4: Push messaging architecture. Adapted from ManageEngine (n.d.) *and Li et al.* (2014a)

ADM utilises Android Device Administration API to execute wiping operation (Google n.d.c; Google n.d.d). This API is also available to third-party app to provide mobile device management (MDM) features at the system level. IT administrator can write app that user installs on the mobile device to enrol into MDM system of the company. The API only provides factory reset functionality via "wipe-data" policy, and developer could choose to use GCM or any other method to trigger the wipe remotely. Any app utilising the API must have "device administrator" permission granted (Figure 5).



Device administrators

Android Device Mana..
Allow Android Device Manager to lock or delete a lost device ☑

Figure 5: Device administrator

2.2.8.2 BlackBerry

Remote wiping can be triggered via BlackBerry Enterprise Server (BES) for corporate environment or BlackBerry Protect for personal customer. The device is preloaded with a root certificate during manufacturing, to authenticate BES (BlackBerry 2014b). BES also has the ability to track wiping status of the device (BlackBerry 2014a), a property discussed earlier (Chapter 2.2.6).

Device enrolled in BlackBerry Protect cannot be associated with BES. In BlackBerry Protect, user has the ability to instruct the lost device to perform back-up (to BlackBerry Protect service) before full wiping when running BlackBerry OS version 7.1 or earlier (BlackBerry 2013). Later version does not have the feature.

2.2.8.3 iOS

Remote wiping can be triggered via iCloud (for personal consumer) or third-party mobile device management (MDM) system (for corporate environment). Third-party MDM utilises either Apple's MDM API or Exchange ActiveSync (EAS) (Apple 2014). In third-party MDM system, employee's mobile device is managed from a MDM server installed with software provided by MDM provider. The server creates *configuration profile* (Apple n.d.a) which is uploaded to Apple's server. User then downloads and installs the *configuration profile* to enrol into the system. Configuration profile allows the system administrator to have control over the employee's iOS device including ability to remotely wipe. Subsequently, whenever the MDM server wants to communicate with an iOS device, it does so via Apple Push Notification Service (APNS), a push messaging protocol (ManageEngine n.d.) secured with SSL/TLS

(Apple n.d.b), instructing the device to check in. The iOS device then initiates SSL/TLS connection with the server to check in. The server uses the connection to perform administrative task including remote wiping. According to official Apple (2014) documentation, when performing remote wipe, the mobile device will reply with an acknowledgement upon receiving the wipe command. The device only checks in when using EAS; thus, it appears that the wipe command via a single "push" message is sent by the MDM server through APNS without the device checking in.

2.2.8.4 Windows Phone

Remote wipe can be initiated via Exchange Management Console (EMC)[12], Microsoft Outlook Web Access (OWA), or third-party MDM system depending how the mobile device was enrolled initially. The wipe command is sent via ActiveSync protocol. ActiveSync is a push messaging protocol used for exchanging messages in Microsoft Exchange environment (Microsoft 2013).

The mobile device can either be partially wiped or fully wiped. Partial wipe applies whenever the device "un-enrols" or "retires" from the corporate MDM system. All the corporate information, email accounts, VPN connections, Wi-Fi connections, policy settings, apps, and data that the apps deployed are removed, except for personal apps or data on the device that the user installed. Full wipe removes all the apps and information on a device and returns the device to factory settings. (Microsoft 2014)

2.2.8.5 Vulnerabilities

Implementations of ActiveSync protocol in Android and iOS were discovered to be flawed (Hannay et al. 2013). The implementations failed to warn the user when presented with untrusted SSL certificates and in some cases, accept any certificate presented to it. This vulnerability can be exploited to spoof an Exchange server to initiate unauthorised policy enforcement such as performing remote factory reset on a mobile device. Similar flaw was also discovered in some Android apps due to incorrect use of Android API when implementing SSL, and consequently, these apps are insecure against man-in-the-middle (MitM) attack (Hubbard, Weimer & Yu Chen 2014; Fahl et al. 2012). SSL certificate validation was also found to be broken in Amazon's EC2 Java library (Georgiev et al. 2012) and Apple's

[12] http://technet.microsoft.com/en-us/library/bb123762%28v=exchg.141%29.aspx

SecureTransport library (Ducklin 2014). These flaws highlighted the potential risk posed by SSL implementation in non-browser application, which has not evolved as much as Web browser’s implementation (Georgiev et al. 2012). Such risk could extend to remote wiping app that utilises SSL.

There was a flaw previously found in Google Cloud Messaging (GCM) that allowed an adversary to control the ADM installed on the victim’s device (Li et al. 2014a). As mentioned in Section Android, before an app can receive message through GCM, it needs to register itself to the service with a registration ID and also app server’s sender ID. This allows connection server to identify which mobile device and app to push message to. With a malicious app installed on the victim’s device which acts as a man-in-the-middle (MitM), an adversary can intercept the registration request and steal the registration ID, in this case registration ID of ADM. The adversary can then proceed to control the ADM with the stolen registration ID. In addition, connection server is supposed to only allow authorised app server to push message by checking sender ID. However, this policy is not enforced and hence, allowing an adversary to push message from his “unauthorised” app server (Li et al. 2014b).

Apple’s iCloud allegedly allowed unlimited password attempts, and thus, is vulnerable to brute-force attack (Greenberg 2014). This vulnerability was apparently responsible for compromising several celebrities' iCloud account that ultimately leads to their photo leaks (Greenberg 2014). Proof-of-concept code to launch brute-force attack on iCloud service was published on August 30, 2014[13]. The vulnerability was apparently fixed two days later (Kingsley-Hughes 2014). Apple denied that the incident was due to vulnerability of iCloud service, and instead claimed that it was a result of “very targeted attack on user names, passwords and security questions,” (Gallagher 2014). Although the photo leaks incident was about exposure of private data, it could have been abused to erase victim’s data stored on their mobile devices.

Another recent high profile incident involves the unauthorised reactivation of iOS devices which were locked using the “Activation Lock” feature (Pagliery 2014). This feature is introduced to deter theft by rendering stolen device unusable. Unauthorised reactivation is performed by redirecting the iCloud connection to spoofed iCloud server. The server will then send unlock command to the locked device. Normally, reactivation requires sign-in to the iCloud account associated with the mobile device. The existence of spoofed iCloud server

[13] https://github.com/hackappcom/ibrute

suggests that “Activation Lock” command might have been successfully reverse-engineered. Such feat could also extend to replicating remote wipe command, at least in theory.

Cross-site request forgery (CSRF) (OWASP 2014) vulnerability was discovered in Samsung’s “Find My Mobile”. This vulnerability is identified as CVE-2014-8346 (US-CERT 2014). The vulnerability allows unauthorised remote locking of mobile device by sending specially crafted link with embedded remote lock command to the victim. Assuming that the victim is logged on to “Find My Mobile” service, when a victim clicks on the malicious link, the web browser would send a remote lock request, in which the service proceeds to lock victim’s mobile device. The attacker can customise the link to lock and change the unlock PIN, resulting in the victim not able to unlock his mobile device.

The implications of unauthorised remote wipe is potentially damaging with increasing reliance on digital devices in modern society. For example, a journalist described that his “entire digital life was destroyed” when his online accounts were compromised (Honan 2012a). He reportedly lost “more than a year’s worth of photos, emails, documents, and more.” after an attacker remotely wiped all his devices, and could not “send or receive text messages or phone calls” after his Google Voice account was deleted (Honan 2012b). His accounts were compromised due to weakness in the password reset policy adopted by customer service representative. Such incidents highlighted the heterogeneity of threats in online services, and the attack vectors are not restricted to web application flaws (OWASP 2013) or technical vulnerabilities.

2.2.9 Summary

This concludes survey of existing proposals on remote wiping for this chapter. Existing proposals generally do not consider securing the wipe command nor provide any mechanism to automatically resume interrupted wiping process. Secure deletion is seldom used on mobile devices. “Factory reset” serves as a simple method to remove all user data from mobile device. However, studies have shown that factory reset does not sufficiently remove personal data from mobile devices (Simon and Anderson 2015a; Schwamm 2014; Schwamm and Rowe 2014; McColgan 2014; The Guardian 2013; Siciliano 2012; Honan 2013). Factory reset typically just logically delete data, leaving data residue that could be forensically recovered. Secure deletion seems to be a clear solution to this issue, but it is actually not that straightforward due to the use of flash storage in mobile device. Various challenges of secure flash storage deletion and how existing proposal attempts to overcome them shall be examined in the next section.

2.3 Secure flash storage deletion

Secure deletion is sometimes referred to as *forgotten*, *erased*, *deleted*, *completely removed*, *reliably removed*, *purged*, *self-destructed*, *sanitized*, *revoked*, *assuredly deleted*, and *destroyed* in literature (Reardon et al. 2013a, p. 301). Before existing secure flash storage deletion approaches are discussed, an overview of flash storage is presented.

2.3.1 Flash storage: An overview

2.3.1.1 Flash storage layers and structures

The process of storing new or deleting existing data by an app usually takes place over different layers as outlined in Figure 6. For example, an app usually modifies data by *calling* the Application Programming Interface (API) function provided by the OS. The data modification is then processed by the file system, and in the Android environment, there are actually different ways for the file system to handle the data.

Before Android version 3.0 (Honeycomb), file system accesses the flash storage through memory technology device (MTD) (Woodhouse 2008). There is a built-in software flash translation layer (FTL) responsible for remapping logical block address to physical location (Intel 1998). FTL emulates a normal block device like magnetic hard drive to enable the use of common block file system (e.g. ext4 and FAT32) (Ma et al. 2014; Skillen & Mannan 2013). Unsorted block image (UBI) is an alternative interface to access flash memory, functioning as a layer on top of MTD. UBI implements a FTL separate from the FTL in the OS (Reardon et al. 2013b). In Android version 3.0, MTD interface is replaced by embedded multi-media card (eMMC) and secure digital (SD). Both eMMC and SD have integrated FTL implemented in the hardware controller, and therefore, software FTL is no longer necessary (Skillen & Mannan 2013).

Like a magnetic hard drive, the flash storage is connected via the host interface connection (e.g. SATA, PCI-Express, SCSI, Fibre Channel and USB). All data input/output (I/O) is processed by the processor regardless of the type of FTL. The processor translates binary data to electrical voltage to store data in the flash package.

The structure of flash package is as follows; each *package* is made up of multiple dies, each *die* consists of multiple planes, each *plan* comprises multiple blocks, and finally, each *block* is made up of multiple pages (Ma et al. 2014). *Page* is the smallest unit of I/O operation, and the flash storage is accessed through the page unit (Kim et al. 2007a). Each page has a spare area, known as out-of-band (OOB), which is used to store the error correcting code (ECC) (Huang et al. 2011).

2.3.1.2 Data overwriting in flash storage

Secure deletion usually involves overwriting the original data to make it unrecoverable (Hughes & Coughlin 2006; Gutmann 1996; Garfinkel & Shelat 2003; Wei et al. 2011), and data overwriting can be performed through software running on OS (Cornell University 2012) or firmware-based such as ATA's Secure Erase (Hughes & Coughlin 2002).

Government standards such as the US (Kissel et al. 2012) and Australia (Australian Signals Directorate 2014) recommend using ATA's Secure Erase command or overwrite the media at least once in its entirety. Researchers such as Garfinkel & Shelat (2003), Joukov et al. (2006) agreed that overwriting once is probably adequately secure, although this view is not necessarily shared by others (Wright et al. 2008). Gutmann (1996), for example, suggested a 35-pass overwriting pattern and subsequently he clarified that a few passes (rather than 35 passes) should be adequate in most situations (Gutmann 2003). Garfinkel and Shelat (2003) also explained that Gutmann’s (1996) demonstration that it is possible to recover data using a one-pass wipe is due to older hard drives having gaps between “tracks” and such gaps are not found in modern high-density magnetic hard drives. However, in recent work, Quick and Choo (2013b; 2013a; 2014) demonstrated that data artefacts could be forensically recovered from devices, even after the original data has been securely deleted using tools such as Eraser[14] and CCleaner[15]. For example, they noted that “there was enough information in prefetch files, such as *notepad.exe.pf*, *wordpad.exe.pf*, *explorer.exe.pf* and *dllhost.exe.pf* to indicate the presence and path of the Enron sample data files and the sample Dropbox files” (Quick & Choo 2013b, p. 10).

However, flash storage could not rely on simple data overwriting for secure deletion. FTL not just remaps logical block address, but also distributes write access across flash storage (Spreitzenbarth & Holz 2010). In flash storage, newer data is written to another valid *block* while the original block is simply marked as “invalid” (Shin 2012). This is known as *out-of-place* update; whilst an *in-place* update writes new data on top of previous. Therefore, in an

[14] http://eraser.heidi.ie/
[15] http://www.piriform.com/ccleaner

out-of-place update, the original content is preserved even with an overwrite request. Wei et al. (Wei et al. 2011) demonstrated that existing single-file secure deletion tools are ineffective in securely deleting file content in flash storage. To truly overwrite the data, flash storage must erase the block prior to overwriting with new data. However, an erase operation is significantly slower than write operation (Choi et al. 2014) and the number of erase operation allowed on a single block is limited, ranging from 10,000 to 100,000 before it becomes a *bad block* - a block that cannot store data anymore (Reardon et al. 2013b; Qin et al. 2013).

2.3.2 Classification of secure flash storage deletion

In this survey, the categorisation approach proposed by Diesburg's (2012) and Reardon et al. (2013a) are adapted. This discussion broadly categorised secure deletion approaches into the user-space layer, the file system layer and the physical layer (Figure 6). An alternative approach is to organise them into complete overwrite, random overwrite, delete sensitive, and block deletion (Gupta et al. 2011). However, the latter approach is more appropriate in describing features provided by a secure deletion method. Thus, it is not suitable to classify each secure deletion method distinctly because each method described in this work most likely incorporates more than one feature.

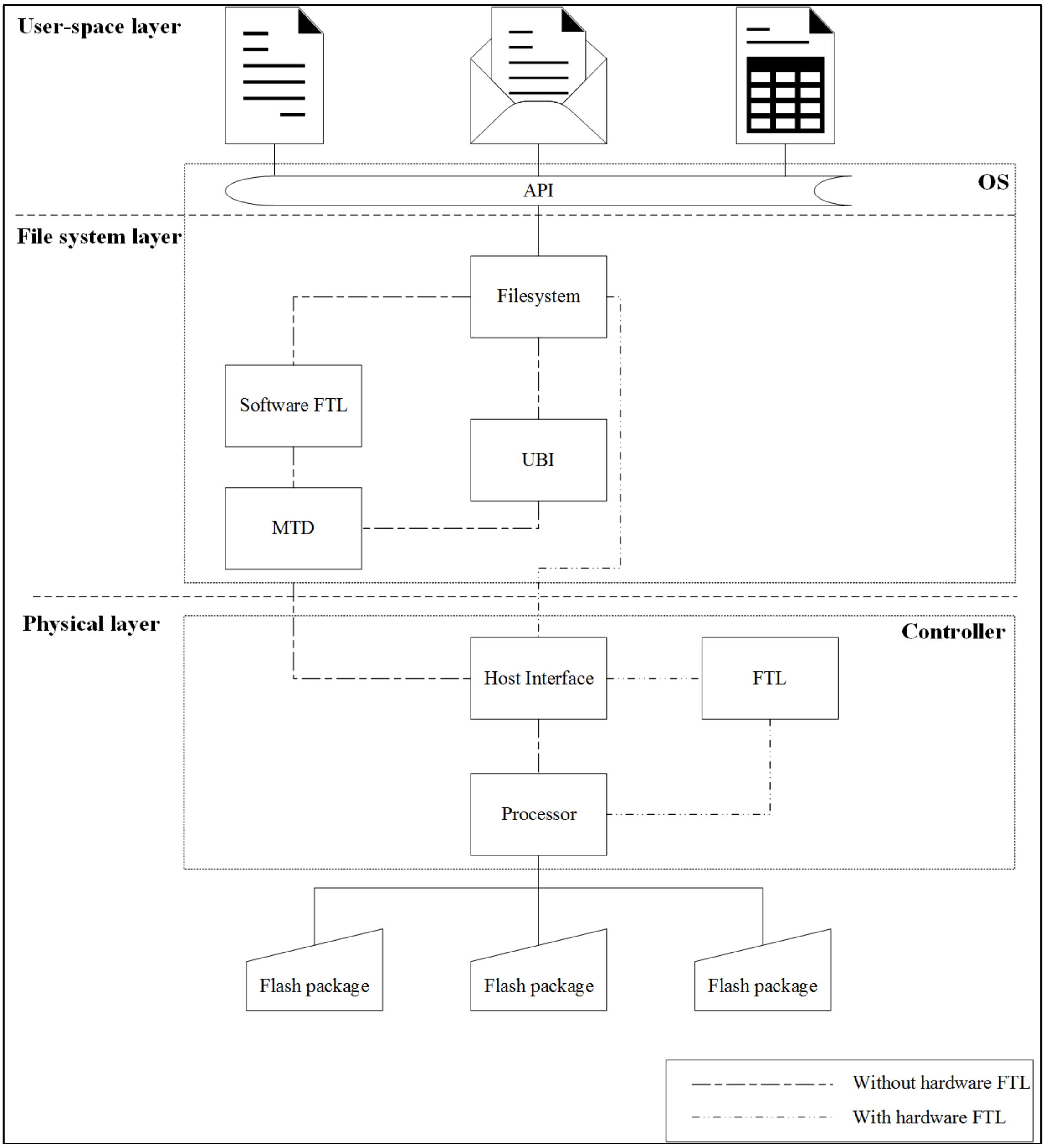


Figure 6: Data access layers in flash storage. (Adapted from Reardon et al. (2013a), *Agrawal et al.* (2008))

2.3.2.1 User-space layer

Spreitzenbarth and Holz (2010) developed a secure deletion tool for Symbian OS, which overwrites personal data (e.g. contacts, calendar entries, and SMS message) using the OS API. Since the tool utilised OS API, it can be easily ported to other platform. Wear levelling techniques to prolong the service life of the storage media (e.g. flash memory) that have a limited number of erase cycles before being rendered unreliable was not considered in this work, and therefore, limiting its widespread adoption.

Reardon et al. (2012) developed a secure deletion tool for Android OS that will monitor the amount of free space and fill it with random data. This ensures unwanted data marked as invalid is filled with random data to achieve random deletion.

Albano et al. (2011) proposed using standard Linux commands (e.g. cp, rm and dd) to delete data of interest in Android devices, without using any cryptographic primitives or kernel modules that will raise suspicion during a forensics analysis. The deletion process is summarised as follows:

1. Copy */data* partition to an external SD card.
2. Zero the partition while deleting the data of interest on external SD.
3. Move the remaining data on external SD back to the */data* partition.
4. Zero the external SD.

The proposed method requires BusyBox (Perens 2014) to be installed (which provides the standard Linux commands). Installing BusyBox requires the user to have root privilege, but such an approach will void the warranty and result in the device being vulnerable to other malicious threat (Pieterse & Olivier 2013).

Kang et al. (2013) proposed another method of data wiping for mobile phone, which overwrites only part of the data that will render it unidentifiable instead of overwriting the entire data. Their approach is designed only for data of the following file formats, namely; JPEG, BMP, FLV, DOC and XLS.

Steele et al. (2009) proposed a system to wipe several USB flash drives simultaneously. The system checks the pre-set status of the drive upon insertion to determine whether to wipe the drive. Data overwriting defaults to zero-overwriting but it is able to accommodate user-defined pattern. The proposed system did not take into consideration wear-levelling or FTL since the motivation behind the publication is simply because “there is literature on the Internet that suggests that such recovery is possible for the dedicated hacker up to at least 10 layers of previously written data in some versions of solid state memory”. However, such literature could not be located.

The secure deletion approach proposed by Jevans et al. (2007) uses either overwriting with zeroes and ones, or cryptographic secure bit patterns. The proposed approach is designed with a mechanism to resume interrupted wiping after the device is subsequently powering on, and to ensure wear levelling. The approach described in this patent was implemented in IronKey's product[16].

2.3.2.2 File system layer

Weng and Wu (2012) proposed using data encryption for secure flash storage deletion. Each data block is encrypted with a key and whenever the data needs to be deleted, the key will be removed. The work did not mention any mechanism to securely erase the key. The proposed method of Lee et al. (2010a) has a similar concept but their method ensures that keys are stored in the same block using "unbalanced binary hash tree" algorithm. Thus, a file can be securely deleted by erasing the file header block which deletes the key. This work has been patented (Park et al. 2012). Lee et al. (2011) extended their previous work (Lee et al. 2010a) to include US government standards on data sanitisation. It will overwrite the data before erase operation whereas previous work was erase operation only. This is without additional operations required in their previous work; and thus, the claim that the newer scheme is more secure and efficient than the previous scheme. The proposal by Guyot et al. (2012) is also based on data encryption. The proposed method not only deletes the keys but includes garbage collection to remove duplicate keys due to wear-levelling effect.

Reardon et al. (2013b) criticised that Lee et al. (2010a) proposal is only conceptual and that if implemented will cause too much wear on flash memory. They then proposed a scheme that is similar to Lee et al. (2010a) where each data block is encrypted with a key and the key is purged through *erase* operation when the data is no longer needed. The proposed scheme encrypts each block of data with distinct 128-bit AES key in counter mode. IV (initialisation vector) is not used due to distinct key. The scheme is implemented in UBIFS (Unsorted Block Image File System), a log-structured file system that builds upon UBI and is tested on Android. The authors conducted various tests including wear analysis, power consumption, and I/O performance. However, Skillen and Mannan (2013) argued that Reardon et al. (2013b) proposed method could only work with memory technology device (MTD) due to dependency on UBI.

Sun et al. (2008) proposed a hybrid secure deletion scheme that utilises two techniques; block cleaning and zero overwriting. Block cleaning is basically an "*erase*" operation (Section Data overwriting in flash storage). The proposed scheme calculates the cost of each technique before

[16] http://www.ironkey.com/en-US/

performing the secure delete and choosing the technique with a lower cost. There are different scenarios which can affect the cost of each technique. In block cleaning, any valid pages (pages that are storing valid data) residing inside a block have to be copied somewhere else before erasing the block. Thus, the cost of block cleaning increases as the number of valid pages in a block increases. In contrast, zero overwriting can selectively overwrite affected pages (pages that store the data user wants to delete) only. Thus, block cleaning tends to be slower whenever large amount of valid pages are involved since the operation has to copy them first. Lee et at. (2011) argued that block erasure does not involve overwriting; thus does not meet US government standard. Subha (2009) further pointed out that the calculation introduces latency.

Choi et al. (2014) proposed a scheme involving password-based per-file encryption and secure data deletion. The proposed encryption scheme encrypts file name and file data separately. When user wants to delete particular file, the process is as follow:

1. Write zeroes to all spaces occupied by file name, file address, and file data.
2. Read the spaces and verify that it is “0x00”.
3. Execute TRIM function that allows an operating system to inform the flash storage that the spaces are no longer used and can be erased.
4. Finally, erase the space(s) so that new data can be stored.

Choi et al. (2014) criticised that Truecrypt, a software-based full disk encryption, for storing secret information (e.g. encryption/decryption key, user’s password, and master key) while their proposed solution does not. The key problem with this criticism is that Truecrypt securely stores the *master* key encrypted using *header* key. The header key is not stored but derived either from user’s password or keyfile or both (Brož & Matyáš 2014). This is similar to the proposed file encryption scheme of Choi et al. (2014), but simpler (Figure 7).

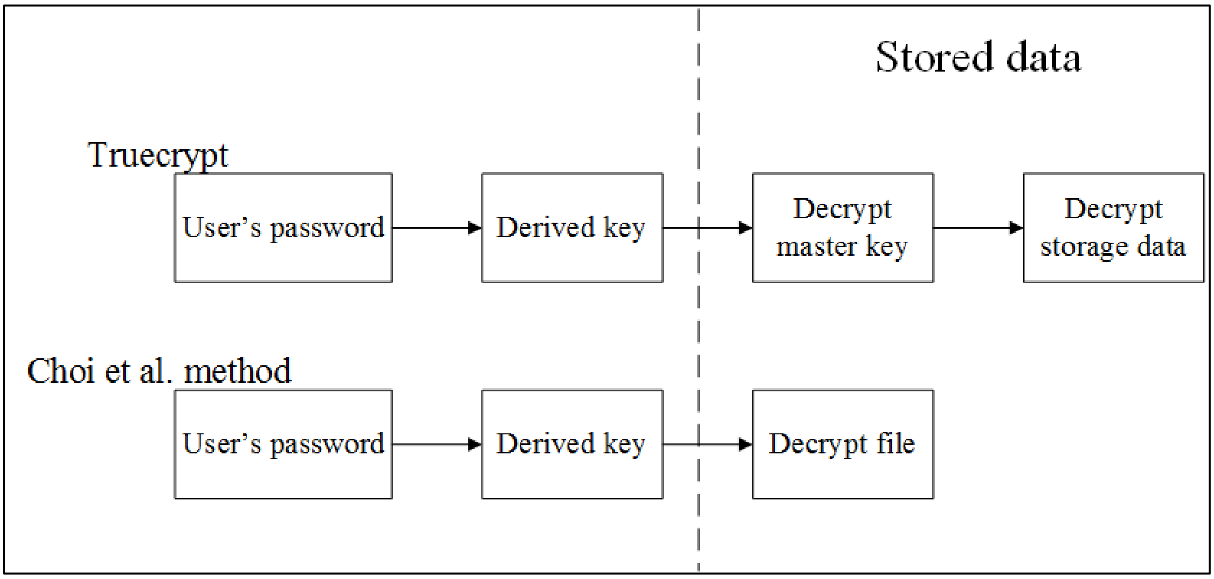


Figure 7: Decryption process in Truecrypt and Choi et al.

Choi et al. chose to use single file encryption because only important data is targeted and that full disk encryption is significantly slower. The advantage is protection against *cold-boot attack* (Halderman et al. 2009; Müller & Spreitzenbarth 2013) because user's password and key created temporarily in RAM can be safely disposed after the encryption or decryption process (Choi et al. 2014). Compared to full disk encryption where data is encrypted and decrypted on-the-fly, the key has to be stored in RAM or cached somewhere else to encrypt and decrypt data. However, important data does not constitute just a single file but may encompass multiple files. In the proposed method, if a user wants to access multiple files, the user has to provide individual passwords for decryption of each file, which is inefficient.

2.3.2.3 Physical layer

Shin (2012) explored the feasibility of implementing secure deletion in different FTL schemes. Shin claimed that current approaches are effective but suffered from low performance due to the design limitation in existing FTL schemes, and suggested the need for new FTL scheme that is both effective and achieves good performance. Wei et al. (2011) developed a new FTL scheme that zero-overwrite unused copies of data. The scheme works by re-programming unused cell to flip remaining ones to zeroes. Wei et al. (2011) cautioned that this approach could trigger a 'Program Disturb' error (i.e. memory cells that are not being programmed receiving elevated voltage stress). Reardon et al (2012) criticised such an approach as reprogramming operates outside existing specification, but Wei et al. (2011) claimed that the 'Program Disturb' issue will not affect all drives.

Qin et al. (2013) also adopted a similar approach as of Wei et al. (2011) in their proposal to ensure traditional data overwriting is still effective. In order to mitigate the negative effect of reprogramming, RAID-5 is suggested. Although RAID-5 is generally found in corporate environment, it is widely supported in consumer product without using dedicated RAID hardware (Intel 2014; AMD 2010). Despite RAID-5 being able to provide fault tolerance, there are several disadvantages. For example, a typical RAID-5 setup requires at least three disks (Vantage Technologies n.d.), and part of the storage capacity is allocated to store parity to achieve fault tolerance. Therefore, RAID-5 is not suitable for the general consumer (due to the expenses involved).

Rather than overwriting the data to be deleted, Subha (2009) proposed to render the data inaccessible. The error correcting code (ECC) is stored in a reserved area known as out-of-band (OOB), and Subha proposed to overwrite certain parts of ECC to introduce read error, and therefore, rendering the data that reside in the block inaccessible. This results in a faster secure deletion time since the size of ECC is typical very small. However, it is yet unknown whether data overwritten using ECC can be subsequently accessible by the OS.

Diesburg et al. (2012) proposed *TrueErase*, an approach to correctly propagate secure deletion information across layers. The approach considered the full data-paths from user-space down to physical layer. Reardon et al. (2013a) liken this approach to a TRIM command but argued that *TrueErase* is more efficient as it can target only sensitive part of file instead of the entire file. Due to consideration of all layers involved, it is the most comprehensive approach but such an approach is complex to implement (Bonetti et al. 2014).

Linnell (2012) proposed a block erasure method whereby a block is erased by reprogramming all pages into zeroes, similar to Wei et al. (2011). Before the block is erased, any pages still holding valid data (i.e. valid page) are copied to another block. However, such an approach was pointed out in earlier work (Sun et al. 2008) to be slower than zero-overwriting when there is a large number of valid pages to be copied.

Rather than using a series of write commands from OS or host interface, Koren et al. (2006) proposed wiping flash storage using a single command from the host device, which will trigger the flash storage to wipe itself. The command can also be sent wirelessly via Bluetooth or infrared. Logical conditions such as higher than usual read operation can be used to trigger the self-wiping to mitigate unauthorised disk duplication. The proposed method includes a mechanism to automatically resume interrupted wiping process (described in Section Ensure wiping operation is completed). After the entire flash storage has been wiped, the flash storage has a built-in status to indicate the process completion.

2.3.3 Android implementation

According to concurrent work by Simon and Anderson (2015a), flash memory can be erased through ioctl() system call provided by Linux kernel. On MTD interface, *ioctl(MEMERASE)* method is used to erase flash blocks so that they are available to store new data (Sang 2012; Jangir 2012). When eMMC was introduced to Android device, Android uses *ioctl(BLKDISCARD)* to send “TRIM” command to the flash controller. It is not considered as secure deletion because it simply marks the block as available for new data. *ioctl(BLKSEDISCARD)* was later implemented in version 4.0 (Ice Cream Sandwich) for secure deletion. This system call will pass “SECURE ERASE” OR “SECURE TRIM” to flash controller. The flash controller would execute “ERASE” or “TRIM” operation followed by “SANITIZE” (Google n.d.a). This is akin to ATA/ATAPI Command Set-2 (ACS-2) “SANITIZE BLOCK ERASE” used in desktop drive (Wei et al. 2011). Table 5 summarise the deletion method implemented throughout history of Android as identified by Simon and Anderson (2015a). Deletion method for *media* partition is only applicable when it is standalone partition. Most devices with Honeycomb or later would have ‘media’ folder under *data* partition (*/data/media*) instead of separate partition.

		Android version				
Code	**Partition**	**Froyo**	**GB**	**ICS**	**JB**	**KK**
Android	*media*	*format()*			*ioctl(BLKDISCARD)*	
	External SD	*None*				
Recovery	*data*	*ioctl(MEMERASE)*	*ioctl(BLKDISCARD)*	*ioctl(BLKSEDISCARD)*		

Table 5: Deletion method of factory reset in AOSP (adapted from Simon and Anderson (2015a))

2.4 Discussion

This section discusses the various limitations in existing approaches (Chapter 2.2 & 2.3) before summarising this chapter.

Technique	Advantages	Disadvantages	Evaluation criteria / claimed features (for approaches without an experiment)	Research set-up
User-space				
Spreitzenbarth & Holz (2010)	• Simple to implement. • No OS modification required. • Cross-platform.	• Does not provide wear-levelling. • Limited data type support.	• Data recoverability	• Experiment using Nokia E90 (Symbian 9.2)/S60 platform
Reardon et al. (2012)	• Data type agnostic. • Provides wear-levelling.	• Excessive writes or wear on storage. • Slow	• Effects of different parameter on deletion latency and lifetime. • Battery consumption.	• HTC Nexus One
Albano et al. (2011)	• Simple operation. • Data type agnostic, • No OS modification required.	• Does not provide wear-levelling. • Works on Android or any Linux-based OS only. • Requires root and BusyBox installed on Android. • Slow.	• Data recoverability	• HTC Nexus One (MIUI ROM based on Android v2.3.4)
Kang et al. (2013)	• Efficiency as only parts of data needs to be overwritten.	• Does not provide wear-levelling. • Limited data type support.	• Data recoverability • Deletion time	• Samsung Galaxy S3
Steele et al. (2009)	• Wipe several USB flash drives simultaneously. • Questionable motivation behind the proposal.	• Does not address wear-levelling.	NA	NA
Jevans et al. (2007)	• Resume interrupted wiping.	• Limited wear levelling.	NA	NA
File system				
Weng & Wu (2012)	• Only small data (key) needs to be deleted.	• No mention of secure deletion for keys.	NA	NA
Lee et al. (2010a) and Park et al. (2012)	• Encryption keys are arranged closely for faster delete operation.	• Excessive wear on flash storage. • Conceptual (yet to be implemented / evaluated).	• Amortized number of block erase	• No experiment conducted
Lee et al. (2011)	• Incorporate US government standards • More efficient than (Lee et al. 2010a) and introduce less wear on flash storage.	• Conceptual (yet to be implemented / evaluated).	• Amortized number of block erase	• No experiment conducted
Guyot et al. (2012)	• Method to remove duplicates of deleted data provided.	• Latency of garbage collection operation.	NA	NA
Reardon et al. (2013b)	• Can be modified into full disk encryption for confidentiality.	• Designed for UBIFS, a file system not found in Android but supported by the Linux kernel. • Depends on now defunct MTD	• Execution time for various file system functionality (e.g. mount/unmount, read/write) • Power consumption	• HTC Nexus One (Linux v2.6.35.7)
Sun et al. (2008)	• Hybrid scheme which chooses faster method by evaluating the nature of data location.	• Latency during cost calculation.	• Time taken to complete various workloads.	• Embedded board 400MHz Intel XScale CPU, 64MB SDRAM, 64MB Samsung NAND flash memory.
Choi et al. (2014)	• Compatibility with TRIM • Verify the data has been overwritten	• Additional operations increase deletion time.	NA	NA
Physical				
Wei et al. (2011)	• Almost native performance	• May result in writing error.	• Write latency	• Custom-built FPGA-based flash testing hardware on 16 chips spanning 6

		• Possible violation of flash storage’s specification.	• Secure deletion latency on different flash storage and applications	manufacturers, 5 technology nodes covering both MLC and SLC chips
Qin et al. (2013)	• Use RAID-5 to mitigate negative effect of reprogram.	• Require multiple drives for RAID; thus not cost-effective / suitable for general consumer usage.	• Read/write response time	• Simulator SSDsim
Subha (2009)	• Least data to be affected thus very fast deletion time.	• Access to ECC is questionable.	• Deletion time	• C program running on Linux file system to simulate ECC, with a Rich Text File (RTF) as input data.
Diesburg et al. (2012)	• Most comprehensive approach	• Complex to implement	• Data recoverability • Disk performance	SanDisk’s DiskOnChip with Inverse NAND • File Translation Layer (INFTL) kernel module on Linux 2.6.25.6
Linnell (2012)	• FTL compatibility.	• Slower than simple zeroes overwriting in some cases.	NA	NA
Koren et al. (2006)	• Erase operation is independent from host device (OS and motherboard). • Resume interrupted wiping.	• Does not provide wear-levelling	NA	NA

Table 6: Comparative summary of existing secure deletion techniques.

2.4.1 Limitations of existing literature

2.4.1.1 Lack of data recovery evaluation

It is clear that most existing approaches focus on data recoverability testing, but there is a lack of data recovery evaluation particularly at the file system and physical layers.

Almost all existing approaches had limited evaluations to determine their suitability (i.e. one experiment per device in most proposals) which brings into question whether the approach can be widely deployed over the wide range of mobile devices. For example, Wei et al. (2011) argued that different flash storage media could exhibit different behaviours and, therefore, conducted tests on a wide range of commercially available consumer hardware. In addition, the findings are dated, many of the devices tested such as HTC Nexus One are either discontinued (Gross 2010) or no longer available. Therefore, findings from Albano et al. (2011), Reardon et al. (2012), Reardon et al. (2013b) and others may no longer be applicable to newer devices and OS.

2.4.1.2 Limitation of using simulation in evaluations

A number of proposals was evaluated in the simulated environment (Qin et al. 2013; Subha 2009). While there are advantages in using a simulated environment such as a mobile emulator (e.g. ease of use, without the need for custom-build hardware platform, and ability to evaluate approach for an expensive or yet to be available commercial flash technology as outlined by Grupp et al. (2010), simulation results are likely to be less reliable than actual hardware-based evaluations, and hence, limit our understanding of the resulting real-world implications (Saxena et al. 2013).

2.4.1.3 Performance

Modification on hardware-level does not necessarily mean modifying the flash memory cell or the processor of the SSD controller; in this paper, this refers to the modifying the software or the firmware running at the physical level. Thus, the *software-based* approach referred in **Error! Reference source not found.** is implemented above the physical level, namely file system and user-space layers.

Software-based implementation generally has a lower throughput due to the number of layers involved; whilst a hardware-based implementation operates on the disk's firmware which allows it to run at the disk's full bandwidth (Reddy & Rao 2014). For instance, software-based full disk encryption (FDE) generally impacts on the disk's performance even in flash storage (Larabel 2014).

2.4.1.4 Possible attacks

Hardware-based secure deletion approach can perform erasure on all memory blocks (Wei et al. 2011), but software-based solution may not be able to access some of the blocks. ATA's *Secure Erase* function is the most commonly available hardware-based secure deletion method, which can be found in most drives manufactured on or after year 2001. However, researchers such as Wei et al. (2011) and Swanson & Wei (2010) found that some drives either fail to complete the deletion process required in the Secure Erase function or do not erase the data at all after executing that function.

Data can also be protected using encryption, such as hardware-based drive encryption (also known as self-encrypting drive – SED). Müller et al. (2012) proposed a *hot plug attack*, where an adversary is able to gain unauthorised access to the data residing in the SED. In short, this is due to the fact that when using the SED, a user unlocks the drive when powering on the machine. After the disk is unlocked, and while the disk is still running ("*hot*"), an adversary simply re-plugs the SATA cable from the original machine to the adversary's machine to access the SED without knowing the password. An adversary can also access the drive directly by attaching a USB drive into the original machine if it is not screen locked.

To ensure high security compliance, there are several industry standards for SED, such as Opal Security Subsystem Class (Opal SSC) by Trusted Computing Group (TCG) (2012) and "Encrypted Hard Drive" (eDrive) by Microsoft (2012). The latter, for example, is partly based on Opal SSC and IEEE 1667 (Rich 2007). Major SED manufacturers offer OPAL-compliant products (Müller et al. 2012; TCG n.d.; Kingston 2013; Intel 2013). Müller et al. (2012) did not evaluate Opal-compliant SEDs, but claimed that *hot plug attack* affects such drives too. Since then, there had been no major revision to the Opal SSC standard, and it is unknown whether such attack claimed by Müller et al. (2012) is valid.

On the other hand, software-based disk encryption may be vulnerable to *cold boot attack* (Halderman et al. 2009; Müller & Spreitzenbarth 2013) because the encryption key is cached in RAM. In such an attack, an adversary removes the RAM, re-plugs into another machine, and extracts the key from the RAM. Such an attack is, however, difficult to carry out and can be mitigated by keeping the key outside of RAM (Wetzels 2014). In addition, software-based disk encryption does not encrypt the boot sector, therefore, an adversary is able to launch a *evil maid attack* (Rutkowska 2009) by installing a *bootkit* (**boot** sector root**kit**) into the victim's machine to capture the password entry. Another form of evil maid attack can be launched against hardware-based disk encryption. In this case, an adversary removes the victim's disk and replaces it with another disk loaded with the adversary's modified OS designed to capture password entry (Müller et al. 2012; Rutkowska 2011). In this case, it can be thwarted using ATA's password. Evil maid attack is possible in both software-based and hardware-based due to a lack of trusted boot environment (Tereshkin 2010) to authenticate the boot sector or the disk *to* the user (Rutkowska 2011).

2.4.2 Summary

Table 7 summarises the key differences between hardware- and software-based implementations of secure flash storage deletion.

	Hardware-based approach	**Software-based approach**
Performance	Generally higher	Slower
Security issue	'Hot plug' attack	'Cold boot' and related attacks
Verifiability	Difficult	Possible
Ease of implementation	Hard	Easy
Cost	High	Low

Table 7: Key differences between hardware- and software-based implementations (Adapted from Choi et al. (2014))

The advantage in using a software-based approach as it allows easy verifiability. For example, if the source code is available, then a public security audit can be conducted using forensic techniques as demonstrated on TrueCrypt (Brož & Matyáš 2014). Hardware-based verification may require not only building a customised platform to access the memory directly but also dismantling the device (Swanson & Wei 2010).

Modification on the hardware level can be very challenging, as one would generally require having access to the source code of the firmware or the specification of the disk controller. It may be possible to replace the firmware, but there is the risk of *bricking* the device. Software-based solutions have a lower risk of damaging the hardware. Software-based modification, especially file system layer, usually builds on an existing open source file system. The improvement can be implemented simply by installing a new patch. Even in the case of a new software component, existing data can be migrated with relative ease.

In addition, software-based modification is generally hardware independent. Implementation on hardware-level, however, may require compatibility at the higher layers. User needs to install new software, for example to support Opal SSC (Sophos 2014), and acquires new hardware. For example, in Intel SED, the drive is encrypted by default using the unique key generated during manufacturing (Intel 2012). It is activated simply by using the drive. This mechanism could only protect in situation where the NAND chip has been removed, assuming that the key is not stored in the NAND chip or the location is only known to the controller. Thus, SED behaves more like “self-decrypting disk” (Müller et al. 2012) in the default configuration. To protect the drive, the user would need to set the ATA password which controls access to the drive, and consequently the data. This mechanism requires ATA specification compatibility. Although the majority of consumer hard drives use SATA (Serial ATA) interface, there are drives that utilise SCSI/SAS, Fibre Channel, or PCIe host interface.

Some new hardware-based features require installation of new hardware, and therefore, and a more expensive option. In software-based approach, a user can take advantage of new features via software updates.

2.5 Summary

This chapter examined (the limited) literature on remote wiping, particularly secure deletion on flash storage. Despite the prevalence of remote wiping, most existing literature provided a high-level approach to remote wiping and secure data deletion. There are relatively few technical papers evaluating the implementation of such approaches or techniques on a wide range of popular mobile devices. One reason this may not have been thoroughly explored is due to the cost and efforts associated with such evaluations.

In addition to conducting a comparative summary of existing approaches (see Table 6), following table (Table 8) identified existing limitations and the research trends over the years.

Publications	**Years**	**Research focus**
Angelo et al. (2003), Kenney (2005), Hasebe (1999)	1999-2005	Remote wiping
Onyon et al. (2007), Gajdos & Kretz (2006)	2006-2010	
Brown et al. (2011), Park et al. (Park et al. 2011), Joe & Lee (2011)	2011	
Kuppusamy et al. (2012)	2012	
Walker & Fyke (2013)	2013	
Yu et al. (2014), Adusumalli (2014)	2014	
Sun et al. (2008), Jevans et al. (2007), Koren et al. (2006)	2006-2008	Secure flash storage deletion
Spreitzenbarth & Holz (2010), Steele et al. (2009), Lee et al. (2010a), Subha (2009)	2009-2010	
Wei et al. (2011), Albano et al. (2011), Lee et al. (2011)	2011	
Linnell (2012), Reardon et al. (2012), Weng & Wu (2012), Park et al. (2012), Guyot et al. (2012)	2012	
Reardon et al. (2013b), Qin et al. (2013), Kang et al. (2013)	2013	
Choi et al. (2014)	2014	

Table 8: Remote wiping and secure flash storage deletion publications by research focus

As shown in Table 8, majority of remote wiping patents were filed prior to 2010, although academic interest on the topic appears to have increase since then. Similar trend was observed on secure flash storage deletion, where there are at least three publications annually since 2010.

This review highlights a number of literature gaps which are as follow:

- *The need to provide message confidentiality using encryption and ensure that wiping process cannot be interrupted.* From the survey described, existing proposals generally do not consider securing the wipe command (Section Secure wipe command) nor provide any mechanism to automatically resume interrupted wiping process (Section 2.1.5).

- *The need for comprehensive evaluations on the security and effectiveness based on real-world implementations of remote wiping*. It is essential to ensure that remote wiping command cannot be hijacked by attackers (e.g. to prevent the wiping of lost or stolen devices) or initiated by attackers (e.g. to remotely wipe contents from victim's device) and wiped data cannot be recovered using contemporary forensic techniques.
- *Real world implementation that mitigate some of the shortcomings.*
- *The need for evaluation of physical layer implementation on actual hardware.* As discussed in Section Limitation of using simulation in evaluations, evaluations of existing hardware-level research on flash storage are generally conducted on simulator. The findings may not take into consideration internal workings of the flash storage (as manufacturers may be hesitant to provide such information to protect their intellectual property) (Saxena et al. 2013; Swanson & Wei 2010). To overcome this limitation, Diesburg et al. (2012) suggested using *OpenSSD* (Lee & Kim 2011), a research platform designed for flash storage research. There are also a few alternative platforms, such as *FRP* (Davis & Zhang 2009), *BlueSSD* (Lee et al. 2010b), and *Ming II* (Bunker et al. 2012). These open platforms allow researchers to have unfettered access to the hardware especially the FTL, which is not possible on commercial flash storage.
- *The need for stronger collaboration between manufacturer and academic researcher*. This review highlighted the importance of FTL as a vital factor in secure flash storage deletion. For instance, Diesburg et al. (2012) acknowledged their work is only possible due to access to software FTL and pointed out the trend of hardware FTL in recent times. For example, newer Android versions have started utilising hardware FTL (Section Flash storage layers and structures). Since hardware FTL is not accessible in commercial hardware or the open research platforms mentioned earlier, any research or evaluation on FTL could not be conducted without the involvement and collaboration of a manufacturer. Therefore, it is argued that a stronger collaboration between manufacturer and academic researcher will result in a more secure product. Researchers can work with Open NAND Flash Interface (ONFi)[17], a consortium of flash memory manufacturer, to incorporate secure deletion method into ONFi specification to facilitate wider adoption.
- *Does stronger security hinder law enforcement?* Stronger security on remote wiping mechanism can help protect the privacy of consumer. However, such benefit could also

[17] http://www.onfi.org/

be abused by criminal to remove incriminating evidence (Mislan et al. 2010; Wakefield 2014). In addition, law enforcement have explained that they could not extract useful evidence from mobile devices due to storage encryption (Timberg & Miller 2014; Hattem 2014; Barrett et al. 2014), particularly in the post NSA revelations as mobile device vendors and other technology companies enforce encryption by default in their products (Frizell 2014; Gustin 2013). Whether such trend really does hinder law enforcement is the subject of controversy and, perhaps, worthy of discussion. For example, how do we balance the need for user privacy with the legitimate needs of government and law enforcement agencies to access data to facilitate their investigations? This issue is briefly explored in Chapter 3 where user data could not be recovered due to secure deletion in Android device.

3 Effectiveness of remote wiping

This chapter compares the effectiveness of seven third-party remote wiping app against Android built-in's, Android Device Manager. Three mobile devices with vary Android version are used: Motorola Moto G, Samsung Nexus S, and LG Nexus 4.

Nexus S was purchased 3 years ago and had been author's primary phone for 2 years. The Moto G was a newly acquired (less than 4 months) mobile device. The Nexus 4 has 2 years of usage. Both phones have prior limited use with student experiments.

All three devices were installed with different Android version. Moto G uses version 4.4.2 KitKat (KK). Nexus S that was initially shipped with 2.3 Gingerbread (GB) has been upgraded to 4.1.2 Jellybean (JB). Nexus 4 was shipped with 4.2 JB has been upgraded to 5.1 Lollipop (LP).

Device	Motorola Moto G	Samsung Nexus S	LG Nexus 4
Model	XT1033	GT-I9020T	LGE960
Android OS	4.4.2	4.1.2	5.1
Android Build	KXB20.25-1.31	JZO54K	LMY47O
Linux kernel	3.4.0	3.0.31	3.4.0
Storage	8 GB	16 GB	8GB
RAM	1 GB	512 MB	2 GB

Table 9: Device specification.

Moto G uses F2FS file system for userdata partition. F2FS was released back in February 2013 (Larabel 2013; Kim 2012; Lee et al. 2015). F2FS is specifically designed for NAND flash memory-based storage devices which is commonly used in mobile device. F2FS is relatively new compared to EXT4 (released in December 2008, (Larabel 2008)). EXT4 has been the default file system for Android device since Android 2.3 (Google 2010).

Partition	Moto G	Nexus S	Nexus 4
	File system		
Recovery	Unknown		
Boot			
Cache	EXT4	YAFFS2	EXT4
System		EXT4	
Userdata	F2FS		
Internal SDcard/Media		FAT32	

Table 10: Partition layout.

3.1 Preparation

Each mobile device was restored to factory image. Set of apps and data are loaded onto the mobile devices as follow:

1. Sign into Google account and connect to “unisa” wireless network.
2. Save 30 contacts and call 10 of them.
3. Sync email. (Emails were generated on the Google account beforehand)
4. Install Google Drive, Dropbox, Box, & OneDrive.
5. Download documents:
    a. 30 DOCX through Google Drive.
    b. 30 PPTX through Dropbox.
    c. 30 XLSX through Box.
    d. 30 PDF through OneDrive.
6. Transfer following files to the mobile device:
    a. 120 JPEG pictures.
    b. 35 MP4 videos.
    c. 30 MP3 audios.
7. Browse to 50 websites and bookmark them.
8. Sign into Reddit (www.reddit.com) and save login.
9. Install and sign into Facebook app.
10. Install and sign into Skype app (except for Nexus S).

“Baseline” dd physical image is then taken. Remote wiping is initiated and another physical image is acquired. Experiment is repeated by using different app to initiate remote wipe and physical image is acquired after each wipe. Before each experiment, the mobile device is restored back to “baseline”. More detailed steps and command used are described at Appendix: Table 38 and Appendix: Figure 13.

The experiment tests the default built-in remote wipe app, Android Device Manager, but also seven third-party apps (Table 11). These apps are chosen based on preference towards cloud anti-virus app and free trial offer.

#	App	Note	Moto G	Nexus S	Nexus 4
			Version		
A	Baseline		N/A		
B	Android Device Manger (ADM)		N/A		
C	Avast		3.0.7650	3.0.7756	3.0.7756

D		(S)			
E	AVG		4.3	4.3.1.1	4.3.1.1
F	Avira		3.9	4.0	4.0
G	Cheetah Mobile (CM)	(W)	2.4.4	2.4.9	2.5.0
H					
I	Panda		2.1.4	2.2.1	2.2.1
J	Sophos	(S)	4.0.1435	5.0.1515	5.0.1515
K	Trustlook		2.5.5	2.5.10	2.6.0
L		(W)			

Table 11: List of apps tested.
Legend: (S) Secure deletion enabled
(W) Without factory reset

3.2 Mobile forensic tool used

Physical image collected on each experiment would be analysed by following mobile forensic tools. Four forensic tools were used to analyse *userdata* partition of Moto G and Nexus 4 and *media* partition of Nexus S where most of user data would be stored.

UFED Physical Analyzer (4.1 Trial). A commercial forensics tool from Cellebrite was used for analysing the physical images collected. Cellebrite offered 30-day free trial for the application. The application can analyse Cellebrite’s proprietary disk image (.UFD), dd raw disk image and logical file system dump. Cellebrite also offers separate hardware and software for data acquisition and save the acquired image in proprietary format (.UFD). The hardware was previously used in Schwamm’s research (2014).

Internet Evidence Finder (IEF 6.5 Trial). A commercial forensics tool from Magnetic Forensics was used in the experiment. 30-day free trial is also offered. The application supports reading from Encase (.E01), FTK (.AD1) proprietary image, virtual machine images (.VDI, .VMDK), DMG images, and dd raw image. As the product name implies, the application focus on recovering Internet-related artefacts (e.g. browsing history and chat logs) yet still support recovering common file format. Magnetic Forensics also offers data acquisition software, Magnet ACQUIRE. Similar to data acquisition technique chosen for this experiment, Magnet ACQUIRE also uses dd.

PhotoRec (6.14) is an open source file recovery program written by Christophe Grenier. PhotoRec utilises data carving technique for searching for known file header without parsing the file system[18]. PhotoRec can recognizes and recovers common file formats. It is bundled

[18] PhotoRec can be configured to operate in a mode optimised for *ext* file system.

with another utility, TestDisk primarily used for repairing partition layout. Although PhotoRec supports more than 400 file formats, it configured to recover most common file formats relevant to Android in the experiment conducted. The formats chosen are shown in the results (Table 19-Table 21).

Scalpel (2.0) is another open source file carving utility. It was originally developed by Golden G. Richard III as improvement to previous file carving utility, Foremost. It was presented at 2005 Digital Forensic Research Workshop (Richard III & Roussev 2005) and has been incorporated into the Sleuthkit. Author finds Scalpel very easy to use and configure, but support less file formats out-of-the-box compared to PhotoRec. The file signature used to recover files is also very basic and prone to false positive. It is configured only recover JPEG thumbnails. Following rule is used:

```
jpg  y    5000:50000     \xff\xd8\xff\xe0    \xff\xd9
jpg  y    5000:50000     \xff\xd8\xff\xdb    \xff\xd9
```

Scalpel is extensively used and described in Chapter 4.

3.3 Other mobile forensic tools considered

Following forensic tools were considered but not used because they cannot recognise or analyse the physical images produced in the experiment.

1. **MSAB XRY 6.13** – Cannot recognise the dd physical images.
2. **Oxygen Forensics Suite 6.4.067 (Analyst edition)** - Cannot recognise the dd physical images.
3. **Guidance Software EnCase Forensic 6** - Cannot recognise the dd physical images.
4. **Piriform Recuva 1.51 (Free edition)** – Physical image must be mounted as a drive. Microsoft Windows does not support mounting F2FS file system.

		Android version				
Code	Partition	Froyo	GB	4.0.x (ICS)	JB	KK
Android	media	*format()*			*ioctl(BLKDISCARD)*	
	External SD	*None*				
Recovery	data	*ioctl(MEMERASE)*	*ioctl(BLKDISCARD)*	*ioctl(BLKSEDISCARD)*		

Table 12: Deletion method of factory reset in AOSP (adapted from Simon and Anderson (2015))

3.4 Discussion

Recovered data is listed in Table 13-Table 24. Table 13-Table 15 lists data recovered by Cellebrite. Table 16-Table 18 list data recovered by IEF. Table 19-Table 21 list data recovered by PhotoRec. Table 22-Table 24 lists thumbnails recovered by Scalpel. Number shown in parentheses indicates deleted file. The numbers outside of parentheses always include deleted file. Even when using the same forensic tool, not all types of data can be recovered, thus some data type may not appear in all tables.

As mentioned in Chapter 2.2.8.1, in order to trigger factory reset, app must have “device administrator” permission granted, including the built-in ADM. Unlike traditional Android permissions, this permission is not automatically granted during installation, except for ADM which has been granted by default as a built-in app. User must first enable “remote wipe” through the remote wipe app’s user interface, and the app will prompt user to grant “device administrator” permission.

However, not all apps display the prompts, specifically CM (all mobile devices) and Trustlook (Nexus S and Nexus 4). Enabling “remote wipe” in CM was a confusing process. It requires setting an optional in-app password (used when changing settings in CM) otherwise user could not initiate wipe command (the ‘Wipe’ option simply not showing). CM not only did not prompt for permission but did not prompt for setting the in-app password as well. Newer version of Trustlook installed in the Nexus S and Nexus 4 also had similar shortcoming. Trustlook did not prompt user to grant necessary permission. It could be due to feature restriction in trial version, despite the app indicated that the devices are qualified for free trial on the “remote wiping” feature. AVG also failed to prompt user but did have “Device Administrator” option on the remote wipe setting, the option is enabled.

Without “device administrator” permission, app could not perform secure deletion. App could only rely on the traditional Android permissions that only it to unlink files. The experiment encountered this issue where app without “device administrator” permission (G & L) failed to remove many files. The issue of failing to inform user of the permission is also raised in concurrent work by Simon and Anderson (2015b). Simon and Anderson referred to this issue as invalid “in-app flow”. According to the work, Avast, AVG, and Avira were identified as having invalid in-app flow. In contrast to this work where Avast and Avira did prompt. This difference could be due to different app version installed or evaluation criteria.

In retrospect, the results suggest that other remote wiping apps in Moto G and Nexus 4 were using secure deletion method provided by the OS (Table 12) (Chapter 2.3.3 has more details on secure deletion in Android). This is indicated by almost nil recovery rate. Even if there were some files recovered, those were system-generated files or false positives in this case. Although the deletion method of Lollipop used by Nexus 4 is not known based on discussion in previous chapter, it should be at least similar or more secure than KitKat. On the other hand, since Jellybean version in Nexus S does not support secure deletion for media partition, some user data can be recovered. This is evident in IEF result (Table) where documents and pictures can be recovered.

In addition, for all the results from Nexus S, picture is the most popular file recovered. Cellebrite and IEF supports picture “carving” for more comprehensive deleted data recovery. There are also large number of pictures recovered by PhotoRec. Picture recovery would be further discussed in the next chapter.

3.5 Related work

On previous academic research, Cardwell (2011) tested factory reset function on HTC Nexus One and Samsung Nexus S and found it is effective. However, author finds Cardwell’s test to be invalid because logical acquisition (“adb pull”) was used instead of physical acquisition. Schwamm and Rowe (2014) tested on 21 devices, made up of iPhone, Android and BlackBerry. Schwamm could recover user data from Android and iPhone devices after factory reset. In a concurrent work, Simon and Anderson (2015) tested on 21 Android devices and also found factory reset fails to remove user data from certain devices. In addition, focusing on implementation secure deletion in factory reset function.

Effectiveness of remote wiping

Cellebrite	Moto G										
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**
Call log	12 (2)	0	0	0	0	0	2 (2)	0	0	0	0
Chats [category]											
Google Talk	1 (1)	0	0	0	0	0	1 (1)	0	0	0	0
Hangouts	2 (1)	0	0	0	0	0	2 (1)	0	0	0	0
Kik	2 (2)	0	0	0	0	0	3 (3)	0	0	0	0
Contacts	31	0	0	0	0	0	38 (37)	0	0	0	0
Cookies	516	0	0	0	0	0	516	0	0	0	0
Emails	45 (17)	0	0	0	0	0	46 (18)	0	0	0	0
Installed applications	41 (4)	0	0	0	0	0	43 (5)	0	0	0	0
Passwords	6	0	0	0	0	0	6	0	0	0	0
Powering Events	1	0	0	0	0	0	2	0	0	0	0
Searched items	1	0	0	0	0	0	2	0	0	0	0
SMS	1 (1)	0	0	0	0	0	2 (2)	0	0	0	0
Timeline	718 (1)	0	0	0	0	0	709 (4)	0	0	0	0
User accounts	9	0	0	0	0	0	9	0	0	0	0
Web bookmarks	50	0	0	0	0	0	50	0	0	0	0
Web history	57 (2)	0	0	0	0	0	61 (11)	0	0	0	0
Wireless networks	1	0	0	0	0	0	1	0	0	0	0
Data Files [category]											
Applications	434	3	3	0	3	3	447	3	3	3	3
Audio	31	0	0	0	0	0	0	0	0	0	0
Databases	226	1	1	0	1	1	252	1	1	1	1
Documents	180	0	0	0	0	0	30	0	0	0	0
Images	1910	0	0	0	0	0	886	0	0	0	0
Text	362	2	2	0	2	2	380	2	2	2	2
Videos	35	0	0	0	0	0		0	0	0	0
Carved Images	1375	2	2	0	2	2	200	2	2	2	2
Activity analytics	58	0	0	0	0	0	66	0	0	0	0
Analytics emails [category]											
User email	3	0	0	0	0	0	3	0	0	0	0

Uncategorized	19	0	0	0	0	0	20	0	0	0	0
Analytics phones	32	0	0	0	0	0	32	0	0	0	0
Skype	1	0	0	0	0	0	1	0	0	0	0

Table 13: Data recovered by Cellebrite in Moto G.

Cellebrite	**Nexus S**										
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**L**
SMS	0	2 (2)	2 (2)	1 (1)	2 (2)	2 (2)	2 (2)	2 (2)	2 (2)	0	2 (2)
Timeline	0	2 (2)	2 (2)	1 (1)	2 (2)	2 (2)	2 (2)	2 (2)	2 (2)	0	2 (2)
Data Files [category]											
Audio	32	0	0	0	0	0	0	0	0	1	0
Databases	0	0	0	0	0	0	0	0	0	0	1
Documents	213 (2)	0	0	0	0	0	0	0	0	213 (3)	0
Images	756 (5)	0	0	0	0	0	0	0	0	616 (3)	0
Text	1	0	0	0	0	0	0	0	0	2	0
Videos	35	0	0	0	0	0	0	0	0	0	0
Carved Images	6238	2759	2759	192	2759	2759	2759	2759	2759	0	2759

Table 14: Data recovered by Cellebrite in Nexus S.

Cellebrite	**Nexus 4**										
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**L**
Call log	10	0	0	0	0	0	0	0	0	0	0
Chats [category]											
Hangouts	2 (1)	0	0	0	0	0	2 (1)	0	0	0	2 (1)
Kik	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)
Contacts	31	0	0	0	0	0	37 (36)	0	0	0	37 (36)
Cookies	20	0	0	0	0	0	20	0	0	0	20
Emails	2 (2)	0	0	0	0	0	3 (3)	0	0	0	2 (2)
Installed applications	49 (23)	0	0	0	0	0	59 (22)	0	0	0	51 (24)
Passwords	6	0	0	0	0	0	6	0	0	0	6

Powering Events	3	0	0	0	0	0	4	0	0	0	2 (1)
Searched items	0	0	0	0	0	0	1	0	0	0	1
SMS	3 (3)	0	0	0	0	0	4 (4)	0	0	0	2 (2)
Timeline	172 (7)	0	0	0	0	0	176 (8)	0	0	0	162 (7)
User accounts	9	0	0	0	0	0	9	0	0	0	9
Web bookmarks	50	0	0	0	0	0	50	0	0	0	50
Web history	40	0	0	0	0	0	50	0	0	0	40
Wireless networks	1	0	0	0	0	0	1	0	0	0	1
Data Files [category]											
Applications	381 (65)	0	0	0	0	0	393 (67)	0	0	0	393 (70)
Audio	32	0	0	0	0	0	0	0	0	0	
Databases	178	0	0	0	0	0	203	0	0	0	179
Documents	181 (1)	0	0	0	0	0	30	0	0	0	30
Images	1363 (21)	0	0	0	0	0	351 (16)	0	0	0	364 (16)
Text	289 (3)	0	0	0	0	0	309 (3)	0	0	0	297 (11)
Videos	35	0	0	0	0	0	0	0	0	0	0
Carved Images	966	0	0	0	0	0		0	0	0	(error)
Activity analytics	36	0	0	0	0	0	43	0	0	0	42
Analytics emails [category]											
User email	4	0	0	0	0	0	4	0	0	0	4
Uncategorized	0	0	0	0	0	0	3	0	0	0	0
Analytics phones	30	0	0	0	0	0	30	0	0	0	30
Skype	1	0	0	0	0	0	1	0	0	0	1

Table 15: Data recovered by Cellebrite in Nexus 4.

IEF	**Moto G**										
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**S**	**K**
IEF Refined results [category]											
Google Analytics First Visit Cookies	23	0	0	0	0	0	23	0	0	0	0
Google Analytics Referral Cookies	23	0	0	0	0	0	23	0	0	0	0
Google Analytics Sessions Cookies	23	0	0	0	0	0	23	0	0	0	0

Google Analytics URLs	33	0	0	0	0	0	33	0	0	0	0
Google Searches	1	0	0	0	0	0	2	0	0	0	0
Social Media URLs	107	0	0	0	0	0	97	0	0	0	0
Chat [category]											
Skype accounts	1	0	0	0	0	0	1	0	0	0	0
Skype Contacts	2	0	0	0	0	0	2	0	0	0	0
Google Drive	2	0	0	0	0	0	2	0	0	0	0
Documents [category]											
Excel	36	0	0	0	0	0	13	0	0	0	0
PDF	38	0	0	0	0	0	10	0	0	0	0
PowerPoint	60	0	0	0	0	0	7	0	0	0	0
Text	77	0	0	0	0	0	84	0	0	0	0
Word	60	0	0	0	0	0	59	0	0	0	0
Media [category]											
Carved video	52	0	0	0	0	0	27	0	0	0	0
Pictures	17946	2	2	0	2	2	16875	2	2	2	2
Videos	35	0	0	0	0	0	0	0	0	0	0
Web related [category]											
Browser Activity	368	0	0	0	0	0	388	0	0	0	0
Chrome cookies	516	0	0	0	0	0	516	0	0	0	0
Chrome favicons	63	0	0	0	0	0	0	0	0	0	0
Chrome logins	2	0	0	0	0	0	2	0	0	0	0
Chrome top sites	1	0	0	0	0	0	0	0	0	0	0
Chrome web history	55	0	0	0	0	0	50	0	0	0	0
Chrome web visits	108	0	0	0	0	0	0	0	0	0	0
Chrome/360 Safe Browser/Opera Carved web history	171	0	0	0	0	0	174	0	0	0	0
Firefox web history	10	0	0	0	0	0	11	0	0	0	0
Google Analytics First Visit Cookies	37	0	0	0	0	0	38	0	0	0	0
Google Analytics Referral Cookies	25	0	0	0	0	0	25	0	0	0	0
Google Analytics Sessions Cookies	41	0	0	0	0	0	42	0	0	0	0
Google Analytics URLs	33	0	0	0	0	0	33	0	0	0	0
Google maps	18	0	0	0	0	0	19	0	0	0	0

Google maps tiles	22	0	0	0	0	0	22	0	0	0	0

Table 16: Data recovered by IEF in Moto G.

IEF	**Nexus S**										
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**L**
Documents [category]											
Excel	36	7	7	0	7	7	7	7	7	36	7
PDF	30	30	30	2	30	30	30	30	30	28	30
PowerPoint	60	30	30	0	30	30	30	30	30	60	30
Word	30	30	30		30	30	30	30	30	30	30
Media [category]											
Carved video	41	38	38	23	38	38	38	38	38	14	38
Pictures	8202	7682	7682	675	7682	7682	7682	7682	7682	8001	7682
Videos	35	0	0	0	0	0	0	0	0	0	0
Web related [category]											
Safari history	0	0	0	0	1	1	0	0	1	1	1

Table 17: Data recovered by IEF in Nexus S.

IEF	**Nexus 4**										
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**L**
IEF Refined results [category]											
Google Searches	0	0	0	0	0	0	1	0	0	0	1
Social Media URLs	2	0	0	0	0	0	1	0	0	0	2
Chat [category]											
Skype accounts	1	0	0	0	0	0	1	0	0	0	1
Skype Contacts	2	0	0	0	0	0	2	0	0	0	2
Skype IP Addresses	2	0	0	0	0	0	2	0	0	0	2
Google Drive	4	0	0	0	0	0	4	0	0	0	4
Documents [category]											
Excel	36	0	0	0	0	0	25	0	0	0	25
PDF	36	0	0	0	0	0	31	0	0	0	30

PowerPoint	60	0	0	0	0	0	29	0	0	0	29
Text	22	0	0	0	0	0	27	0	0	0	12
Word	60	0	0	0	0	0	59	0	0	0	59
Media [category]											
Carved video	43	0	0	0	0	0	44	0	0	0	42
Pictures	14321	0	0	0	0	0	14143	0	0	0	12149
Videos	35	0	0	0	0	0		0	0	0	
Web related [category]											
Browser Activity	20	0	0	0	0	0	43	0	0	0	41
Chrome bookmarks	50	0	0	0	0	0	50	0	0	0	50
Chrome cookies	20	0	0	0	0	0	20	0	0	0	20
Chrome favicons	39	0	0	0	0	0	0	0	0	0	39
Chrome logins	2	0	0	0	0	0	2	0	0	0	2
Chrome top sites	40	0	0	0	0	0	0	0	0	0	0
Chrome web history	40	0	0	0	0	0	50	0	0	0	40
Chrome web visits	1	0	0	0	0	0	0	0	0	0	1
Chrome/360 Safe Browser/Opera Carved web history	0	0	0	0	0	0	1	0	0	0	0
Firefox web history	13	0	0	0	0	0	14	0	0	0	12
Google maps	10	0	0	0	0	0	11	0	0	0	10
Safari history	9	0	0	0	0	0	9	0	0	0	11

Table 18: Data recovered by IEF in Nexus 4.

PhotoRec	**Moto G**										
File type	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**
docx	47	0	0	0	0	0	47	0	0	0	0
http cache	5	0	0	0	0	0	5	0	0	0	0
jar	13	0	0	0	0	0	13	0	0	0	0
java	17	0	0	0	0	0	24	0	0	0	0
jpg	538	0	0	0	0	0	335	0	0	0	0
mp3	30	0	0	0	0	0	1	0	0	0	0
ogg	3	0	0	0	0	0	1	0	0	0	0

pdf	15	0	0	0	0	0	5	0	0	0	0
png	101	0	0	0	0	0	95	0	0	0	0
pptx	20	0	0	0	0	0	9	0	0	0	0
sqlite	9339	7	7	0	4	7	8647	3	3	7	7
txt	500	0	0	1	0	0	402	0	0	0	0
xlsx	7	0	0	0	0	0	11	0	0	0	0
zip	15	0	0	0	0	0	12	0	0	0	0

Table 19: Data recovered by PhotoRec in Moto G.

PhotoRec	**Nexus S**										
File type	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**L**
docx	0	0	0	0	0	0	0	0	0	0	0
http cache	0	0	0	0	0	0	0	0	0	1	0
jar	0	0	0	0	0	0	0	0	0	0	0
java	39	39	39	0	39	39	39	39	39	39	39
jpg	460	460	460	204	460	460	460	460	460	297	460
mp3	373	373	373	373	373	373	373	373	373	346	373
ogg	2	2	2	2	2	2	2	2	2	0	2
pdf	6	6	6	1	6	6	6	6	6	5	6
png	49	49	49	12	49	49	49	49	50	49	49
pptx	1	1	1	0	1	1	1	1	1	1	1
sqlite	1	1	1	0	1	1	1	1	1	1	5
txt	811	811	811	19	811	812	816	816	812	781	811
xlsx	1	1	1	0	1	1	1	1	1	1	1
zip	3	3	3	0	3	3	3	3	3	3	3

Table 20: Data recovered by PhotoRec in Nexus S.

PhotoRec	**Nexus 4**								
File type	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**
docx	53	0	0	0	53	0	0	0	53

http cache	31	0	0	0	29	0	0	0	17
jar	9	0	0	0	11	0	0	0	12
java	95	0	0	0	95	0	0	0	75
jpg	459	0	0	0	459	0	0	0	459
mp3	34	0	0	0	34	0	0	0	34
ogg	4	0	0	0	4	0	0	0	4
pdf	18	0	0	0	18	0	0	0	17
png	372	0	0	0	392	0	0	0	316
pptx	23	0	0	0	24	0	0	0	23
sqlite	178	0	0	0	206	0	0	0	182
txt	2577	2	2	2	2725	2	2	2	2619
xlsx	4	0	0	0	4	0	0	0	4
zip	4	0	0	0	5	0	0	0	5

Table 21: Data recovered by PhotoRec in Nexus 4.

Scalpel	**Moto G**										
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**
	885	0	0	0	0	0	616	0	0	0	0

Table 22: Thumbnail recovered by Scalpel in Moto G.

Scalpel	**Nexus S**										
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**L**
	564	564	564	365	564	564	564	564	564	457	457

Table 23: Thumbnail recovered by Scalpel in Nexus S.

Scalpel	**Nexus 4**										
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**L**
	664	0	0	0	0	0	665	0	0	0	675

Table 24: Thumbnail recovered by Scalpel in Nexus 4.

4 Thumbnail Recovery

Material presented in this chapter is based on the publication:

- Leom, MD, D'Orazio, CJ, Deegan, G & Choo, K-KR 2015, 'Forensic collection and analysis of thumbnails in Android', *Trustcom/BigDataSE/ISPA*, IEEE, pp. 1059-66, doi: 10.1109/Trustcom.2015.483.

Thumbnail is smaller representation of a larger media file such as picture and video, and has been used as evidence in a number of court cases in jurisdictions such as Australia, United Kingdom and United States. Quick, Tassone and Choo (2014, pp. 1-2) also noted that '[i]n many cases, it is the thumbnail image alone that has been the evidence presented to court'. As the resolution of digital cameras increases, picture size and storage requirement also increases. To reduce the storage requirement and increase efficiency, operating system generally renders the thumbnail cache when a user is browsing the computer. Similarly, thumbnail cache can also facilitate forensic and digital investigations as the investigators can view thumbnail images significantly faster than the original images.

Currently, "file carving" tool is often used to recover files including deleted files from the acquired forensic image, as well as files from a damaged device (e.g. recovering a corrupted Master File Table - MFT). File carving tool or file carver searches for files based on the header or footer value, file size, and file format; and typically works only for data stored contiguously. In other words, file carver generally is not able to recover and reassemble fragmented deleted files (Park, Chung & Lee 2012; Garfinkel 2007; Sajja 2010). However, due to the small size of thumbnail, deleted thumbnail is less likely to be fragmented and, therefore, file craving can potentially be used to recover deleted thumbnail.

This chapter describes a thumbnail forensic recovery process for Android devices. The utility of the proposed process is then demonstrated that is applicable even in the event that the file system is no longer accessible and that the recovered thumbnail still could be linked to the associated fragmented deleted picture (in JPEG format). In other words, the investigator would have the picture (albeit in lower quality) and the associated metadata (e.g. identifying previous whereabouts or accomplice of a terrorist suspect and determine whether a child pornography suspect possess illegal content).

4.1 Related work

Due to the widespread adoption of the JPEG format, the latter is the subject of active forensic research. Existing efforts typically focus on improving file carving technique to recover deleted JPEG images (Cohen 2008; Mohamad, Patel & Deris 2011; Mohamad & Deris 2009a; Karresand & Shahmehri 2008; Mohamad, Herawan & Deris 2010) or reassembling JPEG fragments (Guo & Xu 2011; Mohamad & Deris 2009b; Mohamad 2011; Sencar & Memon 2009; Pal, Sencar & Memon 2008; Karresand 2008; Xu & Dong 2009; Zhao et al. 2007).

In a recent work, Quick et al. (2014) provided a detailed overview of thumbnail stores for Windows platform (from Windows 95 to Windows 8) as well as the tools that can be used to view the thumbnails. The researchers also proposed an operational methodology for thumbnail analysis and a reporting and visualisation methodology and software prototype to present the findings from the thumbnail analysis. Other related work included that of Matt (2012) who demonstrated how to recover thumbnail cache from Windows Vista machines and Hurlbut (2005) on machines running Windows Me to Windows XP using FTK (AccessData).

Parsonage (2012) and Morris (2013) investigated thumbnail cache behaviour in Windows Vista and 7, and Windows 7 and Ubuntu Linux machines respectively. They demonstrated how the thumbnail is generated under user interaction with the OS. In one of few work for mobile devices, Hoog (2011d) presented their preliminary study of thumbnail cache folder in Android devices, which required the file system to be accessible. Hoog's study did not include thumbnail cache behaviour, specifically how user action affects the creation of thumbnail cache. Existing thumbnail publications are summarised in Table 25.

Morris (2013) highlighted the need to have a detailed understanding of thumbnail cache structure in order to facilitate automated extraction. Existing literature generally focus on desktop operating system (OS). Considering the increasing popularity of mobile devices and that they are becoming a popular alternative to desktop (e.g. the sales of mobile devices are reportedly three times more desktop and laptops in 2013 - see IDC 2014; Reuters 2014), this paper aims to contribute to a better forensic understanding of thumbnail cache on Android devices which is the most popular mobile OS (comScore 2014).

Publications	**Platform**
Hurlbut (2005)	Windows Me to Windows XP
Hoog (2011d)	Android
Matt (2012)	Windows Vista

Parsonage (2012)	Windows Vista and 7
Morris (2013)	Windows 7, Ubuntu Linux, and Kubuntu Linux
Quick et al. (2014)	Windows 95 to Windows 8

Table 25: Thumbnail publications by platform.

4.2 Contribution and outline of chapter

This chapter proposes a methodology for thumbnail collection and analysis from Android devices. The utility of the methodology is demonstrated using a case study. The case study first determine the characteristics of thumbnail in order to customise existing file carving tools to recover thumbnail from the forensic image in an efficient manner (e.g. by reduce the number of irrelevant files). Previous studies (Simon and Anderson 2015a; Schwamm 2014; Schwamm and Rowe 2014; McColgan 2014; The Guardian 2013; Siciliano 2012; Honan 2013) have shown that performing factory reset on Android devices does not remove the actual content of data. Therefore, the case study demonstrates that it is possible to recover thumbnails even after the photos have been deleted, a factory reset has been undertaken by a user, or a corrupted file system.

Previous studies (Morris & Chivers 2011; Parsonage 2012; Quick et al. 2014) focusing on thumbnail cache behaviour in Microsoft Windows platform have shown thumbnail can be created in thumbcache without the original picture being viewed. This implies presence of thumbnail could not prove a user has knowledge of original picture in question. However, case study (described in Chapter 4.5.3) shows certain size of thumbnail is only created after the picture has been viewed. This could possibly indicate that the user knew the existence of the picture in question.

The remainder of this chapter is organised as follows. Section 3 presents an overview of Android forensics. Section 4 outlines the proposed thumbnail forensic collection and analysis methodology for Android devices. The utility of the process is then demonstrated in Section 5. In Section 6, the potential limitations of using thumbnail in forensics and the proposed process are discussed. The last Section 7 concludes this paper and outlines future research opportunities.

4.3 Background

Android mobile devices typically consist of several partitions (see Table 26) as discussed in Chapter 2.1. Such information would be useful to a forensic investigator as it allows the forensic investigator to focus only on the relevant partition during the evidence identification process.

Name	**Mount point**	**Description**
Recovery	N/A	Recovery mode
Boot	N/A	Linux kernel
System	/system	Operating system files, system apps
Cache	/cache	Cache files
User data	/data	User installed apps
Internal SDcard (Media)	/mnt/sdcard /storage/sdcardX /data/media/X	User-accessible storage to store media files.

Table 26: Partition layout of typical Android device. Adapted from (Vidas, Zhang & Christin 2011)

Prior to Android 3.0 (Honeycomb), userdata (*/data*) and media are two separate partitions. However, in Android Honeycomb, media "partition" is no longer a partition but a subfolder in /data as */data/media*. Moto G and Nexus 4 (used in Chapter 3) have this new layout, so userdata and media partition are actually one partition formatted with F2FS and EXT4 in respective phone (Chapter 3 Table 10). Although the Nexus S (used in Chapter 3 and this chapter as well) has been upgraded to Android 4.2 from the initial Android 2.3, the partition layout still remains. This is possibly to prevent user data loss that would occur when changing file system. Thus, the /data and media partitions are still separated.

Data recovery can be undertaken using logical or physical acquisition techniques. The former allows the extraction of allocated data still accessible on the file system. The latter directly accesses the raw data in the storage medium without attempting to reconstruct the file system, as file system usually deletes the file location (unallocated) without deleting the actual content for efficiency (i.e. it is significantly faster to remove the link to the file location than the actual content).

4.4 A methodology for thumbnail collection and analysis from Android devices

This section now presents the proposed process of recovering thumbnail files from Android device (Figure 8) and map it to the digital forensic framework (McKemmish 1999).

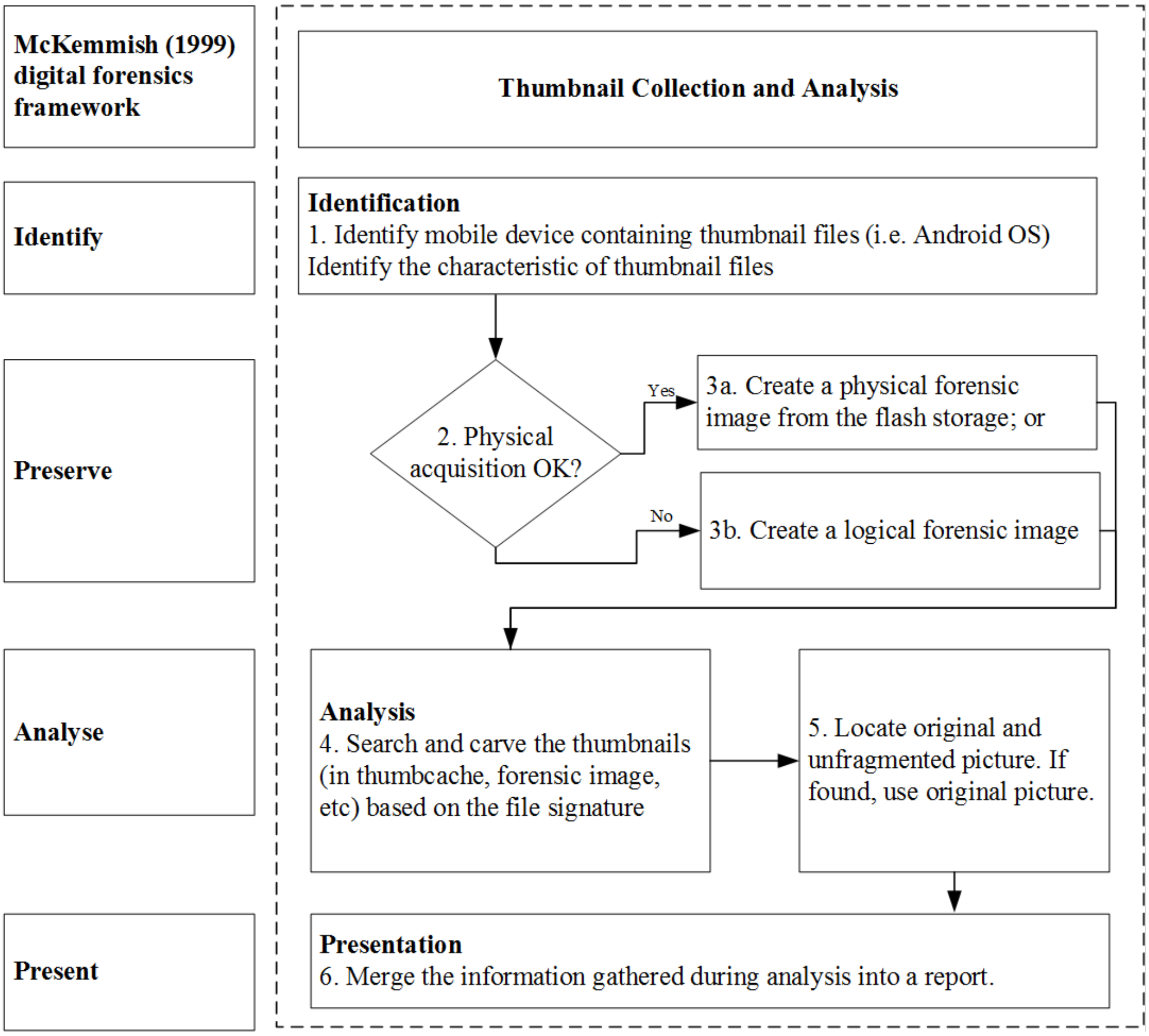

McKemmish (1999) digital forensics framework
Identify
Preserve
Analyse
Present
Thumbnail Collection and Analysis
Identification
1. Identify mobile device containing thumbnail files (i.e. Android OS)
Identify the characteristic of thumbnail files
2. Physical acquisition OK?
Yes
No
3a. Create a physical forensic image from the flash storage; or
3b. Create a logical forensic image
Analysis
4. Search and carve the thumbnails (in thumbcache, forensic image, etc) based on the file signature
5. Locate original and unfragmented picture. If found, use original picture.
Presentation
6. Merge the information gathered during analysis into a report.

Figure 8: Thumbnail forensic recovery process for Android devices

Identify

In this step, the potential evidence and the sources are identified. In the case of thumbnail recovery, an investigator will need to identify the locations of the thumbnails and, if possible, the thumbnail size as knowing the size allows one to customise the file carving tool with better accuracy. Thumbnails generated by an Android device with similar camera specification (i.e. megapixel count) should have similar resolution as well. From the thumbnail resolution, we can estimate the average size of the thumbnail. Finding the average size allows us to customise the file carving tool. It is also necessary to check whether the device has been *rooted*, which will inform actions to be undertaken in the next step.

Preserve

Wherever possible, a bit-for-bit copy of the flash memory should be undertaken. Otherwise, the forensic investigator could choose to acquire bit-for-bit copy of specific partition such as internal SDcard that are more likely to store thumbnail. The forensic image is hashed to ensure integrity throughout analysis. Bit-for-bit copy requires the Android device to have *root* access. Process of rooting often involves unlocking the bootloader and doing so will wipe the /data partition. However, in that scenario, the partition is not securely wiped and its previous content is recoverable through physical extraction (Wartickler 2012). Whether rooting is considered tampering with the evidence and consequently affect its admissibility is not part of discussion in this paper. Nonetheless, rooting may not be possible in certain model. In that case, thumbnail cache (thumbcache) files such as imgcache.0 could be logically extracted and preserved just like a cloned image.

Analyse

File carving tool can be used to locate thumbnail within the forensic image. The software should be customised according to the file signature of thumbnail file. This is necessary to reduce the number of irrelevant files recovered for easier analysis. This technique could also be employed to extract thumbnail from the recovered thumbcache file found in Android device. The recovered thumbnail can then be used to match existing fragmented file.

Presentation

Information gathered during analysis stage are documented and presented. This could comprise; (1) thumbnail picture which matches an existing file, (2) thumbnail picture which matches fragmented file, or (3) standalone thumbnail picture, in a report.

4.5 Case study

This section outlines three case studies conducted based on methodology. First case study focus on logical extraction of thumbnail. Second case study focus on physical extraction of thumbnail. Third case study focus on the behaviour of the thumbnail cache. Prior conducting these case studies, the mobile is factory reset beforehand. *USB debugging* is enabled in the device system settings. The Android mobile device is also *rooted* to access the raw data. USB debugging is required for forensic acquisition conducted on first and second case studies (Chapter 4.5.1 & 4.5.2). Below table (Table 27) outlined the hardware and software specification.

Name/Model	Description	Version/Specification
Samsung Nexus S I9020T (GSMArena n.d.)	Mobile device	Android 4.1.2 (rooted and USB debugging enabled)[19]
BusyBox (installed in mobile device)	Collection of Unix tools (e.g. nc, dd)	1.23.0, installed using BusyBox installer for Android v25 (Stericson 2015)
Gallery	Default Android media gallery app	1.1.40000
Dell Optiplex 960	Workstation	Intel Core 2 Quad Q9400 (2.66Ghz quad-core), 4GB RAM, 150GB hard disk, Windows 7 64-bit
Oxygen Forensic Suite 2014 (Oxygen Forensics n.d.)	Forensic tool	6.2.1.103 (Trial)
Netcat/nc (Pond 2004)	Forensic tool	1.11
HxD (Hörz 2009)	Hex editor	1.7.7.0
Scalpel (machn1k n.d.)	File carver	2.0
ExifTool (Harvey 2014)	JPEG EXIF metadata viewer	9.69

Table 27: Hardware and software specification

Below (Table 28) shows the file system type of each partition in the test device:

Name	File system
Recovery	YAFFS2
Boot	
Cache	
System	EXT4
User data	
Internal SDcard/Media	FAT32

Table 28: Partition's file system

4.5.1 Logical extraction of thumbnail

Logical extraction is when the file system is still accessible or during live forensic. In this case study, 10 pictures snapped using the mobile device are used as as baseline pictures (Table 29: File No. 21-30). Those pictures are viewed and deleted from *Gallery* app. After that, the

[19] More than 2 years of usage. Previously equipped with custom ROM Android 4.4 (XDA Developers n.d.a). Downgrade to stock Android 4.1.2 (Google n.d.e).

partition /storage/sdcard0 is duplicated where pictures are most likely to be stored. The partition is duplicated to the workstation using following command[20]:

```
$ adb pull sdcard/
```

The location of the thumbnail cache (thumbcache) files is then identified. There are three locations where thumbnails are stored. First is the thumbnail embedded inside a JPEG file. Second is from thumbnail generated and used by *Gallery* app found at following location:

```
/sdcard/Android/data/com.google.android.gallery3d.cache/imgcache.0
```

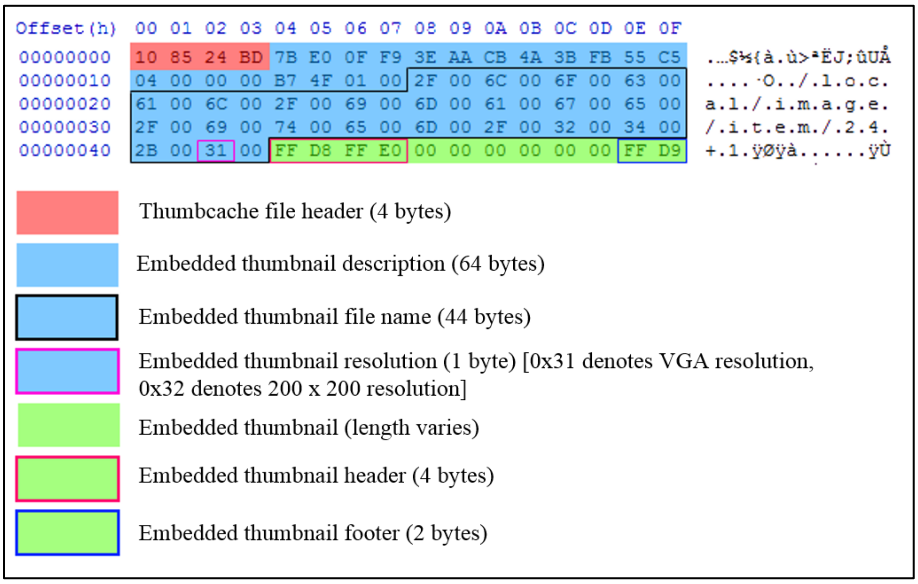
No.	**Filename**	**File size**	**Resolution**
21	IMG_20150117_175812.jpg	1,860	1920 x 2560
22	IMG_20150117_175852.jpg	1,593	1920 x 2560
23	IMG_20150117_175911.jpg	1,937	1920 x 2560
24	IMG_20150117_175930.jpg	1,281	2560 x 1920
25	IMG_20150117_175950.jpg	2,518	1920 x 2560
26	IMG_20150117_180001.jpg	2,524	1920 x 2560
27	IMG_20150117_180024.jpg	1,083	1920 x 2560
28	IMG_20150117_180050.jpg	1,393	1920 x 2560
29	IMG_20150117_180218.jpg	2,014	1920 x 2560
30	IMG_20150117_180249.jpg	1,379	1920 x 2560

Table 29: Baseline pictures

Thumbcache Viewer (Kutcher 2014a) and Thumbs Viewer (Kutcher 2014b) could not detect any thumbnail in File No. 31 since these software are designed to support thumbnail cache generated by Microsoft Windows OS, thus the tools are incompatible in this case. File No. 31 is then inspected using HxD, a hex editor. Figure 9 shows file structure of sample thumbcache file (imgcache.0). The first 4 bytes (red-highlighted) contains header information as identifier for thumbcache file. The next 64 bytes (blue-highlighted) contains description of the following thumbnail. The description contains filename as identifier of the thumbnail inside the thumbcache. Note this is different from filename of the original picture. Next is the actual thumbnail data (green-highlighted). The size varies even with similar resolution. It includes header and footer. After that is description for the next thumbnail and its thumbnail data (as

[20] ADB tool and additional driver required.

portrayed in Table 30 below). Thumbcache file does not have a unique footer value, rather the value is footer value of the last thumbnail.



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  10 85 24 BD 7B E0 0F F9 3E AA CB 4A 3B FB 55 C5  ..…$½{à.ù>ªËJ;ûUÅ
00000010  04 00 00 00 B7 4F 01 00 2F 00 6C 00 6F 00 63 00  ....·O../.l.o.c.
00000020  61 00 6C 00 2F 00 69 00 6D 00 61 00 67 00 65 00  a.l./.i.m.a.g.e.
00000030  2F 00 69 00 74 00 65 00 6D 00 2F 00 32 00 34 00  /.i.t.e.m./.2.4.
00000040  2B 00 31 00 FF D8 FF E0 00 00 00 00 00 00 FF D9  +.1.ÿØÿà......ÿÙ
```

🟥 Thumbcache file header (4 bytes)

🟦 Embedded thumbnail description (64 bytes)

🟦 Embedded thumbnail file name (44 bytes)

🟦 Embedded thumbnail resolution (1 byte) [0x31 denotes VGA resolution, 0x32 denotes 200 x 200 resolution]

🟩 Embedded thumbnail (length varies)

🟩 Embedded thumbnail header (4 bytes)

🟩 Embedded thumbnail footer (2 bytes)

Figure 9: Internal structure of thumbcache file (imgcache.0).

Thumbcache header	Thumbnail 1 description	Thumbnail 1 data	Thumbnail 2 description	Thumbnail 2 data

Table 30: Internal structure of thumbcache file (imgcache.0) with multiple thumbnails.

A thumbnail (File No. 32) is extracted by manually searching the header (**`FF D8 FF E0`**) and the footer (**`FF D9`**). The rest of the thumbnails could be recovered manually using the hex editor, but file carving tool can helps to automate this process. So, the rest of the thumbnails are extracted using Scalpel, a file carving tool due to its ease of configuration and the customisation available fit purpose of case study.

Scalpel can be used through this command:

```
C:\>scalpel -v -c conf/scalpel.conf -o output_directory forensic_image
```

Where:

-v = verbose

-c = configuration file

-o = directory to store recovered files

The default configuration file is located at *conf* folder named *scalpel.conf*. The configuration file contains rules on file carving in each line. Default configuration had all the rules commented out so running Scalpel using that configuration will not recover anything. Each rule describes the file extension, whether the header and footer are case sensitive, the minimum and maximum file size, and the header and footer value.

In this case, following rule is used:

```
jpg   y     1000: 500000 \xff\xd8\xff\xe0  \xff\xd9
```

- ***jpg*** File extension.
- ***y*** Header and footer is case sensitive. Use '*n*' for case insensitive.
- ***1000: 500000*** Carve only file size between 1,000 bytes to 500,000 bytes. Ignore file outside of this range.
- ***\xff\xd8\xff\xe0*** Header value. \x is the representation for hexadecimal character. Use \? to match any byte value (wildcard).
- ***\xff\xd9*** Footer value. Optional.

Judging by the thumbnail resolution (VGA), the thumbnail size is estimated to be at least 1KB and less than 500KB. By using the above rule, 20 thumbnails are extracted. The result (Table 31) implies that the Gallery app generate VGA (640 x 480) sized and 200 x 200 thumbnails for every pictures.

However, although File No. 21-30 are taken in portrait orientation (except File No. 24) (Table 29), yet all thumbnails are in landscape (Table 31 & Table 32).

VGA resolution thumbnail that matches with its 200 x 200 resolution counterpart (Table 32) would have similar file name (**/local/image/item/00**+1) except for the last number (/local/image/item/00+**1**). Value "1" denotes the thumbnail is VGA resolution, while "2" denotes 200 x 200 resolution (described in more details in Figure 9).

No.	Image resolution	Filename	File size
31	N/A	imgcache.0	972.0
32	640 x 480	/local/image/item/24+1	83.9
33		/local/image/item/25+1	71.3
34		/local/image/item/26+1	83.5
35		/local/image/item/27+1	62.8
36		/local/image/item/28+1	129.2
37		/local/image/item/29+1	157.5
38		/local/image/item/30+1	39.8
39		/local/image/item/31+1	46.4
40		/local/image/item/32+1	114.8
41		/local/image/item/33+1	55.9
42	200 x 200	/local/image/item/33+2	7.7
43		/local/image/item/31+2	6.7
44		/local/image/item/32+2	16.8

45		/local/image/item/30+2	6.7
46		/local/image/item/29+2	20.8
47		/local/image/item/28+2	18.7
48		/local/image/item/27+2	11.4
49		/local/image/item/26+2	12.3
50		/local/image/item/25+2	11.8
51		/local/image/item/24+2	12.0

Table 31: Thumbnails extracted from imgcache.0 file.

Original (File No.)	Thumbnail (File No.)	Thumbnail (File No.)
21	32	51
22	33	50
23	34	49
24	35	48
25	36	47
26	37	46
27	38	45
28	39	43
29	40	44
30	41	42

Table 32: Original file and its thumbnail.

In order to extract the embedded thumbnail from a JPEG file, ExifTool is utilised. Although such tool is available, the manual method employed in extracting thumbnails from File No. 31 can work in this case as well.

The command used to extract thumbnail is as follows:

```
C:\> exiftool –b –ThumbnailImage input > output
```

*When using under Microsoft Windows, rename "`exiftool (-k).exe`" to "`exiftool.exe`".

Where:

-b = Output the requested data in binary format without tag names or descriptions.

-ThumbnailImage = Read thumbnail image

input = The location of original image.

output = The location to save the extracted thumbnail in .jpg extension.

Below (Table 33) shows the information on the extracted thumbnail stored in the 10 baseline pictures (File No. 21-30).

No.	File size (KB)	Resolution
52	13.1	320 x 240
53	13.4	
54	13.4	
55	13.0	
56	21.9	
57	27.6	
58	7.0	
59	6.5	
60	21.4	
61	7.8	

Table 33: Thumbnails extracted from original JPEG file.

4.5.2 Physical extraction of thumbnail cache

This section demonstrates method to physically extract the thumbnail from the raw forensic image. The /media[21] partition path is identified as such (highlighted):

```
C:\> adb shell
shell@android:/ $ su
root@android:/ # ls -l /dev/block/platform/s3c-sdhci.0/by-name
lrwxrwxrwx root     root   2015-01-17 10:15 media -> /dev/block/mmcblk0p3
lrwxrwxrwx root     root   2015-01-17 10:15 system -> /dev/block/mmcblk0p1
lrwxrwxrwx root     root   2015-01-17 10:15 userdata -> /dev/block/mmcblk0p2
```

The forensic image of that partition is created using following commands:

```
C:\> adb forward tcp:5555 tcp:5555
C:\> adb shell
shell@android:/ $ su
root@android:/ # nc -l -p 5555 -e dd if=/dev/block/mmcblk0p3
```

(Note[22])

On another command prompt:

```
C:\> adb forward tcp:5555 tcp:5555
C:\> nc 127.0.0.1 5555 > mmcblk0p3.raw
```

Scalpel is utilised to recover thumbnails from the forensic image. The rule is customised to target thumbnails only based on the results gathered in Chapter 4.5.1 (Table 31). The header information is inspected to determine the maximum file size of the thumbnails. The purpose is to determine the parameters that will be used in file carving. The results show different type of thumbnail has different header value starting from 4th byte. Table 34 below illustrate the difference. Do note the header value for thumbnail cache shown in the table is not the header value of the thumbcache file, but rather the individual thumbnail stored inside the file.

Type	Header	Footer	Maximum file size
Thumbnail stored in thumbcache file	FF D8 FF E0	FF D9	157.5 KB
Embedded thumbnail in JPEG file	FF D8 FF DB		27.6 KB

Table 34: Header value and file size of thumbnail.

[21] The partition is also mounted as /storage/sdcard0, same partition used in Section Logical extraction of thumbnail.

[22] *nc* and *dd* are installed through BusyBox.

Based on the Table 34, the rule is customised as follow:

```
#1 rule
jpg y   1000:500000     \xff\xd8\xff\xe0        \xff\xd9
#2 rule
jpg y   1000:50000      \xff\xd8\xff\xdb        \xff\xd9
```

The first rule (*#1*) is to recover thumbnails from thumbnail cache file. 1KB is used as minimum size and 500KB is used as maximum size. The second rule (*#2*) is to recover embedded thumbnail. 1KB is used as minimum size and 50KB is used as maximum size.

Below (Table 35) shows the result of the thumbnails recovered.

Rule	Thumbnail type	Thumbnails recovered	Percentage
#1	200 x 200 resolution thumbnail in thumbcache	**10**/10	100%
	VGA resolution thumbnail in thumbcache	**3**/10 (**9**/10 if include fragmented thumbnail)	30%
#2	Embedded thumbnail in JPEG file	**10**/10	100%

Table 35: Recovery result.

The results show #2 rule is very effective at recovering thumbnails. The rule managed to recover thumbnail of all the test files (File No. 1 to 10). #1 rule also managed to recover all 200 x 200 resolution thumbnails of all the test files, but it is less successful on VGA resolution thumbnail. However, the recovery percentage is up to 90% if fragmented thumbnail is included. This shows larger thumbnail is more likely to be fragmented. Nevertheless, the overall result shows that thumbnail is significantly less likely to be fragmented compared to original image.

4.5.3 How user actions affect creation of thumbnail cache?

Previous study (Hoog 2011d) did not investigate how user action would affect creation of thumbnail cache. This section identifies the behaviour of the thumbcache when user interacts with the OS. To establish the behaviour of the Android thumbnail cache it is necessary to perform a variety of experiments; the experiments establish the way the thumbnail is generated based upon user activity. Prior to experiments in this section, the Android device is factory reset to clear the thumbcache. After factory reset, the device is connected to Wi-Fi and signed in with Google account. No third-party app nor any update are installed throughout this section. After each experiment, thumbcache file (located at

/sdcard/Android/data/com.google.android.gallery3d.cache/imgcache.0) is copied to the workstation for analysis.

Experiment		**Result**
1	Take 10 pictures using default Camera app.	Found 8 VGA-sized thumbnails.
2	Launch Gallery app.	Previous thumbnails plus a 200x200 thumbnail.
3	Open the "Camera" (/sdcard/DCIM/Camera) folder.	Previous thumbnails plus 9 200x200 thumbnail.
4	View first picture.	Previous thumbnails plus 2 VGA-sized thumbnails.
5	View first to fifth picture.	No difference.
6	View first to tenth picture.	No difference.
7	Delete 5 pictures in odd number.	No difference.
8	Delete the remaining 5 pictures.	No difference.
9	Take 5 pictures.	Previous thumbnails plus 4 VGA-sized thumbnails.
10	Copy 31 pictures into "Pictures" (/sdcard/Pictures) folder.	No difference.

Table 36: Thumbcache behaviour in Android.

The experiments result (Table 36) shows VGA-sized thumbnail is generated when the picture is snapped but not for all pictures. When the gallery app is launched, a smaller size (200x200) thumbnail is generated. This thumbnail functions as the camera folder "cover". When the folder is opened, all pictures are shown in "album view", and at the same time the remaining 200x200 thumbnails of the ten pictures are generated. The two remaining VGA-sized thumbnail is only generated after viewing the first picture. The thumbnail is not deleted even though the original image has been removed. No thumbnail is generated when there is new pictures is saved, that are not taken by the camera.

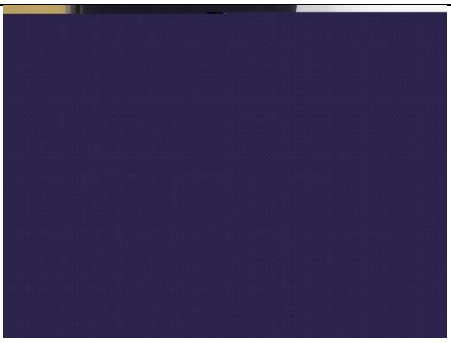
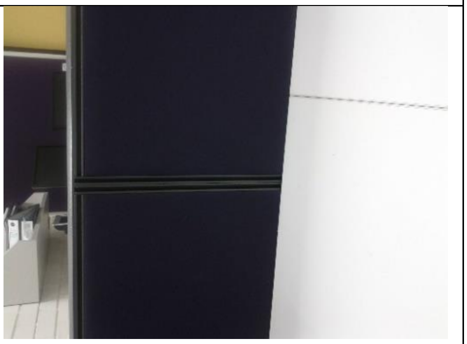
The implication of the results above is the possibility of using thumbnail as indication to determine whether the picture has been viewed or not.

4.6 Discussion

Recovered thumbnail can provide valuable visual clue to investigator. It can be matched with its original picture, especially the original picture's metadata. The metadata is still intact even when the original picture is heavily fragmented, it still contains complete Exchangeable image file format (Exif) metadata (see Tachibanaya 1999 for Exif). This is demonstrated in the following experiment. 10 pictures (File No. 1-10) are snapped using the mobile device and then perform physical acquisition of the mobile device using Oxygen Forensic Suite, a forensic tool. The pictures are manually extracted from the forensic image through following process:

1. Search for the first 8-byte value of the header in hexadecimal form of the intended picture. Mark the location the first byte of the found 8-byte header as START_OFFSET.
2. Select data block with the same value as the size of original picture (LENGTH=size of original picture).
3. Copy out the selected block of data.
4. Save the copied block of data with file extension ".jpg".
5. Inspect the saved file using Windows Photo Viewer.
6. Verify MD5 hash of original and recovered picture.

File No. 1, 2, 7, 8, and 9 can be fully recovered. The rest (File No. 3, 4, 5, 6, and 10) are fragmented (Table 37). In other words, out of 10 pictures, only half is not fragmented.

No.	Recovered	No.	Original
13		3	

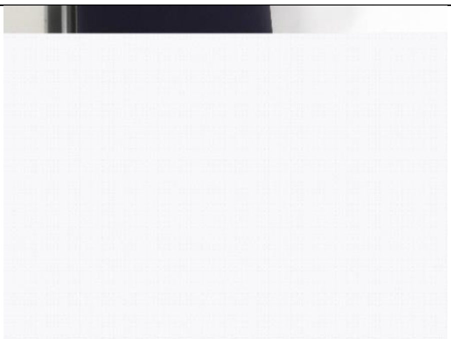
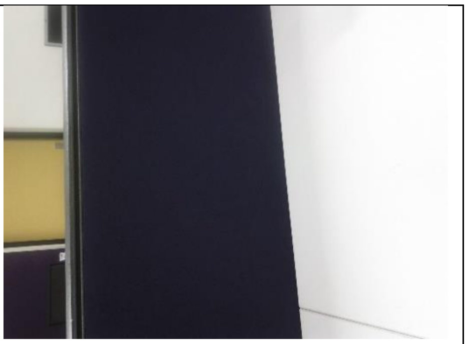
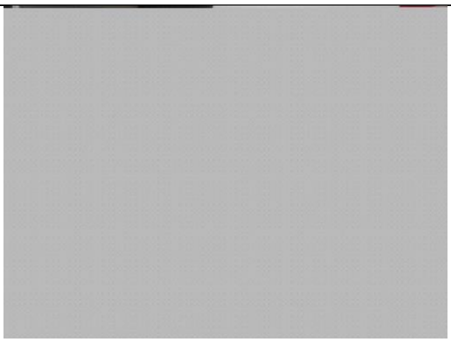
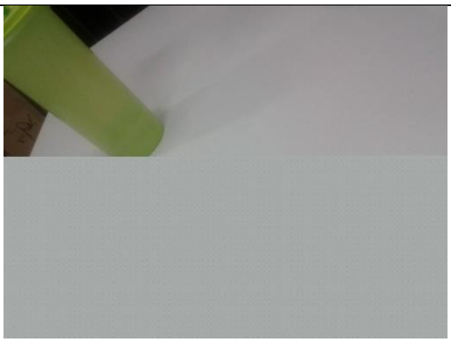
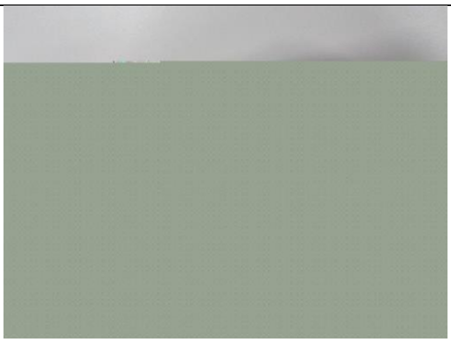
14		4	
15		5	
16		6	
20		10	

Table 37: Recovered fragment and its original.

ExifTool is utilised to determine existence of Exif metadata in File No 15. The results show that although heavily fragmented JPEG could only provide very limited visual clue but at least the EXIF metadata is not likely to be fragmented, thus recoverable. Below show excerpt of EXIF metadata in File No. 15 (the most fragmented among those shown in Table 37).

```
C:\>exiftool 15.jpg
ExifTool Version Number         : 9.69
File Name                       : 15.jpg
Directory                       : C:\Recovered fragment
File Size                       : 921 kB
File Type                       : JPEG
MIME Type                       : image/jpeg
Exif Byte Order                 : Little-endian (Intel, II)
Make                            : google
Camera Model Name               : Nexus S
Orientation                     : Rotate 90 CW
Software                        : JZO54K
Modify Date                     : 2014:06:19 18:06:06
Y Cb Cr Positioning             : Centered
Exposure Time                   : 1/33
F Number                        : 2.6
Exposure Program                : Aperture-priority AE
ISO                             : 50
Exif Version                    : 0220
Date/Time Original              : 2014:06:19 18:06:06
Create Date                     : 2014:06:19 18:06:06
```

By combining the metadata with a fully recovered thumbnail, these information are as valuable as original picture, since it still shows similar visual information, only with lower resolution. Example shown here, although a bank logo can be found in Figure 10, but it can refers to any office tower of the bank. In contrast, Figure 11 although is smaller, the location could be identified by the unique appearance of the shop on the left part. This can be further confirmed with location data from the Exif metadata.

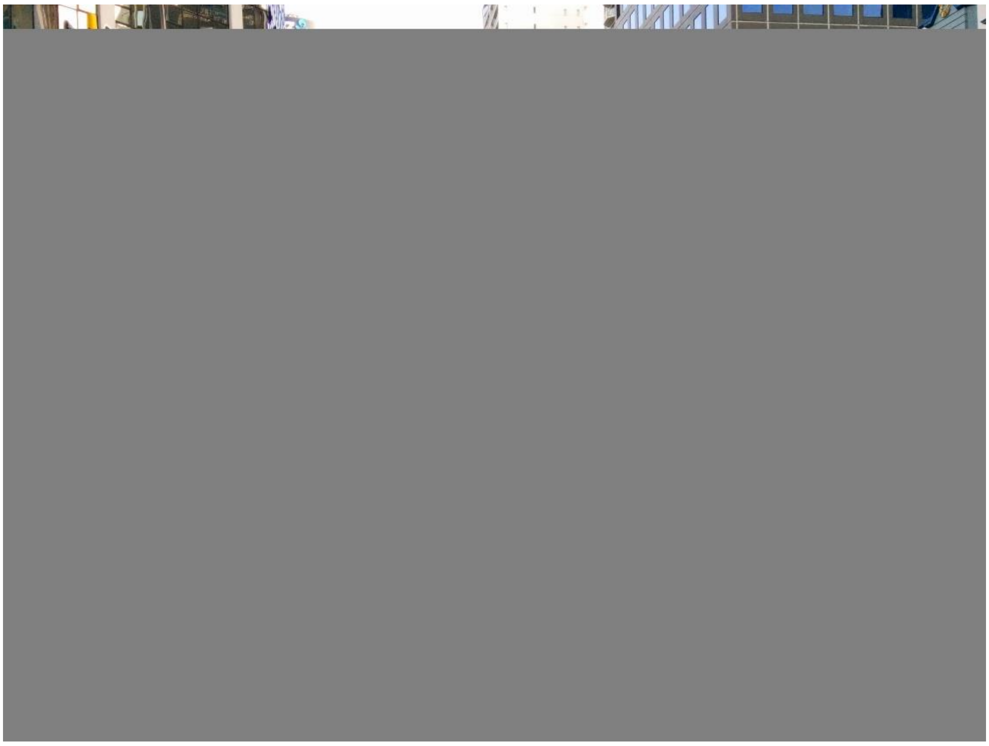


Figure 10: (Simulated) Heavily fragmented picture.

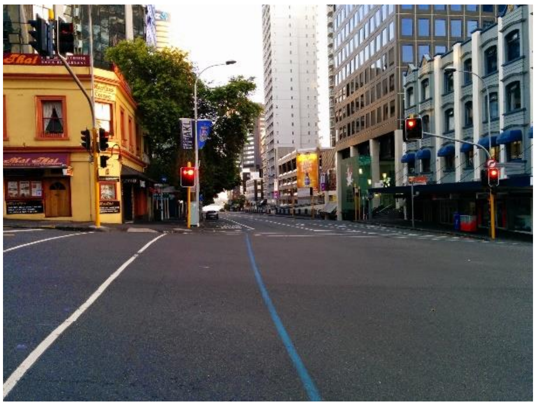


Figure 11: Non-fragmented. (Resized to VGA resolution to simulate a thumbnail)

Although it is best to have a non-fragmented thumbnail, but there is still likelihood of fragmented thumbnail (as shown in Section Physical extraction of thumbnail cache). However, since thumbnail is notably smaller than original picture (VGA vs 5 megapixel), thus less likely to be fragmented. If the thumbnail is not heavily fragmented, it could still contains some unique objects to identify the location of the photo taken. For instance, a simulated fragmented thumbnail depicted below (Figure 12) contains popular landmarks (Royal Malaysia Police Headquarter and Merdeka Square) on the left part. Using the size of the landmarks in image, the distance from the landmark could be estimated. Shops building on the right part has pretty unique roof which should be recognisable for locals.

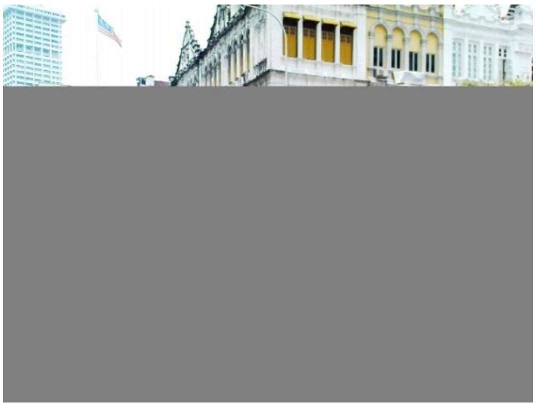


Figure 12: (Simulated) Fragmented thumbnail.

Identifying location can be crucial in forensic investigation. For example, previous whereabouts of a terrorist suspect could be determined. This could either helps to determine whether the suspect is suspicious, or helps to locate popular rendezvous locations for extremists. In those cases, criminal's accomplice might even be identified. In other case such as child pornography, since the thumbnail is retained even after the original picture has been deleted (as demonstrated by the case study described in Chapter 4.5.3), thumbnail can be employed to determine the existence of illegal content in suspect's mobile device. Thumbnail could also be used to determine the authenticity of the original photo, whether the photo has been modified (Kee & Farid 2010). This is because image manipulation program usually do not update the thumbnail after edit, leaving original thumbnail still intact (Murdoch & Dornseif 2004).

4.7 Summary

In this chapter, the location of thumbnail cache file "`imgcache.0`" and technique to extract thumbnails from that file have been described. The file structure of the thumbcache file is then outlined. After understanding the file structure, the technique to recover embedded thumbnail and the property of the embedded thumbnail are then described. The experiment result suggests that the technique is effective in recovering thumbnail even when the file system is heavily fragmented. A case study conducted show the possibility of fragmented JPEG still holding important metadata and when link to thumbnail, is akin to recovering the full picture.

Although the experiments in this chapter are conducted on Android mobile device, the proposed method could also apply to traditional desktop forensic. The proposed method is also demonstrated its effectiveness on bi-fragmented JPEG. Bi-fragmented is when a file is fragmented into two parts. Study (Garfinkel 2007) showed that it is the most common type of fragmentation in hard disk, thus showing the potential of the proposed method on desktop forensic.

5 Conclusion

Chapter 3 compared the effectiveness of seven third-party remote wiping app against Android built-in's, Android Device Manager (ADM). Most of the third-party apps performed similarly to ADM where secure deletion is utilised when supported. However, some apps have design flaw where user is not informed to grant necessary permission. This leaves user having an insecure remote wiping setup. The results also demonstrate the risk of running older version of Android where secure deletion is not available. Significant user data remnant could still be recovered after factory reset/remote wipe when the data is not securely deleted.

Chapter 4 described the location of thumbnail cache file and technique to extract thumbnails from that file. The chapter also describe the file structure of the thumbcache file. The technique to recover embedded thumbnail and the property of the embedded thumbnail have been described. A case study conducted demonstrates that those techniques are effective in recovering thumbnail even when the file system is heavily fragmented. There is also possibility of fragmented JPEG still holding important metadata and when link to thumbnail, is akin to recovering the full picture.

5.1 Future work

Despite the risk of running insecure remote wipe setup as shown in Chapter 3, the results also suggest the effectiveness of secure deletion. When secure deletion is utilised, user data could not be recovered. While the feature is beneficial for consumer, it could be abused by adversary to remove discriminating evidence. This could hinder law enforcement as previously indicated in Chapter 2.5. Although law enforcement agency can always consider Internet as source of evidence (e.g. Google n.d.b) as crime increasing involves online activity, but the mobile device still holds majority of user data. There also could be some delay for tech company to provide the request data.

In this thesis, software-based physical acquisition was employed. There might be some limitation in this approach. In future work, researchers could investigate the effect of secure deletion on hardware-based physical acquisition, if any, which can be compared to software-based.

The case studies conducted in Chapter 4 only focus thumbnail cache in Android generated by *Gallery* app. In future, author hopes to extend the research to other Android gallery apps such as Photos app bundled with Google+ app (Google 2014b) and custom gallery apps shipped by vendors. The research can be extended to newer Android versions as well. The case studies are conducted on single Android device. In theory, Android device with similar camera specification should have similar thumbnail size as well, while Android device equipped with camera that has higher resolution could result in larger thumbnail size. Thus, the research can be extended to include more Android devices for more comprehensive studies.

Instead of relying on manual matching, linking the thumbnail to original image can be automated through computer algorithm to match the thumbnail to the fragmented JPEG. This idea is similar to (Guo & Xu 2011) but that work focus on using thumbnail to rearrange JPEG fragments. There is also possibility of recovering fragmented thumbnail. Feasibility of such approach can be evaluated in future work.

6 References

Crimes Act 1914 (Cwlth) (Australia), s 3LA.

AccessData *Forensic toolkit*, viewed 28 August 2014, <http://accessdata.com/products/computer-forensics/ftk>.

Adusumalli, P 2014, 'Implementation of an android application to retrieve information from a lost android device', Graduate project report, Texas A&M University-Corpus Christi, Corpus Christi, Texas.

Agrawal, N, Prabhakaran, V, Wobber, T, Davis, JD, Manasse, M & Panigrahy, R 2008, 'Design tradeoffs for SSD performance', in *Proceedings of the USENIX technical conference*, USENIX, pp. 57-70.

Al-Zarouni, M 2007, 'Introduction to mobile phone flasher devices and considerations for their use in mobile phone forensics', in *Proceedings of the 5th Australian digital forensics conferenc*e, Edith Cowan University, Perth.

Albano, P, Castiglione, A, Cattaneo, G & De Santis, A 2011, 'A novel anti-forensics technique for the android OS', in *International conference on broadband and wireless computing, communication and applications*, IEEE, pp. 380-385.

AMD 2010, *RAIDXpert user manual,* viewed 23 August 2014, <http://www2.ati.com/relnotes/AMD_RAIDXpert_User_v2.1.pdf>.

AMTA 2011, *Lost and stolen phones*, Australian Mobile Telecommunications Association, viewed 6 June 2014, <http://www.amta.org.au/pages/Lost.and.stolen.phones>.

Anderson, N 2012, *Hacking Scarlett Johansson—and 50 other celebs—using google and gumption,* Ars Technica, viewed 5 September 2014, <http://arstechnica.com/tech-policy/2012/03/hacking-scarlett-johanssonand-50-other-celebsusing-google-and-gumption/>.

Android Wiki 2013, viewed 17 May 2015, <http://android-dls.com/wiki/index.php?title=HOWTO:_Unpack%2C_Edit%2C_and_Re-Pack_Boot_Images>.

Aouad, L & Kechadi, T 2012, 'Android forensics: A physical approach', in *The 2012 international conference on security and management*.

Angelo, M, Novoa, M & Olarig, S 2003, *After the fact protection of data in remote personal and wireless devices,* US20030065934A1, USA.

Aomoth, D 2010, *App of the week: Find my iPhone,* TIME, viewed 15 June 2014, <http://techland.time.com/2010/11/23/app-of-the-week-find-my-iphone/>.

Apple n.d.a, *About configuration profile,* viewed 18 September 2014, <https://help.apple.com/configurator/mac/1.6/#/cadbf9e668>.

Apple n.d.b, *iPhone in business: Manage devices,* viewed 18 September 2014, <https://www.apple.com/iphone/business/it/management.html>.

Apple 2014, *iOS security,* viewed 5 September 2014, <https://ssl.apple.com/ipad/business/docs/iOS_Security_Feb14.pdf>.

References

Apple 2010, *iOS 4.2 software update*, Apple, Inc, viewed 15 June 2014, <http://support.apple.com/kb/DL1061>.

Armadeo, R 2014, The history of android: The endless iterations of Google's mobile OS, Ars Technica, viewed 16 June 2014, <http://arstechnica.com/gadgets/2014/06/building-android-a-40000-word-history-of-googles-mobile-os/>.

Australian Signals Directorate 2014, *2014 information security manual*.

Barrett, D, Yadron, D & Wakaba, D 2014, *Apple and others encrypt phones, fueling government standoff*, Wall Street Journal, viewed 20 November 2014, <http://online.wsj.com/articles/apple-and-others-encrypt-phones-fueling-government-standoff-1416367801>.

Belfiore, J 2012, *Announcing windows phone 8,* Windows Blog, viewed 20 August 2014, <http://blogs.windows.com/bloggingwindows/2012/06/20/announcing-windows-phone-8/>.

Björnheden, M 2009, *The android boot process from power on,* Xdin Android Blog, viewed 17 May 2015, <http://www.androidenea.com/2009/06/android-boot-process-from-power-on.html>.

BlackBerry 2014a, *BlackBerry device service advanced administration guide (v10.2.4),* viewed 17 September 2014, <http://docs.blackberry.com/en/admin/deliverables/66537/BES10_v10.2.4_BDS_Advanced_Admin_Guide_en.pdf>.

BlackBerry 2014b, *BlackBerry device service solution security technical overview (v10.2.4),* viewed 5 September 2014, <http://docs.blackberry.com/en/admin/deliverables/66547/BES10_v10.2.4_BDS_Security_Technical_Overview_en.pdf>.

BlackBerry 2014c, *Security technical overview of BES10 cloud solution,* viewed 18 July 2014, <http://docs.blackberry.com/en/admin/deliverables/16650/BlackBerry_Enterprise_Server-Security_Technical_Overview--1153051-0615043613-001-5.0.2-US.pdf>.

BlackBerry 2013, *BlackBerry protect user guide (v1.2.1)*, viewed 17 September 2014, <http://docs.blackberry.com/en/smartphone_users/deliverables/49400/BlackBerry_Protect-User_Guide-1343391475156-1.2.1-en.pdf>.

BlackBerry 2011, *Blackberry enterprise solution 5.0.2: Security technical overview,* viewed 18 July 2014, <http://docs.blackberry.com/en/admin/deliverables/16650/BlackBerry_Enterprise_Server-Security_Technical_Overview--1153051-0615043613-001-5.0.2-US.pdf>.

BlackBerry 2010, *Enforcing encryption of internal and external file systems on BlackBerry devices*, viewed 20 August 2014, <http://docs.blackberry.com/en/admin/deliverables/3940/file_encryption_STO.pdf>.

Bonetti, G, Viglione, M, Frossi, A, Maggi, F & Zanero, S 2014, 'Black-box forensic and antiforensic characteristics of solid-state drives', *Journal of Computer Virology and Hacking Techniques*, vol. August 2014, pp. 1-17.

Breeuwsma, M, Jongh, MD, Klaver, C, Knijff, RVD & Roeloffs, M 2007, 'Forensic data recovery from flash memory', *Small Scale Digital Device Forensics Journa*l, vol. 1, no. 1, pp. 1-17.

Brown, MK, Brown, MS, Little, HA & Totzke, SW 2011, *Selectively wiping a remote device,* US8056143B2, USA.

Brož, M & Matyáš, V 2014, 'The TrueCrypt on-disk format-an independent view', *IEEE Secur Priv*, vol. 12, no. 3, pp. 74-77.

Bunker, T, Wei, M & Swanson, SJ 2012, *Ming II: A flexible platform for nand flash-based research,* Department of Computer Science and Engineering, University of California, San Diego.

Burnett, RD, Friedman, M & Rodriguez, RP 2011, 'Managing laptop security', *Journal of Corporate Accounting & Finance*, vol. 22, no. 5, pp. 53-61.

Caldwell, T 2011, 'The mobile ‘kill pill’ – poison or panacea?', *Computer Fraud & Security*, vol. 2011, no. 10, pp. 8-12.

Cannon, T 2012, *Into the droid: Gaining access to Android user data*, DEFCON, viewed 12 April 2015, <https://archive.org/stream/Defcon20Slides/DEFCON-20-Cannon-Into-The-Droid>.

Cardwell, GS 2011, 'Residual network data structures in android devices', Master's thesis, Naval Postgraduate School, Monterey, California, USA.

Chester, B & Ho, J 2014, *Encryption and storage performance in android 5.0 lollipop,* AnandTech, viewed 18 May 2015, <http://www.anandtech.com/show/8725/encryption-and-storage-performance-in-android-50-lollipop>.

Choi, Y, Lee, D, Jeon, W & Won, D 2014, 'Password-based single-file encryption and secure data deletion for solid-state drive', in *Proceedings of the 8th international conference on ubiquitous information management and communicatio*n, ACM, pp. 1-7.

Cohen, K 2007, 'Digital still camera forensics', *Small Scale Digital Device Forensics Journa*l, vol. 1, no. 1, pp. 1-8.

Cohen, MI 2008, 'Advanced JPEG carving', in *Proceedings of the 1st international conference on forensic applications and techniques in telecommunications, information, and multimedia and worksho*p, ICST, pp. 16:1-16:6.

comScore 2014, *comScore reports august 2014 U.S. smartphone subscriber market share,* viewed October 7 2014, <http://www.comscore.com/Insights/Market-Rankings/comScore-Reports-August-2014-US-Smartphone-Subscriber-Market-Share>.

Cornell University 2012, *Disk and file erasure,* Best Practices for Media Destruction, viewed 18 August 2014, <http://www.it.cornell.edu/security/depth/practices/media_destruct.cfm#erasure>.

Cunningham, A 2015, *Android M makes another attempt at automated device backups,* Ars Technica, viewed 29 May 2015, <http://arstechnica.com/gadgets/2015/05/android-m-makes-another-attempt-at-automated-device-backups/>.

Diesburg, SM 2012, 'Per-file full-data-path secure deletion for electronic storage', PhD dissertation, Florida State University, Tallahassee, Florida, USA.

Diesburg, SM, Meyers, C, Stanovich, M, Mitchell, M, Marshall, J, Gould, J, Wang, AA & Kuenning, G 2012, 'TrueErase: Per-file secure deletion for the storage data path', in

*Proceedings of the 28th annual computer security applications conferenc*e, ACM, pp. 439-448.

DON CIO Privacy Team 2010, *Methods for hard Drive/Disk destruction,* Department of Navy Chief Information Officer, viewed 18 August 2014, <http://www.doncio.navy.mil/ContentView.aspx?ID=1867>.

Ducklin, P 2014, *Anatomy of a "goto fail" - apple's SSL bug explained, plus an unofficial patch for OS X!,* Naked Security, viewed 12 November 2014, <https://nakedsecurity.sophos.com/2014/02/24/anatomy-of-a-goto-fail-apples-ssl-bug-explained-plus-an-unofficial-patch/>.

Dutta, K n.d., *ClockWorkMod (CWM) recovery,* viewed 15 August 2014, <http://www.clockworkmod.com/rommanager>.

Elenkov, N 2015, *Hardware-accelerated disk encryption in Android 5.1,* viewed 18 May 2015, <http://nelenkov.blogspot.com/2015/05/hardware-accelerated-disk-encryption-in.html>.

Elenkov, N 2014a, 'Chapter 10: Device Security', in *Android security internals: An in-depth guide to Android's security architecture,* No Starch Press, San Francisco, pp. 251-288.

Elenkov, N 2014b, *Revisiting Android disk encryption,* viewed 18 May 2015, <http://nelenkov.blogspot.com/2014/10/revisiting-android-disk-encryption.html>.

Erlichman, J & Miller, H 2014, *BlackBerry CEO briefs white house to keep Obama loyal,* Bloomberg, viewed 19 June 2014, <http://www.bloomberg.com/news/2014-03-06/blackberry-ceo-briefs-white-house-to-cultivate-vip-obama.html>.

Evers, J & Johnston, CJ 2005, 'Chapter 1: System Architecture', in *Professional BlackBerry,* Wiley Publication, Indianapolis, USA, pp. 3-18.

Fahl, S, Harbach, M, Muders, T, Baumgärtner, L, Freisleben, B & Smith, M 2012, 'Why eve and mallory love android: An analysis of Android SSL (in)security', in *Proceedings of the 2012 ACM conference on computer and communications securit*y, ACM, pp. 50-61.

Franco, F 2015, *Continuing the tale of "nexus 6 is way smoother on Android 5.1",* viewed 18 May 2015, <https://plus.google.com/+FranciscoFranco1990/posts/3RKjDGjjPQ7>.

Frizell, S 2014, *Yahoo is making it harder for the NSA to read your emails,* TIME, viewed 20 October 2014, <http://time.com/3092881/email-encryption/>.

Gajdos, T & Kretz, M 2006, *Method for disabling a mobile device,* EP1725056A1, EU.

Gallagher, S 2014, *What Jennifer Lawrence can teach you about cloud security,* Ars Technica, viewed 3 September 2014, <http://arstechnica.com/security/2014/09/what-jennifer-lawrence-can-teach-you-about-cloud-security/>.

Garfinkel, SL 2007, 'Carving contiguous and fragmented files with fast object validation', *Digital Investigatio*n, vol. 4, no. Supplement, pp. S2-S12.

Georgiev, M, Iyengar, S, Jana, S, Anubhai, R, Boneh, D & Shmatikov, V 2012, 'The most dangerous code in the world: Validating SSL certificates in non-browser software', in *Proceedings of the 2012 ACM conference on computer and communications securit*y, ACM, pp. 38-49.

Godfrey, R 2012, *Announcing Windows Phone 8,* Microsoft UK Schools Blog, viewed 20 August 2014, <http://blogs.msdn.com/b/ukschools/archive/2012/06/27/announcing-windows-phone-8.aspx>.

Google n.d.a, *Implementing security,* viewed 22 May 2015, <https://source.android.com/devices/tech/security/implement.html>.

Google n.d.b, *Legal process - Google transparency report,* viewed 29 May 2015, <https://www.google.com/transparencyreport/userdatarequests/legalprocess/>.

Google n.d.c, *Android security overview,* viewed 5 September 2014, <http://source.android.com/devices/tech/security/index.html>.

Google n.d.d, *Device administration,* viewed 17 September 2014, <https://developer.android.com/guide/topics/admin/device-admin.html>.

Google n.d.e, *Factory images for Nexus devices,* viewed 11 August 2014, <https://developers.google.com/android/nexus/images>.

Google 2014a, *A sweet lollipop, with a kevlar wrapping: New security features in Android 5.0.* Official Android Blog, viewed 31 October 2014, <http://officialandroid.blogspot.com/2014/10/a-sweet-lollipop-with-kevlar-wrapping.html>.

Google 2014b, *Google+ for android,* Google Play, viewed 16 August 2014, <https://play.google.com/store/apps/details?id=com.google.android.apps.plus>.

Google 2010, *Saving data safely,* viewed 29th April 2014, <http://android-developers.blogspot.com/2010/12/saving-data-safely.html>.

Götzfried, J & Müller, T 2014, 'Analysing Android's full disk encryption feature', *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Application*s, vol. 5, no. 1, pp. 84-100.

Götzfried, J & Müller, T 2013, 'ARMORED: CPU-bound encryption for android-driven ARM devices', in *Eighth international conference on availability, reliability and securit*y, IEEE, pp. 161-168.

GSMArena n.d., *Samsung google nexus S,* viewed 10 August 2014, <http://www.gsmarena.com/samsung_google_nexus_s-3620.php>.

Guo, H & Xu, M 2011, 'A method for recovering JPEG files based on thumbnail', in *International conference on Control, automation and systems engineerin*g, WASET, pp. 1-4.

Greenberg, A 2014, *The police tool that pervs use to steal nude pics from Apple's iCloud,* Wired, viewed 5 September 2014, <http://www.wired.com/2014/09/eppb-icloud/>.

Gross, D 2010, *Google quietly kills its once-hyped nexus one phone,* CNN, viewed 2 August 2014, <http://edition.cnn.com/2010/TECH/mobile/07/19/nexus.one.discontinued/index.html>.

Grupp, LM, Caulfield, AM, Coburn, J, Davis, JD & Swanson, S 2010, 'Beyond the datasheet: Using test beds to probe non-volatile memories' dark secrets', in *Workshop on application of communication theory to emerging memory technologie*s, IEEE, pp. 1930-1935.

Guo, C, Wang, HJ & Zhu, W 2004, 'Smart-phone attacks and defenses', in *Third workshop on hot topics in network*s, ACM.

Gupta, A, Singhal, M, Gangil, A & Mishra, A 2011, 'SDC: Secure deletion classification', in *International conference on Recent trends in information technolog*y, IEEE, pp. 1303-1307.

Gustin, S 2013, *NSA spying scandal could cost U.S. tech giants billions,* TIME, viewed 20 October 2014, <http://business.time.com/2013/12/10/nsa-spying-scandal-could-cost-u-s-tech-giants-billions/>.

Gutmann, P 2003, *Secure deletion of data from magnetic and solid-state memory, Department of Computer Science, University of Auckland, New Zealand, viewed 27th March 2014, <https://www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html>*.

Gutmann, P 1996, 'Secure deletion of data from magnetic and solid-state memory', in *Proceedings of the sixth USENIX security symposiu*m, USENIX.

Guyot, C, Bandic, ZZ, Cassuto, Y, Espeseth, AM & Sanvido, M 2012, *Implementing secure erase for solid state drives*, US8250380B2, USA.

Halderman, JA, Schoen, SD, Heninger, N, Clarkson, W, Paul, W, Calandrino, JA, Feldman, AJ, Appelbaum, J & Felten, EW 2009, 'Lest we remember: Cold-boot attacks on encryption keys', *Communications of the ACM*, vol. 52, no. 5, pp. 91-98.

Hannay, P, Carpene, C, Valli, C, Woodward, A & Johnstone, M 2013, 'Exchanging demands: Weaknesses in SSL implementations for mobile platforms', in *Proceedings of the 11th australian information security management conferenc*e, SRI Security Research Institute, Edith Cowan University, pp. 42.

Hansen, CK 2010, *Technology trends in mobile communications how mobile are your data?* IEEE Reliability Society 2009 Annual Technology Report.

Harauz, J & Kaufman, LM 2009, 'A new era of presidential security: The president and his BlackBerry', *IEEE Security & Privacy*, vol. 7, no. 2, pp. 67-70.

Harvey, P 2014, *Exiftool,* viewed 16 August 2014, <http://owl.phy.queensu.ca/~phil/exiftool/>.

Hasebe, M 1999, *System for remotely securing/locking a stolen wireless device via an email message*, US005987609A, USA.

Hattem, J 2014, *'Crypto wars' return to Congress,* The Hill, viewed 23 October 2014, <http://thehill.com/policy/cybersecurity/221147-crypto-wars-return-to-congress>.

Hildenbrand, J 2012, *How-to unlock the nexus 4 bootloader,* Android Central, viewed 20 April 2015, <http://www.androidcentral.com/how-unlock-nexus-4-bootloader>.

Hofmann, M & Fakhoury, H 2012, *Appeals court upholds constitutional right against forced decryption*, Electronic Frontier Foundation, viewed 27 May 2015, <https://www.eff.org/press/releases/appeals-court-upholds-constitutional-right-against-forced-decryption>.

Honan, M 2013, *Break out a hammer: You'll never believe the data 'wiped' smartphones store*, Wired, viewed 19 April 2015, <http://www.wired.com/2013/04/smartphone-data-trail/>.

Honan, M 2012a, *How Apple and Amazon security flaws led to my epic hacking*, Wired, viewed 5 September 2014, <http://www.wired.com/2012/08/apple-amazon-mat-honan-hacking/all/>.

Honan, M 2012b, *Yes, I was hacked. hard.* viewed 5 September 2014, <http://www.emptyage.com/post/28679875595/yes-i-was-hacked-hard>.

Hoog, A 2011a, 'Chapter 2: Android hardware platforms', in *Android forensics: Investigation, analysis and mobile security for Google Android*, Syngress, Boston, pp. 41-63.

Hoog, A 2011b, 'Chapter 4: Android file systems and data structures', in *Android forensics: Investigation, analysis and mobile security for Google Android,* Syngress, Boston, pp. 105-157.

Hoog, A 2011c, 'Chapter 6 - Android forensic techniques', in *Android forensics: Investigation, analysis and mobile security for Google Android,* Syngress, Boston, pp. 195-284.

Hoog, A 2011d, *Android forensics: Investigation, analysis and mobile security for Google Android,* Syngress, Boston.

Hörz, M 2009, *HxD - freeware hex editor and disk editor,* viewed 13 August 2014, <http://mh-nexus.de/en/hxd/>.

Huang, P, Zhou, K & Wu, C 2011, 'ShiftFlash: Make flash-based storage more resilient and robust', *Performance Evaluatio*n, vol. 68, no. 11, pp. 1193-1206.

Hubbard, J, Weimer, K & Yu Chen 2014, 'A study of SSL proxy attacks on android and iOS mobile applications', in *11th consumer communications and networking conferenc*e, IEEE, pp. 86-91.

Hughes, GF & Coughlin, TM 2006, *Tutorial on disk drive data sanitization,* Center for Magnetic Recording Research (CMRR), University of California, San Diego.

Hughes, GF & Coughlin, TM 2002, 'Secure erase of disk drive data', *IDEMA Insight Magazine,* p. 25.

Hurlbut, D 2005, *Thumbs DB files forensic issues,* AccessData, viewed 28 August 2014, <http://repo.zenk-security.com/Techniques%20d.attaques%20%20.%20%20Failles/THUMBS%20DB%20FILES%20FORENSIC%20ISSUES.pdf>.

IDC 2014, *Worldwide smartphone shipments top one billion units for the first time, according to IDC,* Press Release, viewed 22nd March 2014, <http://www.idc.com/getdoc.jsp?containerId=prUS24645514>.

Intel 2014, *RAID 0, 1, 5, 10, matrix RAID, RAID-ready,* Intel® Rapid Storage Technology (Intel® RST), viewed 23 August 2014, <http://www.intel.com/support/chipsets/imsm/sb/CS-009337.htm>.

Intel 2013, *Intel® solid-state drive pro 1500 series (M.2),* Product Specification, viewed 2 August 2014, <http://www.intel.com/content/dam/www/public/us/en/documents/product-specifications/ssd-pro-1500-series-m2-specification.pdf>.

Intel 2012, *Data security features in the intel solid - state drive 520 series,* Intel SSD 520 Series Technology Brief, viewed 30 July 2014, <http://www.intel.com/content/dam/www/public/us/en/documents/technology-briefs/ssd-520-aes-tech-brief.pdf>.

Intel 2006, *Moore's law and Intel innovation,* viewed 5 June 2014, <http://www.intel.com/content/www/us/en/history/museum-gordon-moore-law.html>.

Intel 1998, *Understanding the flash translation layer (FTL) specification,* viewed 22 October 2014, <http://www.jbosn.com/download_documents/FTL_INTEL.pdf>.

Jangir, ML 2012, *Working with MTD devices,* Open Source For U, viewed 25 May 2015, <http://www.opensourceforu.com/2012/01/working-with-mtd-devices/>.

Jevans, D, Ficcaglia, R, Spencer, G & Ryan, S 2007, *Memory data shredder,* US20070300031A1, USA.

Joe, I & Lee, Y 2011, 'Design of remote control system for data protection and backup in mobile devices', in *4th international conference on interaction science*s, IEEE, pp. 189-193.

Jones, MT 2006, *Inside the linux boot process,* IBM, viewed 17 May 2015, <http://www.ibm.com/developerworks/library/l-linuxboot/index.html>.

Joukov, N, Papaxenopoulos, H & Zadok, E 2006, 'Secure deletion myths, issues, and solutions', in *Proceedings of the second ACM workshop on storage security and survivabilit*y, ACM, pp. 61-66.

Kang, S, Park, K & Kim, J 2013, 'Cost effective data wiping methods for mobile phone', *Multimedia Tools and Applications,* vol. 71, no. 2, pp. 643-655.

Karresand, M & Shahmehri, N 2008, 'Reassembly of fragmented JPEG images containing restart markers', in *European conference on computer network defens*e, IEEE, pp. 25-32.

Karresand, M 2008, *Completing the picture: Fragments and back again,* PhD's thesis, Linköping University, Linköping, Sweden.

Kee, E & Farid, H 2010, 'Digital image authentication from thumbnails', in *Proceedings of SPIE: Media forensics and security II,* SPIE Press, pp. 75410E-1-75410E-10.

Kenney, T 2005, *Systems and methods that provide user and/or network personal data disabling commands for mobile devices,* US20050186954A1, USA.

Kim, H, Kim, J, Choi, S, Jung, H & Jung, J 2007a, 'A Page Padding Method for Fragmented Flash Storage', in *Computational science and its applications,* Springer Berlin Heidelberg, pp. 164-177.

Kim, J 2012, *f2fs: Introduce flash-friendly file system,* viewed 20 April 2015, <https://lwn.net/Articles/518718/>.

Kim, K, Hong, D, Chung, K & Ryou, J 2007b, 'Data acquisition from cell phone using logical approach', in *Proceedings of world academy of science: Engineering & technolog*y, pp. 29-32.

Kingsley-Hughes, A 2014, *Apple patches 'find my iPhone' exploit,* ZDNet, viewed 2 November 2014, <http://www.zdnet.com/apple-patches-find-my-iphone-exploit-7000033171/>.

Kingston 2013, *Kingston introduces optional TCG opal 1.0 compliant SSD,* 2013 Flash Press Release, viewed 2 August 2014, <http://www.kingston.com/us/company/press/article/6979>.

Kissel, R, Scholl, M, Skolochenko, S & Li, X 2012, *Guidelines for media sanitization,* NIST Special Publication 800-88, viewed 4 June 2014, <http://csrc.nist.gov/publications/nistpubs/800-88/NISTSP800-88_with-errata.pdf>.

Koren, R, Leibinger, E, Wiesz, N, Zilberman, E, Tzur, O, Aharonoff, S & Teicher, M 2006, *Methods of sanitizing a flash-based data storage device,* US7089350B2, USA.

Kuppusamy, KS, Senthilraja, R & Aghila, G 2012, 'A model for remote access and protection of smartphones using short message service', *International Journal of Computer Science, Engineering and Information Technology*, vol. 2, no. 1, pp. 91-100.

Kutcher, E 2014a, *Thumbcache viewer,* Google Project Hosting, viewed 16 August 2014, <https://code.google.com/p/thumbcache-viewer/>.

Kutcher, E 2014b, *Thumbs viewer,* Google Project Hosting, viewed 16 August 2014, <https://code.google.com/p/thumbs-viewer/>.

Larabel, M 2014, *The performance impact of Linux disk encryption on Ubuntu 14.04 LTS,* Phoronix, viewed 19 August 2014, <http://www.phoronix.com/scan.php?page=article&item=ubuntu_1404_encryption>.

Larabel, M 2013, *Linux 3.8 kernel officially released,* Phoronix, viewed 20 April 2015, <http://www.phoronix.com/scan.php?page=news_item&px=MTMwNTQ>.

Larabel, M 2008, *Linux 2.6.28 kernel released,* Phoronix, viewed 20 April 2015, <http://www.phoronix.com/scan.php?page=news_item&px=Njk1Nw>.

Lee, B, Son, K, Won, D & Kim, S 2011, 'Secure data deletion for USB flash memory.', *J Inf Sci Eng*, vol. 27, no. 3, pp. 933-952.

Lee, C, Sim, D, Hwang, J & Cho, S 2015, 'F2FS: A new file system for flash storage', in *13th USENIX conference on file and storage technologies,* USENIX Association, pp. 273-286.

Lee, J, Yi, S, Heo, J, Park, H, Shin, SY & Cho, Y 2010a, 'An efficient secure deletion scheme for flash file systems.', *Journal of Information Science and Engineering*, vol. 26, no. 1, pp. 27-38.

Lee, S & Kim, J 2011, *Understanding SSDs with the OpenSSD platform,* Sungkyunkwan University, Seoul.

Lee, S, Fleming, K, Park, J, Ha, K, Caulfield, A, Swanson, S, Arvind, & Kim, J 2010b, 'BlueSSD: An open platform for cross-layer experiments for NAND flash-based SSDs', in *5th annual workshop on architectural research prototypin*g.

Lessard, J & Kessler, G 2010, 'Android forensics: Simplifying cell phone examinations', *Small Scale Digital Device Forensics Journa*l, vol. 4, no. 1.

Li, T, Zhou, X, Xing, L, Lee, Y, Naveed, M, Wang, X & Han, X 2014a, 'Mayhem in the push clouds: Understanding and mitigating security hazards in mobile push-messaging services', in *21st ACM conference on computer and communications securit*y, ACM.

Li, T, Zhou, X, Xing, L, Lee, Y, Naveed, M, Wang, X & Han, X 2014b, *Supplement materials for mayhem in the push clouds paper,* viewed 6 November 2014, <https://sites.google.com/site/cloudmsging/>.

Lim, JH, Song, CW, Chung, KY, Rim, KW & Lee, JH 2013, 'Forensic Evidence Collection Procedures of Smartphone in Crime Scene', in *IT convergence and security 2012,* Springer, Dordrecht, Netherlands, pp. 35-41.

Linnell, TE 2012, *Securely erasing flash-based memory,* US8130554, USA.

Lynn, G & Davey, E 2014, *'Black market' for stolen smartphones exposed,* BBC, viewed 6 June 2014, <http://www.bbc.com/news/uk-england-london-26979061>.

machn1k *Scalpel-2.0,* viewed 15 August 2014, <https://github.com/machn1k/Scalpel-2.0>.

Ma, D, Feng, J & Li, G 2014, 'A survey of address translation technologies for flash memories', *ACM Computing Surveys*, vol. 46, no. 3, pp. 1-39.

Malchev, I 2014, *Enable hardware crypto for userdata encryption,* Git at Google, viewed 18 May 2015, <https://android.googlesource.com/platform/system/vold/+/bb7d9afea9479eabbc98133d3d968225a1e1019e%5E%21/#F0>.

ManageEngine n.d., *MDM architecture,* viewed 18 September 2014, <http://www.manageengine.com/products/desktop-central/mobile-device-management-mdm-architecture.html>.

Markoff, J 2007, *Intel says chips will run faster, using less power,* The New York Times, viewed 5 June 2014, <http://www.nytimes.com/2007/01/27/technology/27chip.html?_r=0&ei=5087&em=&en=59a4d10473c4a8c8&ex=1170046800&pagewanted=print>.

Matt 2012, *Analyzing thumbcache,* ESCForensics, viewed 28 August 2014, <http://escforensics.blogspot.com/2012/11/analyzing-thumbcache.html>.

Mayers, S & Lee, M 2011, 'From MobileMe to iCloud', in *Learn OS X Lion,* Apress, New York, pp. 245-253.

McColgan, J 2014, *Tens of thousands of americans sell themselves online every day,* AVAST Software, viewed 10 July 2014, <https://blog.avast.com/2014/07/08/tens-of-thousands-of-americans-sell-themselves-online-every-day/>.

McKemmish, R 1999, 'What is forensic computing?', in *Trends and issues in crime and criminal justic*e, Australian Institute of Criminology, pp. 1-6.

Microsoft 2014, *Windows Phone 8.1 security overview,* viewed 5 September 2014, <http://www.microsoft.com/en-us/download/details.aspx?id=42509&751be11f-ede8-5a0c-058c-2ee190a24fa6=True>.

Microsoft 2013, *Exchange ActiveSync,* viewed 12 November 2014, <http://technet.microsoft.com/en-US/library/aa998357%28v=exchg.150%29.aspx>.

Microsoft 2012, *Encrypted hard drive,* TechNet Library, viewed 19 August 2014, <http://technet.microsoft.com/en-us/library/hh831627.aspx>.

Microsoft 2010, *Device encryption,* Security for Windows Mobile Devices, viewed 20 August 2014, <http://msdn.microsoft.com/en-us/library/bb964600.aspx>.

Microsoft 2009, *Deploying Windows Mobile 5.0 with windows small business server 2003,* TechNet Library, viewed 27 July 2014, <http://technet.microsoft.com/en-us/library/cc747512(v=WS.10).aspx>.

Microsoft 2005, *Microsoft releases Windows Mobile 5.0,* viewed 27 July 2014, <http://www.microsoft.com/en-us/news/press/2005/may05/05-10windowsmobile5pr.aspx>.

Mislan, RP, Casey, E & Kessler, GC 2010, 'The growing need for on-scene triage of mobile devices', *Digital Investigation*, vol. 6, no. 3-4, pp. 112-124.

Memon, N, Pal, A & Shanmugasundaram, K 2011, *Reassembling fragmented files or documents in a file order-independent manner,* US7941464B2, USA.

Mohamad, KM 2011, 'File carving for contiguous and linearly-fragmented jpeg images and thumbnails', PhD thesis, Universiti Tun Hussein Onn, Batu Pahat, Malaysia.

Mohamad, KM & Deris, MM 2009a, 'Single-byte-marker for detecting JPEG JFIF header using FORIMAGE-JPEG', in *Fifth international joint conference on INC, IMS and ID*C, IEEE, pp. 1693-1698.

Mohamad, KM & Deris, M 2009b, 'Fragmentation Point Detection of JPEG Images at DHT Using Validator', in *Future generation information technology,* Springer, Berlin, pp. 173-180.

Mohamad, KM, Herawan, T & Deris, M 2010, 'Dual-Byte-Marker Algorithm for Detecting JFIF Header', in *Information security and assurance,* Springer, Berlin, pp. 17-26.

Mohamad, KM, Patel, A & Deris, MM 2011, 'Carving JPEG images and thumbnails using image pattern matching', in *Symposium on computers informatic*s, IEEE, pp. 78-83.

Motorola n.d., *Unlock your bootloader,* viewed 15 February 2015, <https://motorola-global-portal.custhelp.com/app/standalone/bootloader/unlock-your-device-a>.

Morris, SLA 2013, 'An investigation into the identification, reconstruction, and evidential value of thumbnail cache file fragments in unallocated space', PhD's thesis, Cranfield University, Shrivenham, Oxfordshire, UK.

Morris, S & Chivers, H 2011, 'An analysis of the structure and behaviour of the windows 7 operating system thumbnail cache', in *Proceedings from 1st cyberforensics conference*, University of Strathclyde, Glasgow, UK.

Morrison, G 2005, *Implementation guide for email protective markings for Australian government agencies,* Australian Government Information Management Office, Department of Finance and Administration, Australia, viewed 8 June 2014,

Müller, T & Spreitzenbarth, M 2013, 'FROST: Forensic Recovery of Scrambled Telephones', in *Applied cryptography and network security,* Springer Berlin Heidelberg, pp. 373-388.

Müller, T, Latzo, T & Freiling, FC 2012, 'Self-encrypting disks pose self-decrypting risks', in *29th chaos communication congres*s, Chaos Computer Club, pp. 1-10.

Munro, K 2008, 'Ghost in the machine', *Itno*w, vol. 50, no. 2, pp. 10-10.

Nickinson, P 2010, *How to unlock the nexus S bootloader,* Android Central, viewed 17 September 2014, <http://www.androidcentral.com/how-unlock-nexus-s-bootloader>.

Ockenden, W & Sveen, B 2015, *Abdilo, infamous Australian teen hacker, raided by police and ordered to surrender passwords,* ABC News, viewed 27 May 2015, <http://www.abc.net.au/news/2015-04-02/infamous-australian-teenager-hacker-abdilo-raided-by-police/6368612>.

Ogg, E 2009, *Updated: iPhone OS 3.0 now available*, CNET, viewed 15 June 2014, <http://www.cnet.com/news/updated-iphone-os-3-0-now-available/>.

Onyon, R, Stannard, L & Ridgard, L 2007, *Remote cell phone auto destruct*, US20070056043A1, USA.

OWASP 2014, *Cross-site request forgery (CSRF)*, viewed 2 November 2014, <https://www.owasp.org/index.php/Cross-Site_Request_Forgery_%28CSRF%29>.

OWASP 2013, *OWASP top 10*, viewed 5 November 2014, <https://www.owasp.org/index.php/Top10>.

Oxygen Forensics *Oxygen forensic® suite - mobile forensic software for cell phones, smartphones and other mobile devices*, viewed 10 August 2014, <http://www.oxygen-forensic.com/>.

Pagliery, J 2014, *Hackers can 'un-brick' stolen iPhones*, CNN, viewed 28 October 2014, <http://money.cnn.com/2014/05/21/technology/security/icloud-hack/>.

Pal, A, Sencar, HT & Memon, N 2008, 'Detecting file fragmentation point using sequential hypothesis testing', *Digital Investigatio*n, vol. 5, supplement issue, pp. S2-S13.

Park, J, Chung, H & Lee, S 2012, 'Forensic analysis techniques for fragmented flash memory pages in smartphones', *Digital Investigatio*n, vol. 9, no. 2, pp. 109.

Park, K, Ma, GI, Yi, JH, Cho, Y, Cho, S & Park, S 2011, 'Smartphone remote lock and wipe system with integrity checking of SMS notification', in *International conference on consumer electronic*s, IEEE, pp. 263-264.

Park, S, Kim, J, Jung, Y, Lim, D, Seo, Y, Cho, Y, Yi, S, Lee, J, Kim, S & Oh, J 2012, *Flash memory device having secure file deletion function and method for securely deleting flash file*, US8117377B2, USA.

Parsonage, H 2012, *Under my thumbs - revisiting windows thumbnail databases and some new revelations about the forensic implications*, viewed 28 August 2014, <http://computerforensics.parsonage.co.uk/downloads/UnderMyThumbs.pdf>.

Percival, C 2009, 'Stronger key derivation via sequential memory-hard functions', in *Bsdcan*.

Pieterse, H & Olivier, MS 2013, 'Security steps for smartphone users', in *Information security for South Afric*a, IEEE, pp. 1-6.

Poiesz, B 2013, *Find your lost phone with android device manager*, Android Official Blog, viewed 21 April 2014, <http://officialandroid.blogspot.com/2013/08/find-your-lost-phone-with-android.html>.

Pond, W 2004, *Netcat (windows)*, viewed 14 August 2014, <http://www.securityfocus.com/tools/139>.

Punja, SG & Mislan, RP 2008, 'Mobile device analysis', *Small Scale Digital Device Forensics Journa*l, vol. 2, no. 1, pp. 1-16.

Quick, D & Alzaabi, M 2011, 'Forensic analysis of the android file system YAFFS2', in *Proceedings of the 9th Australian digital forensics conferenc*e, Edith Cowan University, Perth, pp. 100-109.

Quick, D & Choo, KKR 2014, 'Google drive: Forensic analysis of data remnants', *Journal of Network and Computer Applications*, vol. 40, pp. 179-193.

Quick, D & Choo, KKR 2013a, 'Digital droplets: Microsoft SkyDrive forensic data remnants', *Future Generation Computer Systems*, vol. 29, no. 6, pp. 1378-1394.

Quick, D & Choo, KR 2013b, 'Dropbox analysis: Data remnants on user machines', *Digital Investigation*, vol. 10, no. 1, pp. 3-18.

Quick, D, Tassone, C & Choo, KKR 2014, 'Forensic analysis of windows thumbcache files', in *20th Americas conference on information system*s, Association for Information Systems.

Reardon, J, Basin, D & Capkun, S 2013a, 'SoK: Secure data deletion', in *Symposium on security and privac*y, IEEE, pp. 301-315.

Reardon, J, Capkun, S & Basin, D 2013b, 'Data node encrypted file system: Efficient secure deletion for flash memory', in *Proceedings of the 21st USENIX security symposiu*m, USENIX, pp. 333-348.

Reardon, J, Marforio, C, Capkun, S & Basin, D 2012, 'User-level secure deletion on log-structured file systems', in *Proceedings of the 7th ACM symposium on information, computer and communications securit*y, ACM, pp. 63-73.

Reddy, VK & Rao, JE 2014, 'A survey on security in cloud using homographic and disk encryption methods', *International Journal of Computer Science Engineering and Technology*, vol. 2, no. 4, pp. 107-112.

Reuters 2014, *PC shipments continue global decline as mobile wins out -IDC,* viewed 4 September 2014, <http://www.reuters.com/article/2014/01/10/technology-pcs-idUSL3N0KK2O720140110>.

Rich, D 2007, 'Authentication in transient storage device attachments', *Computing*, vol. 40, no. 4, pp. 102-104.

Richard III, GG & Roussev, V 2005, *Scalpel: A frugal, high performance file carver.* Digital Forensic Research Workshop.

RIFF box, 2014, <http://www.riffbox.org/>.

RootzWiki *RootzWiki forum,* viewed 15 August 2014, <http://rootzwiki.com/index>.

Russakovskii, A 2010, *Custom ROMs for Android explained - here is why you want them,* Android Police, viewed 18 May 2015, <http://www.androidpolice.com/2010/05/01/custom-roms-for-android-explained-and-why-you-want-them/>.

Rutkowska, J 2011, *Anti evil maid,* The Invisible Things Lab's blog, viewed 2 August 2014, <http://theinvisiblethings.blogspot.com/2011/09/anti-evil-maid.html>.

Rutkowska, J 2009, *Evil maid goes after TrueCrypt,* The Invisible Things Lab's blog, viewed 1 August 2014, <http://theinvisiblethings.blogspot.com/2009/10/evil-maid-goes-after-truecrypt.html>.

Sajja, A 2010, 'Forensic reconstruction of fragmented variable bitrate mp3 files', Master's thesis, University of New Orleans, USA.

Sang, W 2012, *[RFC 04/10] devfs & mtd: Add MEMERASE ioctl support*, viewed 25 May 2015, <http://lists.infradead.org/pipermail/barebox/2012-October/010713.html>.

Saxena, M, Zhang, Y, Swift, MM, Arpaci-Dusseau, AC & Arpaci-Dusseau, RH 2013, 'Getting real: Lessons in transitioning research simulations into hardware systems', in *Proceedings of the 11th USENIX conference on file and storage technologies*, USENIX, pp. 215-228.

Schwamm, R 2014, 'Effectiveness of the factory reset on a mobile device', Master's thesis, Naval Postgraduate School, Monterey, California, USA.

Schwamm, R & Rowe, NC 2014, 'Effects of the factory reset on mobile devices', *Journal of Digital Forensics, Security and Law*, vol. 9, no. 2, pp. 205-220.

Sencar, HT & Memon, N 2009, 'Identification and recovery of JPEG files with missing fragments', *Digital Investigation*, vol. 6, Supplement issue, pp. S88-S98.

Sennett, DWA & Daly, BK 2013, *Remote disablement of a communication device*, US008375422B2, USA.

Shin, I 2012, 'Secure file delete in NAND-based storage', *International Journal of Security and its Applications*, vol. 6, no. 2, pp. 257-260.

Shin, Y 2005, 'Non-volatile memory technologies for beyond 2010', in *Symposium on VLSI circuits digest of technical papers*, IEEE, pp. 156-159.

Siciliano, R 2012, *I found your data on that used device you sold*, McAfee, viewed 19 April 2015, <http://blogs.mcafee.com/consumer/i-found-your-data-on-that-used-device-you-sold>.

Simão, André Morum de Lima, Sícoli, FC, Melo, LPd, Deus, Flávio Elias Gomes de & Sousa Júnior, Rafael Timóteo de 2011, 'Acquisition and analysis of digital evidence in android smartphone', *The International Journal of Forensic Computer Science*, vol. 6, no. 1, pp. 28-43.

Simon, L & Anderson, R 2015a, *Security analysis of android factory resets*, 4th Mobile Security Technologies Workshop.

Simon, L & Anderson, R 2015b, *Security Analysis of Consumer-Grade Anti-Theft Solutions Provided by Android Mobile Anti-Virus Apps*, 4th Mobile Security Technologies Workshop.

Skillen, A & Mannan, M 2013, 'On implementing deniable storage encryption for mobile devices', in *20th annual network & distributed system security symposium*, Internet Society, pp. 1-17.

Son, N, Lee, Y, Kim, D, James, JI, Lee, S & Lee, K 2013, 'A study of user data integrity during acquisition of android devices', *Digital Investigatio*n, vol. 10, pp. S3-S11.

Sophos 2014, *SafeGuard device encryption: OPAL support*, Sophos Knowledgebase Support, viewed 30 July 2014, <http://www.sophos.com/en-us/support/knowledgebase/113366.aspx>.

Spreitzenbarth, M & Holz, T 2010, 'Towards secure deletion on smartphones.', in *5th conference of the GI special interest group "Sicherheit, schutz und zuverlässigkeit"*, Gesellschaft für Informatik e.V. (GI), pp. 165-176.

Steele, RK, Key, DS, Abbasi, MA & Lutz, BC 2009, *Method and apparatus for sanitizing or modifying flash memory chip data*, US20090113113, USA.

Stericson, S 2015, *BusyBox*, viewed 10 January 2015, <https://play.google.com/store/apps/details?id=stericson.busybox>.

Storer, MW, Greenan, K & Miller, EL 2006, 'Long-term threats to secure archives', in *Proceedings of the second ACM workshop on storage security and survivability*, ACM, pp. 9-16.

Subha, S 2009, 'An algorithm for secure deletion in flash memories', in *2nd IEEE international conference on computer science and information technology*, IEEE, pp. 260-262.

Sun, K, Choi, J, Lee, D & Noh, SH 2008, 'Models and design of an adaptive hybrid scheme for secure deletion of data in consumer electronics', *IEEE Transactions on Consumer Electronics*, vol. 54, no. 1, pp. 100-104.

Swanson, S & Wei, M 2010, *Safe: Fast, verifiable sanitization for SSDs*, University of California, San Diego.

Sylve, J, Case, A, Marziale, L & Richard, GG 2012, 'Acquisition and analysis of volatile memory from android devices', *Digital Investigation*, vol. 8, no. 3-4, pp. 175-184.

Symantec 2012, *Symantec smartphone honey stick project*, Press Kits, viewed 2 June 2014, <https://www.symantec.com/about/news/resources/press_kits/detail.jsp?pkid=symantec-smartphone-honey-stick-project>.

Tachibanaya, T 1999, *Description of exif file format*, Personal Information Architecture, MIT Media Laboratory, viewed 17 August 2014, <http://www.media.mit.edu/pia/Research/deepview/exif.html>.

TCG n.d., *Data production solution*, Trusted Computing Group, viewed 2 August 2014, <www.trustedcomputinggroup.org/solutions/data_protectiona>.

Tereshkin, A 2010, 'Evil maid goes after PGP whole disk encryption', in *Proceedings of the 3rd International conference on security of information and networks*, ACM, pp. 2-2.

Teufl, P, Zefferer, T & Stromberger, C 2013, 'Mobile Device Encryption Systems', in *Security and privacy protection in information processing systems*, Springer Berlin Heidelberg, pp. 203-216.

The Economist 2009, *The boom in smart-phones: Cleverly simple*, viewed 5 June 2014, <http://www.economist.com/node/14563636>.

The Guardian 2013, *Recycled mobile phones retain previous owner data*, viewed 19 April 2015, <http://web.archive.org/web/20140529182505/http://www.theguardian.com/media-network/partner-zone-infosecurity/mobile-phones-previous-owner-data [archived]>.

Timberg, C 2014, *Newest androids will join iPhones in offering default encryption, blocking police*, The Washington Post, viewed 19 September 2014, <http://www.washingtonpost.com/blogs/the-switch/wp/2014/09/18/newest-androids-will-join-iphones-in-offering-default-encryption-blocking-police/>.

Trusted Computing Group 2012, *TCG storage security subsystem class (SSC): Opal, specification version 2.00, revision 1.0 edition*, viewed 25 July 2014, <http://www.trustedcomputinggroup.org/resources/storage_work_group_storage_security_subsystem_class_opal>.

Tsai, YC & Yang, CH 2013, 'Physical Forensic Acquisition and Pattern Unlock on Android Smart Phones', in *Future information communication technology and applications*, Springer, Dordrecht, Netherlands, pp. 871-881.

TWRP 2015, *TeamWin recovery project,* viewed 13 January 2015, <http://teamw.in/project/twrp2/>.

US-CERT 2014, *Cve-2014-8346,* National Vulnerability Database, viewed 28 October 2014, <http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-8346>.

Vantage Technologies n.d., *RAID 5 data recovery FAQ,* viewed 16 July 2014, <http://www.vantagetech.com/faq/raid-5-recovery-faq.html>.

Verizon 2014, *2014 data breach investigation report,* viewed 8 July 2014, <http://www.verizonenterprise.com/DBIR/2014/>.

Vidas, T, Zhang, C & Christin, N 2011, 'Toward a general collection methodology for android devices', *Digital Investigatio*n, vol. 8, pp. S14-S24.

Wakefield, J 2014, *Devices being remotely wiped in police custody,* BBC, viewed 13 October 2014, <www.bbc.co.uk/news/technology-29464889a>.

Walker, DR & Fyke, SH 2013, *System and method for remote wipe through voice mail,* EP2575384A1, EU.

Wartickler 2012, *[GUIDE] internal memory data recovery - yes we can!*, XDA Forums, viewed 3 December 2014, <http://www.xda-developers.com/android/restore-galaxy-nexus-internal-memory-after-bootloader-unlock-wipe/>.

Wei, MYC, Grupp, LM, Spada, FE & Swanson, S 2011, 'Reliably erasing data from flash-based solid state drives', in *9th USENIX conference on file and storage technologies,* USENIX, pp. 8-20.

Weng, WK & Wu, HH 2012, *Secure erase system for a solid state non-volatile memory device*, US20120079289A1, USA.

Wetzels, J 2014, *Hidden in snow, revealed in thaw: Cold boot attacks revisited,* Seminar Information Security Technology, Kerckhoffs Institute, Netherlands.

Woodhouse, D 2008, *General MTD documentation,* Memory Technology Device (MTD) Subsystem for Linux, viewed 6 August 2014, <http://www.linux-mtd.infradead.org/doc/general.html>.

Wright, C, Kleiman, D & Sundhar, S 2008, 'Overwriting hard drive data: The great wiping controversy', in *Information systems security,* Springer Berlin Heidelberg, pp. 243-257.

XDA Developers n.d.a, *Nexus S Android development,* viewed 11 August 2014, <http://forum.xda-developers.com/nexus-s/development>.

XDA Developers n.d.b, *XDA forum,* viewed 15 August 2014, <http://forum.xda-developers.com/>.

Xu, M & Dong, S 2009, 'Reassembling the fragmented JPEG images based on sequential pixel prediction', in *International symposium on Computer network and multimedia technolog*y, IEEE.

Yu, X, Wang, Z, Sun, K, Zhu, WT, Gao, N & Jing, J 2014, 'Remotely wiping sensitive data on stolen smartphones', in *Proceedings of the 9th ACM symposium on information, computer and communications securit*y, ACM, pp. 537-542.

Zetter, K 2008, *Palin E-mail hacker says it was easy*, Wired, viewed 5 September 2014, <http://www.wired.com/2008/09/palin-e-mail-ha/>.

Zhang, H 2012, *Make mobile more manageable*, Official Google for Work Blog, viewed 20 April 2015, <http://googleforwork.blogspot.com/2012/08/make-mobile-more-manageable.html>.

Zhao, X, Xu, C, Chi, Z, Yan, H, Feng, DD & Chen, G 2007, 'Image recovery from broken image streams', in *International conference on image processing*, IEEE, pp. 533-536.

Zimmermann, C, Spreitzenbarth, M, Schmitt, S & Freiling, FC 2012, 'Forensic analysis of YAFFS2', in *6th conference of the GI special interest group "Sicherheit, schutz und zuverlässigkeit"*, Gesellschaft für Informatik e.V. (GI), pp. 59-69.

7 Appendix

Step	**Command**
1. Unlock bootloader. Restore the mobile devices to factory defaults by flashing factory image.	
2. Prepare baseline data according to protocol detailed in Table x	
3. Reboot into fastbooot mode and install custom recovery, Team Win Recovery Project (TWRP) for physical acquisition tools.	[fastboot mode] fastboot flash recovery twrp.img
4. Acquire physical forensic image Moto G	[recovery mode] Unmount user data partitions C:\> adb shell ~ # mount rootfs on / type rootfs (rw) tmpfs on /dev type tmpfs (rw,seclabel,nosuid,relatime,size=443852k,nr_inodes=110963,mode=755) devpts on /dev/pts type devpts (rw,seclabel,relatime,mode=600) proc on /proc type proc (rw,relatime) sysfs on /sys type sysfs (rw,seclabel,relatime) selinuxfs on /sys/fs/selinux type selinuxfs (rw,relatime) tmpfs on /tmp type tmpfs (rw,seclabel,relatime,size=443852k,nr_inodes=110963) **/dev/block/mmcblk0p36 on /data** type f2fs (rw,relatime,background_gc=on,user_xattr,inline_xattr,acl,errors=continue,active_logs=6) **/dev/block/mmcblk0p36 on /sdcard** type f2fs (rw,relatime,background_gc=on,user_xattr,inline_xattr,acl,errors=continue,active_logs=6) **/dev/block/mmcblk0p33 on /cache** type ext4 (rw,seclabel,relatime,data=ordered) ~ # umount /dev/block/mmcblk0p36 ~ # umount /dev/block/mmcblk0p33

	Identify partition path. C:\> adb shell ~ # ls -l /dev/block/platform/msm_sdcc.1/by-name lrwxrwxrwx root root cache -> /dev/block/mmcblk0p33 lrwxrwxrwx root root userdata -> /dev/block/mmcblk0p36 Acquire forensic image. C:\> adb forward tcp:5555 tcp:5555 C:\> adb shell ~ # nc -l -p 5555 -e dd if=/dev/block/mmcblk0p33 Launch a new command prompt to receive physical image. C:\> adb forward tcp:5555 tcp:5555 C:\> nc 127.0.0.1 5555 > moto-baseline-cache.raw Repeat for userdata partition.
Nexus S	Unmount user data partitions. C:\> adb shell ~ # mount rootfs on / type rootfs (rw,seclabel) tmpfs on /dev type tmpfs (rw,seclabel,nosuid,relatime,mode=755) devpts on /dev/pts type devpts (rw,seclabel,relatime,mode=600) proc on /proc type proc (rw,relatime) sysfs on /sys type sysfs (rw,seclabel,relatime) selinuxfs on /sys/fs/selinux type selinuxfs (rw,relatime) tmpfs on /tmp type tmpfs (rw,seclabel,relatime) **/dev/block/mtdblock4 on /cache** type yaffs2 (rw,seclabel,nodev,noatime,nodiratime) **/dev/block/mmcblk0p3 on /sdcard** type vfat (rw,relatime,fmask=0000,dmask=0000,allow_utime=0022,codepage=cp437,iocharset=iso8859-1,shortname=mixed,errors=remount-ro) /dev/block/mmcblk0p3 on /and-sec type vfat (rw,relatime,fmask=0000,dmask=0000,allow_utime=0022,codepage=cp437,iocharset=iso8859-1,shortname=mixed,errors=remount-ro) ~ # umount /dev/block/mtdblock4 ~ # umount /dev/block/mmcblk0p3 Identify partition path. C:\> adb shell ~ # ls -l /dev/block/platform/s3c-sdhci.0/by-name

	lrwxrwxrwx root root media -> /dev/block/mmcblk0p3 lrwxrwxrwx root root userdata -> /dev/block/mmcblk0p2 Acquire forensic image . C:\> adb forward tcp:5555 tcp:5555 C:\> adb shell ~ # nc -l -p 5555 -e dd if=/dev/block/mmcblk0p3 Launch a new command prompt to receive physical image. C:\> adb forward tcp:5555 tcp:5555 C:\> nc 127.0.0.1 5555 > nexus-baseline-media.raw Repeat for userdata partition.
Nexus 4	Unmount user data partitions. C:\> adb shell ~ # mount rootfs on / type rootfs (rw) tmpfs on /dev type tmpfs (rw,seclabel,nosuid,relatime,mode=755) devpts on /dev/pts type devpts (rw,seclabel,relatime,mode=600) proc on /proc type proc (rw,relatime) sysfs on /sys type sysfs (rw,seclabel,relatime) selinuxfs on /sys/fs/selinux type selinuxfs (rw,relatime) **/dev/block/mmcblk0p23 on /data** type ext4 (rw,seclabel,relatime,data=ordered) **/dev/block/mmcblk0p23 on /sdcard** type ext4 (rw,seclabel,relatime,data=ordered) **/dev/block/mmcblk0p22 on /cache** type ext4 (rw,seclabel,relatime,data=ordered) ~ # umount /dev/block/mmcblk0p23 ~ # umount /dev/block/mmcblk0p22 Identify partition path. C:\> adb shell ~ # ls -l /dev/block/platform/msm_sdcc.1/by-name lrwxrwxrwx root root cache -> /dev/block/mmcblk0p22 lrwxrwxrwx root root userdata -> /dev/block/mmcblk0p23 Acquire forensic image . C:\> adb forward tcp:5555 tcp:5555 C:\> adb shell ~ # nc -l -p 5555 -e dd if=/dev/block/mmcblk0p22 Launch a new command prompt to receive physical image.

<table>
<tr><td></td><td>C:\> adb forward tcp:5555 tcp:5555
C:\> nc 127.0.0.1 5555 > mako-baseline-cache.raw

Repeat for userdata partition.</td></tr>
<tr><td>5. Reboot to fastboot mode and install stock recovery (extracted from factory image).</td><td></td></tr>
<tr><td>6. Install targeted app.</td><td></td></tr>
<tr><td>7. Initiate remote wipe command.</td><td></td></tr>
<tr><td>8. Install custom recovery.</td><td></td></tr>
<tr><td>9. Acquire physical image using similar command in step 4.</td><td></td></tr>
<tr><td>10. Wipe specific partition through fastboot mode.
Moto G & Nexus 4: cache and userdata
Nexus S: cache, userdata, and media</td><td>[fastboot mode]
fastboot erase [partition name]</td></tr>
<tr><td>11. Restore to baseline state.
Moto G</td><td>[recovery mode]
Restore forensic image to specific partition.
C:\> adb forward tcp:5555 tcp:5555
C:\> adb shell
~ # nc -l -p 5555 | dd of=/dev/block/mmcblk0p33

Launch a new command prompt to send physical image.
C:\> adb forward tcp:5555 tcp:5555
C:\> dd if=moto-baseline-cache.raw | pv -i 0.5 | nc 127.0.0.1 5555

Repeat for userdata partition.</td></tr>
<tr><td>Nexus S</td><td>Restore forensic image to specific partition.
C:\> adb forward tcp:5555 tcp:5555</td></tr>
</table>

<table>
<tr><td></td><td>C:\> adb shell
~ # nc -l -p 5555 | dd of=/dev/block/mmcblk0p3

Launch a new command prompt to send physical image.
C:\> adb forward tcp:5555 tcp:5555
C:\> dd if=nexus-baseline-media.raw | pv -i 0.5 | nc 127.0.0.1 5555

Repeat for userdata partition.</td></tr>
<tr><td>Nexus 4</td><td>Restore forensic image to specific partition.
C:\> adb forward tcp:5555 tcp:5555
C:\> adb shell
~ # nc -l -p 5555 | dd of=/dev/block/mmcblk0p22

Launch a new command prompt to send physical image.
C:\> adb forward tcp:5555 tcp:5555
C:\> dd if=mako-baseline-cache.raw | pv -i 0.5 | nc 127.0.0.1 5555

Repeat for userdata partition.</td></tr>
<tr><td>12. Repeat step 5-11 for each experiment.</td><td></td></tr>
</table>

Table 38: Experiment steps.

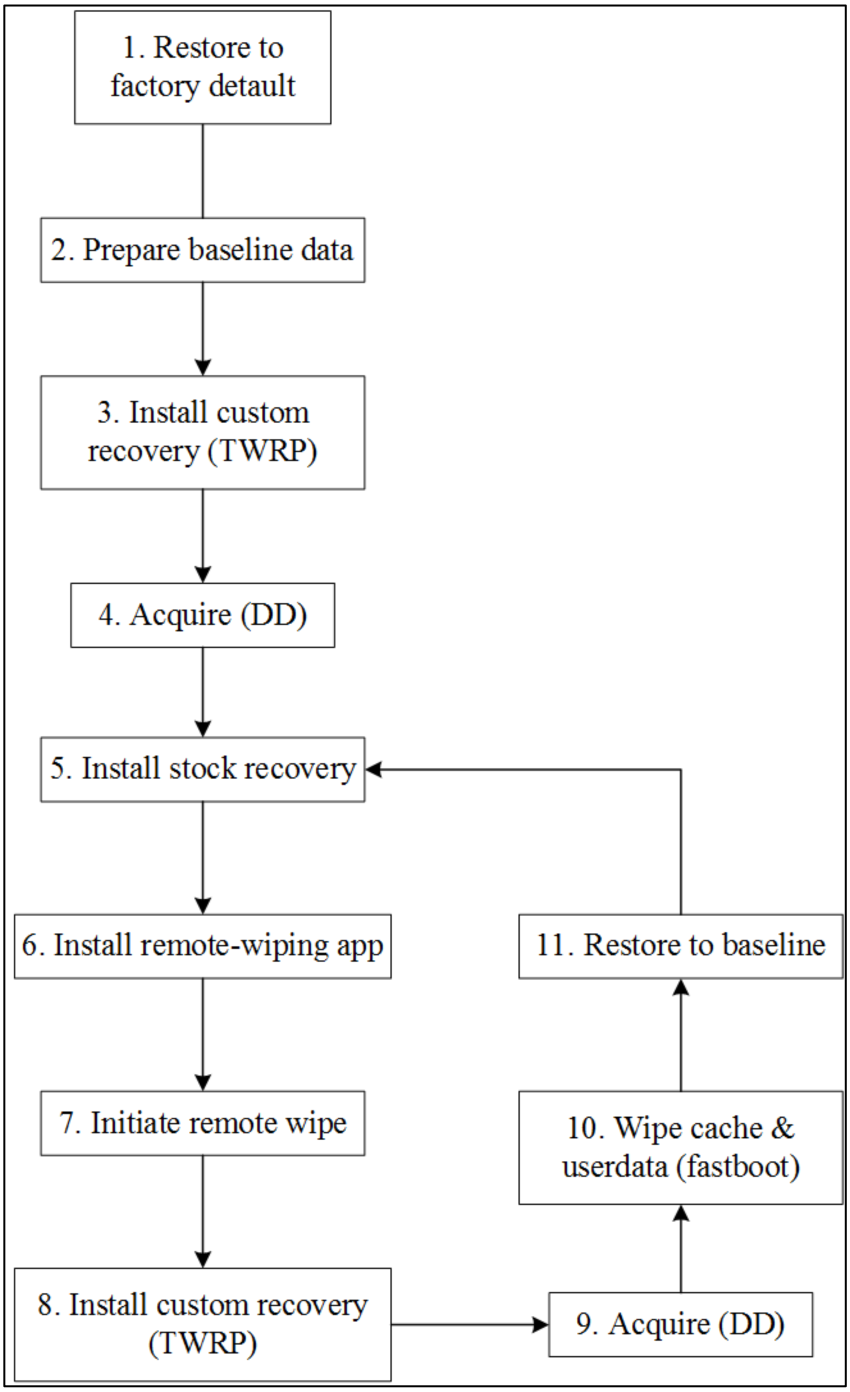

1. Restore to factory detault
2. Prepare baseline data
3. Install custom recovery (TWRP)
4. Acquire (DD)
5. Install stock recovery
6. Install remote-wiping app
7. Initiate remote wipe
8. Install custom recovery (TWRP)
9. Acquire (DD)
10. Wipe cache & userdata (fastboot)
11. Restore to baseline

Figure 13: Overview of experiments conducted.