Android Anti-Forensics Through a Local Paradigm

By

Alessandro Distefano, Gianluigi Me and Francesco Pace

From the proceedings of

The Digital Forensic Research Conference

DFRWS 2010 USA

Portland, OR (Aug 2nd - 4th)

available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/diin

Digital Investigation

Android anti-forensics through a local paradigm

Alessandro Distefano[a], *Gianluigi Me*[a,*], *Francesco Pace*[b]

[a] *Computer Science, Systems and Production Department University of Rome Tor Vergata, Via del Politecnico 1 00133, Rome, Italy*
[b] *IAC-CNR, Istituto per le Applicazioni del Calcolo, Viale Manzoni 30, Rome, Italy*

Index Terms:
Android
Anti-Forensics
Mobile Devices
Evidence

A B S T R A C T

Mobile devices are among the most disruptive technologies of the last years, gaining even more diffusion and success in the daily life of a wide range of people categories. Unfortunately, while the number of mobile devices implicated in crime activities is relevant and growing, the capability to perform the forensic analysis of such devices is limited both by technological and methodological problems. In this paper, we focus on Anti-Forensic techniques applied to mobile devices, presenting some fully automated instances of such techniques to Android devices. Furthermore, we tested the effectiveness of such techniques versus both the cursory examination of the device and some acquisition tools.

1. Introduction

Mobile devices, and in particular mobile phones, are among the most common and diffused current technologies (Kalba, 2008). The 2.6 billion of subscribers in the world and the recent trend of growth registered especially in the rural areas (e.g., China Mobile declare that they are adding 6 million of new subscribers per month) confirm how this technology is over diffused. In addition to the market penetration of such devices, another interesting item is represented by the advanced functionalities they have; these functionalities range from user interface, to computational resources, to connectivity, to application development. According to the classification of the US National Institute of Standards and Technology given in (J.W. and A.R., 2007), it is possible to identify three different classes of mobile phones: Basic, Advanced and Smart that are summarized in Table 1.

During the evolution from Basic to Smart mobile phones (often shortened as Smartphones), the amount and kind of personal data stored evolved from simple phonebooks and text messages for Basic phones, to a complex and wide set of personal information (e.g., MultiMedia and e-mail messages, browser history, social network accounts, application data) for Smartphones.

Regarding the forensic environment, the greater quantity of personal information is stored by the mobile device, the more interesting is the device itself; hence, it is easy to figure out that the class of SmartPhones is the most interesting for forensic purposes (van der Knijff, 2007). However, due to the heterogeneity and the limits in acquiring the data stored by such kind of devices, the forensic discipline that focuses on mobile devices (called Mobile Forensics) is still experiencing a number of difficulties and problems that are to overcome (Jansen et al., 2008).

In the last years, a new and potentially alarming trend was detected by the forensic investigators (B.S., 2007): the uptick in the use of Anti-Forensics (AF). Currently, such trend seems to be confined in the classic forensic environments; however, in the near future, it could become interesting for Mobile Forensics as well. The work presented in this paper focuses on the instance of some common AF techniques to Android mobile devices.

The paper is structured as follows: Section 2 provides a general definition of AF and of the common AF techniques. Section 3 provides a brief description of the Android Operating System, with particular attention to the features that can support AF techniques; while Section 4 describes the instances, to Android devices, of such techniques that have

* *Corresponding author.*
E-mail addresses: distefano@disp.uniroma2.it (A. Distefano), me@disp.uniroma2.it (G. Me), paxanto@gmail.com (F. Pace).
1742-2876/$ – see front matter © 2010 Digital Forensic Research Workshop. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.diin.2010.05.011

Table 1 – Illustrative classification of mobile phones in terms of different capabilities.

Capability	Classes of mobile phones		
	Basic	Advanced	Smart
Processor & Memory	Limited	Improved	Superior
Display	Grayscale	Color	16-bit Color or higher
Input	Keypad	Keypad	Keyboard, Touch Screen Handwriting Recognition
Cell Interface	Voice	Voice, High Speed Data	Voice, Very High Speed Data
Wireless	IrDA	IrDA, Bluetooth	IrDA, Bluetooth, WiFi

been designed, implemented and tested. Section 5 describes both the definition and the results of the experiments we have performed; the paper ends with the discussion of the future developments (Section 6) and the conclusions (Section 7).

2. State of the art

2.1. *Definition of anti-forensics*

AF is a quite young and immature discipline even more if we consider the Mobile Environment (ME); regarding ME, a number of difficulties and issues during forensics analysis are still to overcome (Jansen et al., 2008), hence the possible shapes of AF techniques are continuously and rapidly evolving (Geiger, 2005; Peron; Berghel, 2007).

Currently, there is no unique and standard definition of AF, while several definitions exist and focus on different and specific aspects. Among those, some focus on breaking forensic tools or avoiding the detection of evidence (Foster and Liu, 2005) while some others relate AF to system intrusions (S. B., 2002).

However, in accordance with (Arriving at an anti-forensics consensus, 2006), in this paper we consider AF to be *any attempts to compromise the availability or usefulness of evidence in the forensic process*. The availability of the evidence can be compromised by preventing its creation, hiding its existence and by manipulating the evidence as well; the usefulness can be compromised by deleting the evidence or by tampering its integrity.

By the comprehension and the study of the AF techniques, a number of useful conclusions and guidelines can be drawn, in order to improve and harden the currently used forensic tools and techniques.

2.2. *Kinds of anti-forensics*

The work described in this paper refers to the kinds of AF techniques which are described in Arriving at an anti-forensics consensus (2006). These techniques, which have been identified in general, are briefly summarized as follows.

2.2.1. *Destroying evidence*

It involves the destruction of evidence, in order to make it unusable during the investigative process. Although the destruction of evidence is often fatal, it is worth noticing that the tools, or the operations, used to destroy the evidence can produce evidence themselves in terms of traces of their usage.

2.2.2. *Hiding evidence*

Actions taken to subvert the analyst, rather than a specific forensic analysis application, to decrease, or even nullify, the evidence visibility during the forensics analysis. The strength of this technique is strictly connected to the limitations of the people or, if any, of the used forensics tools. As for the previous item, the presence of any hiding tools can generate evidence.

2.2.3. *Eliminating evidence sources*

It is the neutralization of the evidentiary sources; this technique does not concern the destruction but the prevention of evidence creation.

2.2.4. *Counterfeiting evidence*

It is the creation of a fake version of the evidence which is properly made to carry wrong or deviated information in order to divert the forensic process.

2.3. *Mobile anti-forensics*

The definitions provided in the Sections 2.1 and 2.2 can be applied to any kind of evidence and to any Forensic discipline as well. In this paper, we focus on Mobile Forensics with the related implications on the kind of digital evidence we are interested in.

In the recent years, Mobile Devices (MDs) became even more important in the Forensic field (Williams) due to the rich amount of personal information they store. However, the classical forensic guidelines (IOCE, 2010; ACPO, 2010) and tools, often, are not suitable for MDs as well. Probably, the main impediment to overcome is the unavailability of a direct access to the internal memory of such devices. In fact, if the removable storage volumes (e.g., memory cards, SIM cards) can be isolated from the device and analyzed in a straightforward manner, the internal memory volume cannot. In such scenario, several commercial tools and suites exist (e.g., Paraben corporation official website; Micro systemation official website) and are widely used in practice; however, as for any other commercial forensic tool, the issue of investigating the tool behavior arises (Carrier, 2003). Furthermore, due to the inaccessibility of the volume, the internal memory seems to be an ideal candidate in order to apply some AF techniques. However, as the techniques identified in Arriving at an anti-forensics consensus (2006) and briefly summarized in Section 2.2 are defined only in general, in Section 4.2, we provide an example of their instance to the ME.

3. The android operating system

Android is a set of open source software elements specifically designed for MDs developed by Google; it includes the Operating System (OS), a middleware and a set of applications. Although it has been designed and developed for MDs (e.g., Smartphones), several laptop manufacturers plan to equip their products with Android.

At the time of writing, less than 2% of Smartphones (Gartner Mobile OS Share Forecast, 2009) runs Android and Gartner Inc. forecasts a 15% market share in 2012; in such case, Android will be the second OS, behind Symbian, in terms of Smartphone's market penetration. Furthermore, if Android will be hosted on laptops, the integration of Smartphones and portable computer could be boosted with the natural side-effects on the market.

3.1. *The android architecture*

Android is composed by five major components, depicted in Fig. 1, that are briefly introduced below.

- Applications: Android is distributed with a set of typical applications for MDs (e.g., e-mail client, text messaging management, browser, contacts management) written using the Java Programming Language.
- Application Framework: Android offers the capability of Java applications development providing a rich set of services which can be exploited. Developers can consume and provide services through of a wide set of Application Programming Interfaces (APIs), with the objective of the reuse of components, always respecting the security constraints enforced by the framework.
- Libraries: Android includes a set of libraries (e.g., Standard C System Library, Media Libraries, 3D Libraries) used by the components of the system through the Android Application Framework just outlined.
- Android Runtime: the Runtime is composed by a set of Core Libraries and by the Dalvik Virtual Machine (DVM). Every running application holds its own instance of the DVM and executes in its own process.
- Linux Kernel: one of the most interesting features of Android is the underlying Linux kernel supporting the core services, such as memory and process management, network stack, drivers and security.

3.1.1. *Basics on android file system*

Another interesting element of Android is the natively supported YAFFS2 File System (FS). YAFFS (YAFFS official website, 2010) stands for Yet Another Flash FS and, at the time of writing, it is the only FS that has been specifically designed for NAND flash chips. The use of NAND flash chips in the field of embedded and mobile devices (Kinam, 2005) is increasing and replacing the common-old NOR chips because of the improved density, speed and the reduced cost. At the time of writing, YAFFS was released in two version:

- YAFFS1: designed for old NAND chips with 512 byte pages plus 16 byte spare areas;
- YAFFS2: evolved from YAFFS1 to accommodate newer chips with 2048 byte pages plus 64 bytes spare areas.

In addition to the different NAND chips supported, YAFFS2 evolved in terms of performance, reliability, efficiency and support to the "write once" requirement for modern NAND flash (Kinam, 2005; An introduction to NAND flash, 2006).

In the next section, a brief description of some items related to the Android Security Architecture is provided; a more extended description is available at Android security architecture (2010). Such description is fundamental in order to highlight what are the issues to be faced and the features to

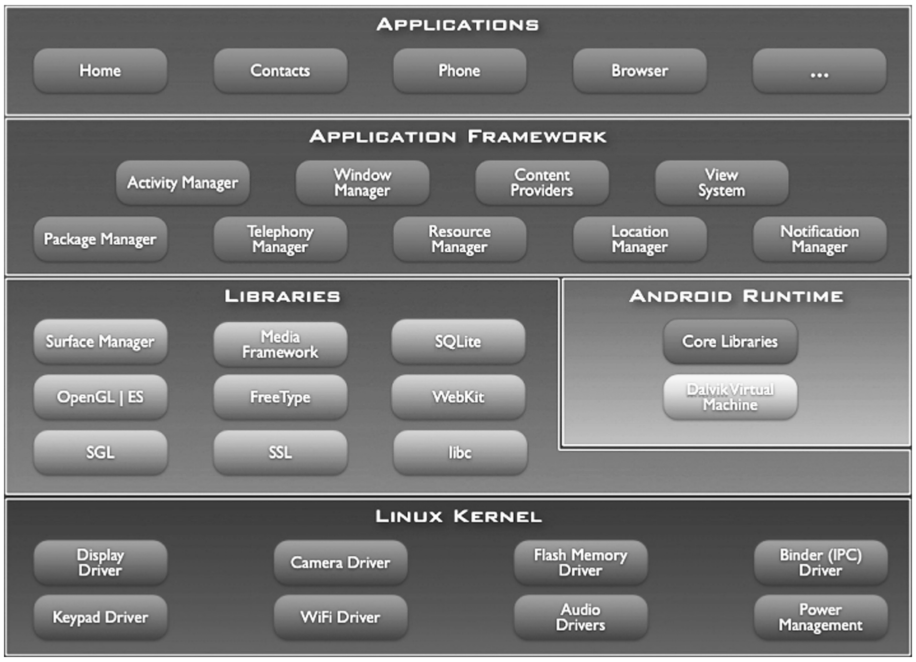


Fig. 1 – Major components of the Android Operating System.

be exploited in relation to the AF approach presented in this paper.

3.2. *The android security architecture*

As outlined in the Section 3.1, Android is a multi-process platform which relies on the standard Linux facilities for processes and users management; in fact, most of the security between applications is enforced at process level exploiting such standard facilities. However, in order to support the reuse of components and the provisioning of services between different applications, some finer-grained security features are provided by the mechanism of permissions.

3.2.1. *Applications and sandboxes*

Android, by default, denies to any application the capability to perform operations with the objective to hamper any other application, the OS or the end-user. Hence, due to this design pillar, for applications it is impossible to perform any operation on end-user private data (e.g., contacts, messages), to gain access to another application's files, to perform network accesses, to manage the device state, and so on.

Following this idea, Android binds any running application to a secure Sandbox which cannot interfere with any other applications, except by the explicit declaration of the required permissions to access to the desired capabilities which are not provided by the Sandbox. The set of permissions held by an application is defined in a static way, verified at installation time and cannot change during the lifetime of the application. Any Android application is required to be signed with a certificate, held by the developer, in order to establish and to manage relationships between applications.

3.2.2. *User IDs and permissions*

By default, Android manages each installed application as a different Linux user; in fact, at installation time, any application is provided with its own unique Linux user ID. All the data stored by a given application will receive the application's user ID as well; to grant to other applications any access to such data, it is required to enable the access from the Others group of Users.

By default, a basic Android application has no associated permissions; in order to overcome the limitations which could arise using only the DVM default capabilities, and to allow service provisioning between applications, it is possible to declare further permissions. The declaration of the needed permissions is performed at development time through the inclusion of <uses-permission> tags in the application's Android manifest.xml file. During the installation, the permissions required by the application are granted by the package installer module; the policy to grant a permission can leverage both on applications' signatures and on interaction with the end-user. Once the application is installed, the set of the granted and denied permissions is built and cannot be modified: during the execution, no more checks are performed.

4. Android anti-forensics

This section deals with depicting some of the possible instances of the AF techniques that have been identified in Arriving at an anti-forensics consensus (2006) and summarized in Section 2.2. It is worth to noticing that these instances do not represent a comprehensive set of all the possible techniques that can be implemented; in fact, some other possible instances are outlined in Section 6. Furthermore, all the implemented techniques exploit some features directly provided by the Android operating system, as described in Section 4.2, leading to the definition of three distinct automatic processes described in the Sections 4.3, 4.4 and 4.5.

4.1. *Current android forensic techniques and tools*

This section outlines the possible techniques currently available for Android Forensics; these techniques were firstly introduced by Hoog (backup tools are included in the set of the forensics tools).

4.1.1. *Android debug bridge(ADB)*

Android Debug Bridge (Android debug bridge tool) is a tool provided with the Android SDK (Android software development kit) which allows the interaction between the mobile device and a remote workstation (e.g., a personal or laptop computer). It is a Client-Server program composed by three main items: a client and a server both running at workstation side and a daemon running at emulator or device side. ADB allows the remote execution of commands onto the mobile device; if the device is the emulator, or if it has been rooted, the commands are executed with the root privileges, otherwise they are strictly limited. Several commands are available through ADB, some of them (e.g., dd, ls, pull) could be interesting for forensic purposes because allow the extraction of internal memory data.

4.1.2. *Nandroid backup*

Nandroid (Nandroid tool) is a set of tools supporting the backup and restore capabilities for rooted Android Devices. Nandroid seems to support the full NAND flash memory imaging which can be performed by a special boot mode. This mode can be very helpful in forensics; in fact, the capability to recover the entries deleted from the internal memory File System is not widely supported even by some forensic tools.

4.1.3. *Physical imaging by dd*

The availability of a Unix-like command shell can allow the usage of traditional forensic command line tools also for mobile devices. Among these, the dd tool allows the byte-level physical imaging of Unix files and can be applied to regular files and to device files as well. As many Unix storage partitions can be seen as a device file, this command can be very useful to perform the physical imaging of partitions; unfortunately, this command is useless for Android devices due to the features of YAFFS.

4.1.4. *Commercial tools*

Since the establishment of the Forensic Science, the availability of both commercial and free tools has been a contentious item; this phenomenon is true for the Mobile Forensics as well. Currently, there are several major manufacturers of commercial tools competing in the market; some of those (e.g., Paraben corporation official website; Micro systemation official website; Celle brite official website) support, or plan to support, Android mobile devices. On the other hand, Open Source tools still lack of competitiveness; at the time of writing, the only one that has been forensically sound proven (Distefano and Me, 2008) is Mobile Internal Acquisition Tool (MIAT).

4.1.5. *Serial commands over USB*

Regarding the specific model HTC Dream (also known as the Android G1), the Serial over USB connection is available; the cabling was reverse engineered (Android G1 serial to USB cable) allowing some capabilities (e.g., USB snooping) to eavesdrop the data conveyed over-the-wire. However, this approach is still less complete and requires much more testing; some other device models (e.g., HTC Magic, Samsung i7500 Galaxy) are not guaranteed to be compatible and the reconstruction of the used protocol is still work-in-progress.

4.1.6. *Simulated SD card*

This approach exploits a possible flaw in the update process of the Android G1. The underlying idea is to use a modified update file, in order to avoid the destruction of internal memory data and to provide kernel-level tools to support the acquisition of data. However, this technique has not been tested and represents just a possible, but uncertain, alternative.

4.1.7. *Software application*

Android, as any other Mobile Operating System, allows the third-party applications development. From a forensic perspective, the set of all the available Application Programming Interfaces (APIs) contains some interesting capabilities such those providing the interface with the File System. Such APIs could support the realization of common applications that are able to explore, read and mirror the contents stored by the File System even for the internal memory storage volume.

4.2. *Instantiating anti-forensics*

In this section, we depict some possible instances of the general techniques introduced in Section 2.1, focusing on an MD equipped with Android and on the related main ideas that are behind the work being presented.

4.2.1. *Exploiting android features*

As described in Section 3.2, Android relies on the strong Linux processes and users management policies, which have been improved and hardened with the introduction of the Sandbox execution approach. By default, every application, identified as a different user, runs in an isolated safe area. Furthermore, the protection of application's files is ensured by the file permissions management. In such scenario, if a given application desires to defend some files or directories, the protection is ensured and enforced at OS level.

4.2.2. *A private folder*

Thanks to the standard Android security features, for a given application it is possible to create, in the desired storage volume, a directory that is inaccessible for any other applications. Such directory, which can be defined as private, can be used to store any kind of information (e.g., text files, multimedia), it is created at install time and removed, including the entire content, when the owning application is uninstalled.

It is easy to figure out how this kind of folders can be exploited in order to perform some AF techniques; for instance, the folder could store all the compromising data selectively identified by the end user of the device. The inaccessibility of the folder for any other applications ensures the protection of the stored data, while the automatic deletion of the folder contents, at uninstall time, ensures the data to be logically erased. Obviously, if the volume where the private folder resides could be isolated and all the physically stored data acquired, the private data will be discovered; however, if the private folder is created in the internal memory of the device, both the isolation and the physical imaging, currently, are hard tasks.

Furthermore, regardless of the used storage volume, the private folder impedes the cursory examination of the device because any sensitive information, when migrated to the private folder, is actually made invisible to the end-user and to the default device's applications as well. Moreover, the private folder can be used to transfer any kind of compromising data (e.g., paedopornographic materials) without using removable volumes which can be easily investigated. Finally, it is worth noticing that the data stored by the private folder are not required to be encrypted; in such way, some issues related to cryptography (e.g., keys management) are avoided in favor of a kind of steganography which is ensured by the OS of the device.

4.2.3. *Anti-forensics by a common application*

The following sections describe some possible instances of the AF techniques, relying on the private folder mechanism, which have been successfully implemented and tested both on the device emulator and on an actual device. It is worth noticing that all the following techniques have been implemented by a common Android application that can be installed and executed onto the device; such application is called AFDroid.

At install time, AFDroid creates the private folder while, at execution time, it allows the execution of two distinct processes:

- the Evidence Export Process: described in Section 4.3, it refers to the application of the AF techniques;
- the Evidence Import Process: described in Section 4.4, it aims at reversing the Export Process.

Finally, the AFDroid application allows also a secure Evidence Destruction Process which is quick and straightforward; such process is described in Section 4.5.

4.3. *The evidence export process (EEP)*

This process, depicted in Fig. 2, represents the fully automated instance of the AF techniques which have been introduced in Section 2.1; for each technique, we developed the related features exploiting the Android Application Framework. For practical reasons, it is unfeasible to describe in detail the

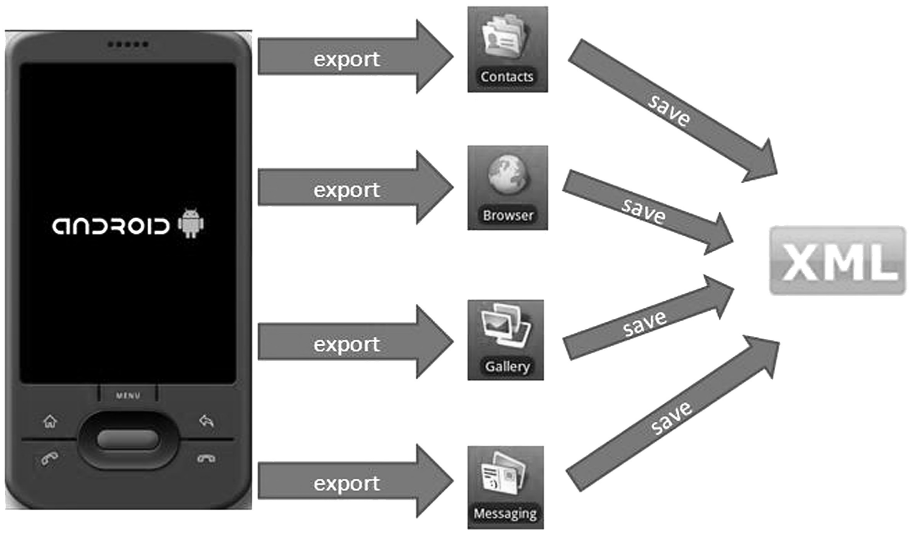


Fig. 2 – Representation of the idea behind the EEP. The export.xml file that is produced contains the evidence gathered by the target Android databases and it is stored by the private directory.

realization of each feature; however, in the following sections, we group and briefly describe them in relation to the AF techniques they implement.

4.3.1. *Android destroying evidence*

This technique has been applied to text messages, to the browser bookmarks and to the call log, in order to delete, from the related databases, any records carrying sensitive information. The deletion of such database records, and the subsequent secret storage performed by the private folder, is a typical example of evidence destruction: the investigator cannot find any information by the examination of the target device.

4.3.2. *Android hiding evidence*

This technique has been applied to multimedia files that are stored in the removable memory card; as described in Sections 2.3 and 4.2.2, this storage volume can easily investigated and the compromising data discovered. In order to hide such evidence, it is enough to store the sensitive multimedia data into the private folder, which is stored in the internal memory; in such way, the multimedia management applications cannot index the data and the cursory examination of the device is unable to detect such materials.

4.3.3. *Android eliminating evidence sources*

This technique has been applied to MultiMedia Messages (MMSs) managed by Android in terms of conversations, like Gmail e-mail messages management. Each conversation is marked by an identifier which allows the reconstruction of the flow of messages and each sent/received MMS must be bound to a given conversation. I.e., the inhibition of the creation of compromising MMS evidence can be achieved by tampering the mechanism of conversation identifiers. More precisely, we selectively modified the conversation identifiers which can refer to any flow of sensitive MMSs; in such way, any related MMS cannot be properly indexed by the system and, although it is properly processed, it results as invisible to the end-user.

4.3.4. *Android counterfeiting evidence*

This technique has been applied to contacts information. Android provides, in addition to the personal data (e.g., name, surname, phone numbers), two interesting information about contacts: a flag that states if the contact is among the preferred ones, and the related number of performed interactions. Such information can be helpful in order to discriminate the more frequently touched contacts from the others; this evidence can lead to a fast identification of strong relations between contacts. By the modification of such values, it is possible to counterfeit such evidence, leading to confounding data or to the lost of evidence.

4.4. *The evidence import process (EIP)*

The private folder mechanism is fundamental in order to protect the evidence (Section 4.3), however it can be used as the pillar in order to reconstruct the evidence previously exported. In fact, the occulted evidence could be as precious as compromising; hence, the capability to restore the previous state of the device could be a valuable knowledge asset. The EIP, described in this section, is able to restore the previous state of the device, in order to reverse the EEP.

4.4.1. *The private storage of the evidence*

In order to support the reconstruction of the exported data, it is mandatory to store the evidence in some way; we chose to organize the entire exported evidence using a set of common files contained by the private folder. This set is composed by a single XML-styled file (called export.xml) and by a number of files of various formats. The first file is responsible for the storage of all the evidence gathered by the internal databases, while the other files are those imported by the removable memory card (e.g., multimedia files). For practical reasons, it is impossible to detail the structure of the export.xml file; however, some sample fragments are shown in Figs. 3, 4 and 5.

4.4.2. *How to reconstruct the evidence*

Trivially, it is possible to use some device specific tools properly designed to restore system images (e.g., the Nandroid tool); however this approach presents three main drawbacks. The former is the availability of a recent and updated system image, otherwise a lot of information will be lost, the second is the mastery of the specific tools that are often designed for programmers or Android system experts and the latter is the availability of the entire equipment required for the restore process. Due to these three items, we can state that the usage of some specific tools in order to reverse the Export Process seems quite complex and, of course, it cannot be quickly performed.

4.4.2.1. *Fully automated evidence reconstruction*. In order to support the EIP in a straightforward way, we enriched the AFDroid application with features able to reverse, in a fully automated way, the EEP. The steps being performed during the evidence reconstruction are the following:

1. Private folder inspection: in order to properly identify the occulted evidence, the private folder is examined;
2. export.xml file processing: in order to restore the evidence collected by the various system's databases, such file is processed leveraging on a SAX XML-parser:
   (a) For each entry, the related database and table are identified by the meta-data stored by the export.xml file;
   (b) For each entry, the connection to the related database is established and the required query is performed.
3. Other files processing: in order to restore any additional generic entries which have been previously imported, its original absolute path is retrieved through the export.xml file and the entry is created as new.

```
<database name='MMSSMS'>
<table name='sms'>
<row>
    <col name='_id'>977</col>
    <col name='thread_id'>15</col>
    <col name='address'>YYYYYYYYYYYYY</col>
    <col name='person'>1148</col>
    <col name='date'>1265591133661</col>
    <col name='protocol'>0</col>
    <col name='read'>1</col>
    <col name='status'>-1</col>
    <col name='type'>1</col>
    <col name='reply_path_present'>0</col>
    <col name='subject'>null</col>
    <col name='body'>Text of the message</col>
    <col name='service_center'>XXXXXXXXXXXXX</col>
</row>
</table>
</database>
```

Fig. 3 – Sample fragment representing an SMS message that has been exported into the export.xml file.

```
<database name='CONTACTS'>
<table name='calls'>
<row>
    <col name='_id'>2896</col>
    <col name='number'>YYYYYYYYYYYYY</col>
    <col name='date'>1263580272900</col>
    <col name='duration'>288</col>
    <col name='type'>1</col>
    <col name='new'>1</col>
    <col name='name'>NameOfContact</col>
    <col name='numbertype'>2</col>
    <col name='numberlabel'>null</col>
</row>
</table>
</database>
```

Fig. 4 – Sample fragment representing a call that has been exported into the export.xml file.

4.4.2.2. *Evidence reconstruction and forensic properties*. The automatic process for the reconstruction of the occulted evidence, that has been described in the previous section, leverages on the capability of restoring both the generic files and the databases contents; obviously, some basic forensic properties cannot be assured. For instance, the last-modification-time file attribute, that is often used in the analysis of a digital evidence, cannot be preserved because of the ex-novo creation of the target file. Furthermore, it cannot be guaranteed the integrity preservation of the whole structure of all the databases, because of the typical compression and optimization tasks the database management systems perform. For these reasons, we can state that the Evidence Export process is reversible from the perspective of the end-user, even if it lacks of some Forensic requirements, mandatory to define such process as Forensically sound repeatable. However, due to the proposed scenario of a subject that needs to protect some possible sources of evidence with the capability to recover them, the missing Forensic properties just mentioned are less important and, often, useless.

```
<database name='BROWSER'>
<table name='bookmarks'>
<row>
    <col name='_id'>1</col>
    <col name='title'>Google</col>
    <col name='url'>http://www.google.com/</col>
    <col name='visits'>7</col>
    <col name='date'>1263761875450</col>
    <col name='created'>0</col>
    <col name='description'>null</col>
    <col name='bookmark'>1</col>
</row>
</table>
</database>
```

Fig. 5 – Sample fragment representing a browser bookmark that has been exported into the export.xml file.

4.5. *The evidence destruction process (EDP)*

When a given common Android application is uninstalled, the entire set of the related information, including data files and directories, is logically deleted from the File System. Obviously, as anticipated in Section 4.2.2, this also applies to any private folders and to the related content. Regarding the objective of the Evidence Destruction, it is interesting to investigate the possibility that a deleted data can be identified and, possibly, retrieved through some forensic techniques and tools currently available both in the literature and as COTS.

4.5.1. *Internal memory and data recovery*

The master thesis discussed in (Regan, 2009) details the forensic potential of flash memory physical images, even describing the Android FS. Regarding mobile forensics, a standard method to acquire such kind of images of the internal memory is still missing although a lot of effort has been made (e.g., J.W. and A.R., 2007; Jansen et al., 2008; NIST, 2010). Prior works (e.g., Breeuwsma et al., 2007; Breeuwsma, 2006) proposed and focused mainly on the application of JTAG in order to acquire an intact physical image of the storage circuitry. However, the practical application of JTAG to mobile forensics is still limited by its usability; often, the required connectors are not easily identifiable, nor published by the manufacturers and the required equipment can be complex. Furthermore, after the factory quality testing being performed onto prototypes, a significant number of mobile device models are commercialized with a disabled JTAG interface which impedes any attempts to acquire the physical image. More recently, some commercial hardware tools (e.g., Paraben corporation official website) advertised the physical copy feature; however, often, their compatibility with models is very limited and the outcomes are not guaranteed.

In the experiments being presented in Section 5, we adopted the Nandroid tool in order to acquire the images of the internal memory of the device. It must be noted that Nandroid is not a forensic tool, but it is able to perform a copy of the entire storage volume we target; however, at the time of writing, it is unclear if the Nandroid tool is able to produce a complete image of the data physically stored by the internal memory. The images produced by Nandroid can be extracted and examined using the unyaffs tool that is freely available.

4.5.2. *A fully automated process*

As described in Section 4.4.1, we chose to store all the occulted evidence in the private folder held by AFDroid; hence, the uninstall of such application can represent a straightforward and fully automated EDP. The end-user is only required to start the uninstall of AFDroid and all the related data are logically deleted by the File System. The time required to uninstall a generic Android application can depend on the amount of the related data to be deleted; however, during the experiments described in Section 5, it was always less than 5 s. Hence, when compared to the typical effort required to perform a manual evidence destruction, we can argue that such time is surely negligible. Furthermore, the fully automated evidence destruction can avoid any typical human errors (e.g., forgetting some elements), leading to a systematic evidence destruction. Moreover, among the possible improvements to AFDroid, is expected to implement a finer-grained mechanism to select the evidence to be processed (e. g., keywords driven, identity driven) as described in the conclusions of this paper (Section 7). In such case, the overall amount of the occulted data to be deleted, while uninstalling AFDroid, will be significantly reduced and the time required for EDP shortened. Finally, as described in Section 4.5.1 and confirmed by the experimental results described in Section 5.2.2, the straightforward recovery of the application data, which have been deleted by the uninstall process, is still difficult.

5. Experiments

This section describes the experiments which have been defined, planned and executed in order to test both the EEP and the EDP; the reason for the selection of such processes is described in Section 5.1.1. The Section 5.1 describes the definition and the planning of the experiments, while the Section 5.2 provides the related results.

5.1. *Experiments definition*

5.1.1. *Objectives*

The objective of both the experiments is to test the strength of the selected processes in relation to the tools that are currently able to acquire a snapshot of the internal memory of the target device. Hence, it is important to define what we mean for the strength of the related process. AF techniques aim at impeding the acquisition and analysis of evidence. For such reason, the strength of a given process that instantiates some AF techniques is inversely related to the capability to recover the processed evidence.

The identified AF techniques are fully implemented by the EEP (described in Section 4.3); furthermore, the EDP (described in Section 4.5) requires to definitely destroy all the exported evidence. Hence these are the two processes on which the experiments focus, while the EIP (described in Section 4.4) is not considered.

5.1.2. *Used devices and replication*

Android devices can be of two main types: device emulators and actual devices. However, since the emulators can significantly differ from the actual devices, we focused only on actual devices. All the experiments were performed on the same Samsung Galaxy i7500 device equipped with the Android 1.5 Software Development Kit.

The replication is one of the most crucial aspects in forensic experimentation; although it provides more reliable results when applied to forensics, it has to cope with legal issues and limitations. Often, the most representative devices are those seized in real investigations, but access to this kind of devices is restricted. In our experiments, we used a single device; however, this does not undermine the conclusions drawn because our objective is simply to test the effectiveness and the strength of the EIP and EDP.

5.1.3. *Used acquisition tools*

In order to test the strength of the aforementioned processes, the capability to acquire the data stored by the internal memory of the device is required.

During our experiments, we planned to use both commercial tools, general purpose tools and a local application specifically designed to acquire the data logically stored by the internal memory file system. Among the commercial forensic tools, we selected the Paraben Device Seizure (Paraben corporation official website) that is freely available for demo and testing. Unfortunately, this tool revealed some incompatibilities with the device model we have, so that it was impossible to perform the data acquisition. Regarding the general purpose tools, we used Nandroid to perform the imaging of the actual device we have. Furthermore, we developed a custom Android application specifically designed and implemented to perform the logical copy of the data stored by the file system of the internal memory. This application, named MIAT, represents the Android instance of the local forensic acquisition paradigm which has been firstly proposed for Symbian devices in (Me and Rossi, 2008), instanced for Windows Mobile devices in (Dellutri et al., 2008) and analyzed, in the scope of the forensic properties, in (Distefano and Me, 2008). The detailed description of such application will be the main subject of a subsequent paper.

5.1.4. *Experimental workflows*

This section describes the experimental workflows both for the EEP and for the EDP.

1. Evidence export process

The experimental workflow related to this process is the following:

- First imaging with the Nandroid tool;
- Execution of AFDroid;
- Acquisition with the MIAT tool;
- Second imaging with the Nandroid tool.

2. Evidence destruction process

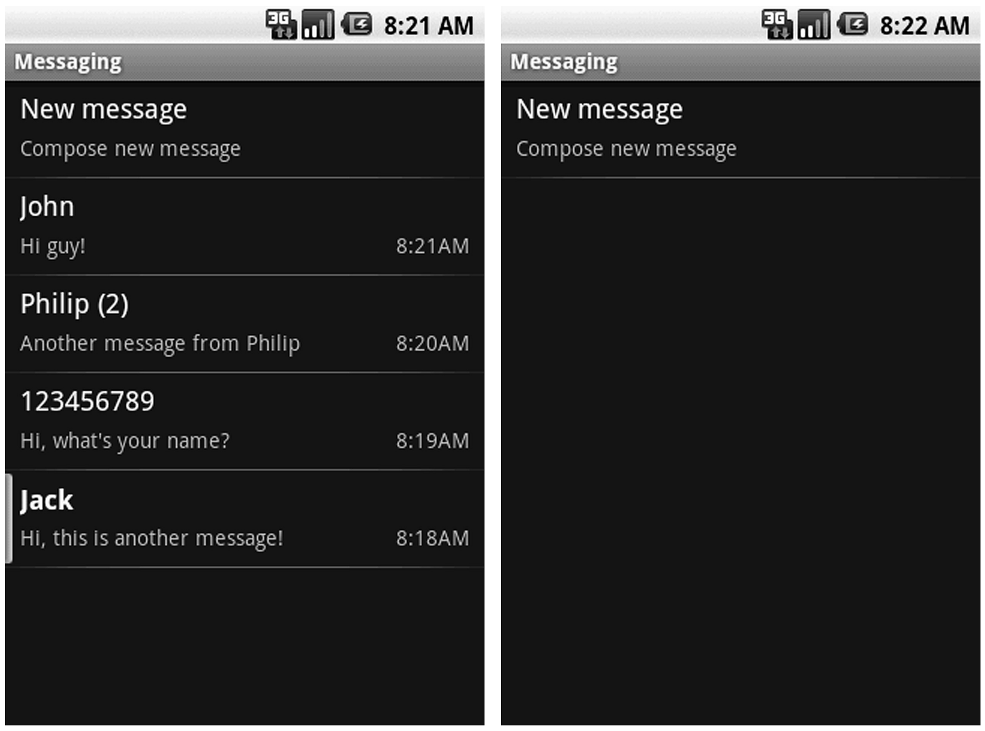
The experimental workflow related to this process is the following:

- First imaging with the Nandroid tool;
- Execution of AFDroid;
- Second imaging with the Nandroid tool;
- Uninstall of the AFDroid;
- Acquisition with the MIAT tool;
- Third imaging with the Nandroid tool.

5.2. *Experimental results*

In this section we discuss the results obtained with the information acquired during the experiments. We considered two different kinds of analysis of the target device:

- Cursory examination: consists in analyzing the evidence stored by a device simply by the examination of the given device. In this kind of analysis, the examiner behaves like a common end-user.



8:21 AM
Messaging
New message
Compose new message
John
Hi guy! 8:21AM
Philip (2)
Another message from Philip 8:20AM
123456789
Hi, what's your name? 8:19AM
Jack
Hi, this is another message! 8:18AM

8:22 AM
Messaging
New message
Compose new message

a Before EEP b After EEP

Fig. 6 – Cursory examination of the SMS/MMS database before and after the EEP. The entire set of SMS/MMS messages is emptied.

Table 2 – Size differences (in k Byte) between the files that store the database affected by the EEP.

File	Before	After
com.android.browser/databases/browser.db	58	5
com.android.providers.telephony/databases/mmssms.db	189	48

• Acquisition & Analysis of the internal memory: uses the abovementioned acquisition tools to extract the data stored by the device, the acquired data are subsequently examined. In this kind of analysis, the examiner act as a forensic operator.

5.2.1. *Evidence export process*

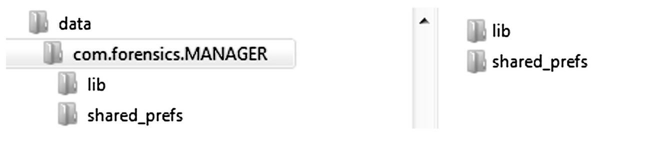
As described in Section 4.3, during this process the evidence is processed and stored by the private folder. After this task, any cursory examination of the device shows the following situation:

• There are no differences, in terms of number of interactions, among the contacts;
• The database of the SMS and MMS messages is empty;
• The call log is empty;
• The multimedia gallery is empty.

The Fig. 6 shows the cursory examination of the database of messages before and after the EEP.

Regarding the analysis of the acquired data, we can exploit the information extracted both by the Nandroid and by the MIAT tool. As described in Section 4.5.1, at the time of writing, the former data can be extracted only with the unyaffs tool; while the latter is already represented as a logical structure of the file system. Both the tools were able to recover all the evidence that has been previously exported in the private folder. Furthermore, it is possible to compare the structure of the data acquired before and after the EEP, in order to identify the files and directories that change. Table 2 shows the difference in size of the database files that are affected by the EEP; while the Figs. 7 and 8 respectively show the contents of the acquired private folder before and after the EEP.

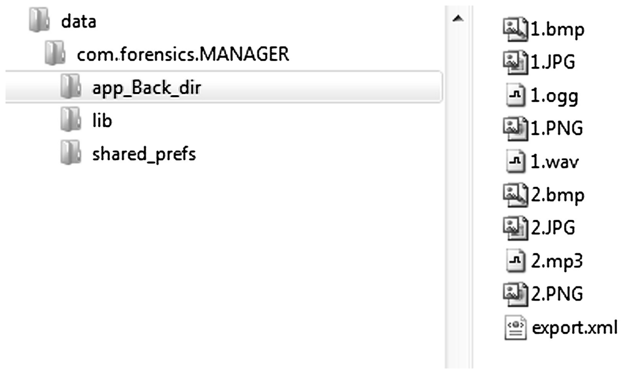
5.2.1.1. *Duration of the process*. In order to show the practical feasibility of the EEP, we have measured the duration of such process with two different amounts of evidence to be exported. We have built the different test cases as described in Table 3; Fig. 9 shows the duration in seconds of the EEP in relation to both the amount and the kind of evidence. For the sake of simplicity, during the process we do not consider multimedia data; in fact, the evidence to be exported is only related to the Android databases. Obviously, a large amount of multimedia data can negatively affect the duration of the process; however, due to the typical limited capacity of the current internal memory, it is realistic to suppose that just a reduced amount of such data can be exported into the private folder.



```
data
  com.forensics.MANAGER
    lib
    shared_prefs
```

```
lib
shared_prefs
```

Fig. 7 – Contents of the private folder before the EEP.



```
data
  com.forensics.MANAGER
    app_Back_dir
    lib
    shared_prefs
```

```
1.bmp
1.JPG
1.ogg
1.PNG
1.wav
2.bmp
2.JPG
2.mp3
2.PNG
export.xml
```

Fig. 8 – Contents of the private folder after the EEP; for practical reasons, the exported contents are kept limited.

5.2.2. *Evidence destruction process*

In order to assess also the final destruction of the processed evidence, we tested the effectiveness of the EDP described in Section 4.5. The remarks given in Section 5.2.1 for the EEP are still valid until the disinstallation of AFDroid is performed. In fact, both the Nandroid and the MIAT tools were able to recover the exported evidence if the AFDroid application is installed, because its private folder is still resident in the file system. However, when the application is uninstalled and the EDP completed, such folder is removed including all the stored contents; after that, neither the Nandroid nor the MIAT tools were able to recover the deleted data.

In addition, we tried to process the Nandroid image acquired right after the disinstallation of AFDroid with the Scalpel file carver (The scalpel file carver official website), in order to test if any remnant data can be recovered. Scalpel was able to recover many files that were logically stored by the internal memory, but was unable to recover any files stored by the private folder that have been removed. At the time of writing, it is still unclear if the Nandroid tool is able to produce an actual physical dump of the internal memory; anyway, the file carving performed onto the images produced by the Nandroid tool was unable to recover any remnant data of the previously exported evidence.

6. Current and future work

The work being presented in this paper aims at raising about the possibility to instantiate to mobile devices some common AF techniques in a relatively straightforward manner. In

Table 3 – Test Cases for the EEP described in terms of number of elements stored by the target databases.

Load	Contacts & calls	SMS & MMS	Bookmarks
Medium	120	250	45
High	222	591	93

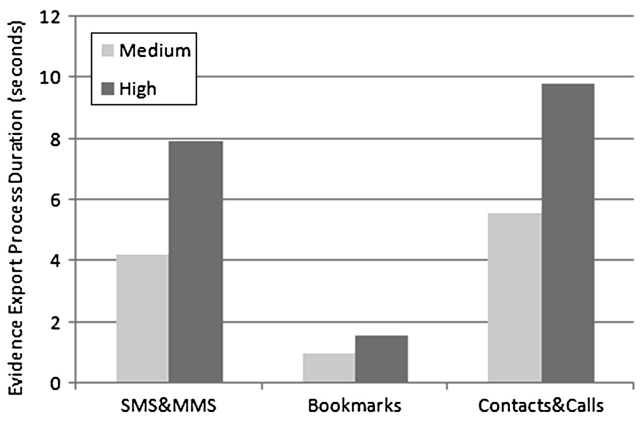


Fig. 9 – Duration, measured in seconds, of the EEP for two different amounts of Evidence to be processed.

particular, we focused on Android devices that are among the most promising in terms of market forecasts. At the time of writing, we are working following two different directions:

- Improving the AFDroid application that has been developed: among the possible improvements, it is possible to notice the capability to selectively choose the target evidence (e.g., using automatic techniques, such as Bayesian classifiers, to discriminate between different classes of messages) and the expansion of the kinds of target evidence (e.g., considering the evidence related to the installed applications and their usage).
- Expanding the compatibility to other operating systems: regarding the mobile environment, it is possible to identify different operating systems (e.g., Windows Mobile, Symbian) that can be targeted in order to study the practical feasibility of instantiating some AF techniques.

7. Conclusions

AF is a quite young and immature discipline, even more when contextualized to the ME. Several efforts have been made in order to properly describe and classify the widespread AF techniques (e.g., Arriving at an anti-forensics consensus, 2006), but they do not focus on mobile devices.

In this paper, we recalled the classification of the Anti-Forensics techniques that have been proposed in Arriving at an anti-forensics consensus (2006). Furthermore, we have proposed some possible instances of such techniques to the mobile environment and, in particular, to Android mobile devices. The instances we proposed were fully automated and supported by a common Android application, called AFDroid, that has been specifically designed and implemented. Finally, in order to test the effectiveness and strength of the implemented AF techniques, we planned and performed some experiments proving that AFDroid is able to hold on versus both the cursory examination of the device and some tools able to acquire the data from the internal memory of Android devices.

REFERENCES

ACPO. Association of chief police officers official website. Available: http://www.acpo.police.uk/; 2010 [Online].

An introduction to NAND flash [Online]. Available: http://www.commsdesign.com/showArticle.jhtml?articleID=183700957; 2006.

Android debug bridge tool [Online]. Available: http://developer.android.com/guide/developing/tools/adb.html.

Android G1 serial to USB cable [Online]. Available: http://www.instructables.com/id/Android_G1_Serial_Cable/.

Android security architecture [Online]. Available: http://developer.android.com/guide/topics/security/security.html; 2010.

Android software development kit [Online]. Available: http://developer.android.com/guide/index.html.

Arriving at an anti-forensics consensus: e xamining how to define and control the anti-forensics problem. Digital Investigation 2006;3(Suppl. 1):44–9. the Proceedings of the 6th Annual Digital Forensic Research Workshop (DFRWS '06).

B. S. The rise of anti-forensics [Online]. Available: http://www.csoonline.com/article/221208/The_Rise_of_Anti_Forensics; 2007.

Berghel H. Hiding data, forensics and antiforensics. Communications of the ACM 2007;50(4):15–20.

Breeuwsma M, De Jongh M, Klaver C, van der Knijff R, Roeloffs M. Forensics data recovery from flash memory. Small Scale Device Forensics Journal 2007;1:1–17.

Breeuwsma IMF. Forensic imaging of embedded systems using JTAG (boundary-scan). Digital Investigation 2006;3:32–42.

Carrier B. "Open source digital forensics tools: the legal argument," 2003.

Celle brite official website [Online]. Available: http://www.cellebrite.com/.

Dellutri F, Ottaviani V, and Me G. "MIAT-WM5: forensic acquisition for Windows mobile PocketPC," Proceedings of the 2008 Workshop on Security and High Performance Computing Systems, part of HPCS, 2008.

Distefano A, Me G. An overall assessment of mobile internal acquisition tool. Digital Investigation 2008;5:S121–7.

Foster J, Liu V. Catch me if you can. Available: http://www.blackhat.com/presentations/bh-usa-05/bh-us-05-foster-liu-update.pdf; 2005 [Online].

Gartner mobile OS share forecast. Available: http://techcrunchies.com/gartner-mobile-os-market-share-forecast/; 2009 [Online].

Geiger M. "Evaluating Commercial Counter-forensics Tools." 5th Ann. Digital Forensic Research Workshop, 2005.

Hoog A. Android forensics [Online]. Available: http://www.mobileforensicsworld.org/2009/.

IOCE. International organization on computer evidence official website. Available: http://www.ioce.org; 2010 [Online].

J.W. and A.R. "Guidelines on Cell Phone Forensics," 2007.

Jansen W, Delaitre A, and Moenner L. "Overcoming Impediments to Cell Phone Forensics." 41th Annual Hawaii International Conference on System Sciences, 2008.

Kalba K. The adoption of mobile phones in emerging markets: global diffusion and the rural challenge. International Journal of Communication 2008;2:631–61.

Kinam K. "Memory Technologies for Mobile era," Asian Solid-State Circuits Conference, 2005, pp. 7–11.

Me G. and Rossi M. "Internal forensic acquisition for mobile equipments," 4th International Workshop on Security in Systems and Networks, Proceedings of the International Parallel and Distributed Processing Symposium, 2008.

Micro systemation official website [Online]. Available: http://www.msab.com/.

Nandroid tool [Online]. Available: http://svn.infernix.net/nandroid/.

NIST. Information technology laboratory: computer forensics tool testing program. Available: http://www.cftt.nist.gov/mobile_devices.htm; 2010 [Online].

Paraben corporation official website [Online]. Available: http://www.paraben.com/.

Peron CSJ. Digital antiforensics: emerging trends in data transformation techniques [Online]. Available: http://www.seccuris.com/documents/whitepapers/Seccuris-Antiforensics.pdf.

Regan J.E. "The Forensic Potential of Flash Memory – Master thesis approved for public release," Master's thesis, Naval Postgraduate School – Monterey, California, 2009. [Online]. Available: http://simson.net/clips/students/09Sep_Regan.pdf.

S.B. Anti-forensics. Available: http://www.aversion.net/presentations/HTCIA-02/anti-forensics.ppt; 2002 [Online].

The scalpel file carver official website [Online]. Available: http://www.digitalforensicssolutions.com/Scalpel/.

van der Knijff R. 10 good reasons why you should shift focus to small scale digital device forensics [Online]. Available: http://dfrws.org/2007/proceedings/vanderknijff_pres.pdf; 2007.

Williams G. Mobile forensics turns up heat on suspects [Online]. Available: http://www.theregister.co.uk/2007/02/11/.

YAFFS official website [Online]. Available: http://www.yaffs.net; 2010.