# A Novel Anti-Forensics Technique for Android OS

Pietro Albano*, Aniello Castiglione†, Giuseppe Cattaneo§, Alfredo De Santis‡
Dipartimento di Informatica "R.M. Capocelli"
Università degli Studi di Salerno
I-84084 Fisciano (SA), Italy
pietro.albano@gmail.com*, castiglione@ieee.org†, cattaneo@dia.unisa.it§, ads@dia.unisa.it‡

***Abstract*—In questi ultimi anni i classici cellulari, utilizzati solo per effettuare chiamate ed inviare SMS, si sono evoluti in dispositivi sempre più versatili e potenti (SmartPhone, Tablet, etc.). Tali dispositivi adottano come strumento di archiviazione flash memory di tipo NAND, memorie ottimizzate per l'aggiornamento rapido dei dati. Tali flash memory contengono dati sensibili che possono essere un pericolo per la privacy degli utenti.**

**In questo articolo viene proposta una nuova tecnica Anti-Forensics per dispositivi mobili con Sistema Operativo Android. Tale tecnica rende possibile la modifica e la cancellazione, sicura e selettiva, delle digital evidence presenti in un dispositivo Android senza l'ausilio di primitive crittografiche o modifiche al file system. Mentre l'utilizzo di primitive crittografiche o modifiche apportate al file system desterebbero non pochi sospetti in una indagine forense, la tecnica proposta utilizza semplici strumenti software di uso comune in Sistemi Operativi *nix-like quale Android.**

***Index Terms*—Digital Forensics; Mobile Forensics; Anti-Forensics; Mobile Anti-Forensics; Counter-Forensics; Android OS; Android Forensics; Android Anti-Forensics; Flash Memory; NAND; Secure Deletion; Sanitization.**

## I. Introduzione

Nel 2011, ha sottolineato Gartner [1], dovrebbero essere venduti 468 milioni di dispositivi mobile, vale a dire il 57.7% in più rispetto allo scorso anno. Sempre secondo la società di ricerche, nel 2012 saranno 632 milioni di SmartPhone nel mondo e nel 2015 saranno 1.1 miliardi. Nel 2012 metà (49.2%) dei 632 milioni di SmartPhone in circolazione avranno cuore Android.

I dati presenti in uno SmartPhone possono rivelarsi una minaccia alla privacy degli individui. Basti pensare che un classico cellulare racchiude al suo interno un gran numero di informazioni private (contatti, SMS, lista delle chiamate). Altre informazioni sensibili contenute negli SmartPhone sono e-mail, cronologia dei siti web visitati, messaggi scambiati in chat, informazioni di geolocalizzazione, etc. Queste informazioni, sommate a quelle precedentemente citate, permettono di individuare un profilo ben definito del suo possessore e di ricostruire le sue azioni in un preciso spazio temporale. Un'individuo che vuole difendere la propria privacy ed in particolare cancellare o modificare in maniera sicura e selettiva delle digital evidence presenti sul prorpio SmartPhone può utilizzare tecniche anti-forensics. A definition of anti-forensics, according to [2], is *"...we will consider anti-forensics to be any attempts to compromise the availability or usefulness of evidence to the forensics process. Compromising evidence availability includes any attempts to prevent evidence from existing, hiding existing evidence or otherwise manipulating evidence to ensure that it is no longer within reach of the investigator. Usefulness maybe compromised by obliterating the evidence itself or by destroying its integrity."*.

While no paper has addressed the problem of secure modification in flash memories (to the best of author's knowledge), recently alcuni lavori hanno proposto tecniche di cancellazione sicura per le flash memory NAND che sono di comune utilizzo negli SmartPhone.

Flash memory drives (NANDs) differ from hard drives in both the technology they use to store data (flash chips vs. magnetic disks) and the algorithms they use to manage and access that data. NANDs maintain a layer of indirection between the logical block addresses that computer systems use to access data and the raw flash addresses that identify physical storage. The layer of indirection enhances NAND performance and reliability by hiding flash memory's idiosyncratic interface and managing its limited lifetime, but it can also produce copies of the data that are invisible to the user but that a smart attacker can recover. The differences between NANDs and hard drives make it uncertain whether techniques and commands developed for hard drives willl be effective on NANDs.

In [3] ed in [4] tutti i dati vengono cifrati, ogni file possiede la propria chiave di cifratura la quale è memorizzata nell'header del file stesso. Quando si vuole cancellare un singolo file basta cancellare o sovrascrivere il suo header. Cifrare il file system o modificarlo per applicare tecniche anti-forensics, desta non pochi sospetti in una analisi forense.

In [5], sfruttando una security feature di Android, possibili digital evidence vengono nascoste in *private folder* inaccessibili da applicazioni di terze parti. Le private folder sono directory private, create nel momento in cui viene installata un'applicazione, all'interno delle quali è possibile salvare qualsiasi tipo di file (e.g., file di testo, file multimediali). According to [5] 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.

In questo articolo viene introdotta una nuova tecnica Anti-Foresics per dispositivi mobile con Sistema Operativo Android

Corresponding author: Aniello Castiglione, Member, *IEEE*, castiglione@ieee.org, Phone: +39089969594, FAX: +39089969821 che permette di modificare e cancellare, in modo sicuro e selettivo, le digital evidence generate dal Sistema Operativo Android utilizzando semplici tool software, senza apportare modifiche al file system e senza l'uso di primitive crittografiche.

The paper is organized as follows: la sezione II fornisce una breve descrizione dell'OS Android, con particolare attenzione al tipo di File System adottato generalmente dai dispositivi Android ed ai supporti di memorizzazione (NAND) presenti in questi ultimi. Section III illustra le principali tecniche forensi utilizzate su dispositivi Android. La sezione IV illustra la tecnica anti-forensics proposta; mentre la sezione V describes the instances, to Android devices, of such techniques that have been designed, implemented and tested. L'articolo termina con le conclusioni (sezione VI).

## II. The Android OS

Android is a set of open-source software elements specifically designed for mobile devices developed by Google. It includes the Operating System, a middleware and a set of applications. Although it has been designed and developed for mobile devices (e.g., Smartphones, Tablet, etc.).

### A. The Android File System

The vast majority of mobile devices running the Android OS uses, and natively supports, the YAFFS (Yet Another Flash File System) file system, developed in 2002 specifically to be used on NAND flash memories and being completely open-source. To the authors knowledge, the YAFFS is the only file system type which fully support the flash memory technology [6]

YAFFS has several interesting features such as *journaling*, *error correction* and *verification* techniques very useful to deal with hardware failures/problems intrinsic to the flash memory technology. The Android OS is based on the Linux kernel v2.6 and, thanks to it, allows the distribution of its file system on different storage device, both internal flash memory and on the microSD that can be inserted in the mobile device.

In Table I is described the structure and the organization of the file system present on the mobile device adopted for the case study. The organization of the partitions as well as the mount points shown in I may change on different devices. In the column named "Partition" are listed the storage device and the partitions of the internal flash memory and of the microSD. The partitions host different data: `mtd0` contains miscellaneous data, `mtd1` the *recovery image*, `mtd2` the boot partition, `mtd3` the system files, `mtd4` the system cache and `mtd5` the user data. The "Mount Point" column lists only those partitions which are accessible from the file system, in other words, only the partitions that can be mounted on a branch of the file system. The last column of Table I tells whether the partition is present on the "Internal" flash memory or on the "External" microSD. The most interesting partitions that are interesting to be considered during a digital forensics analysis are the `mtd3` and the `mtd5` which contains respectively the core of the OS and the work space of the user.

The partition `mmcblk0p1` usually formatted with the FAT file system type, represents the external memory of the mobile device and it left to the user wishes. The partition `mmcblk0p2`, which is formatted with the EXT3/EXT4 file system type, starting from Android v2.2 is used as an extension of the internal memory of the mobile device. Such partition is accessible only by the OS and by the *root* user.

### B. NAND flash memory

Flash memory is a solid-state storage technology and hence non-volatile, which can be used as a read-write memory thanks to its performance. Mainly there exist two kind of flash memory: the one called *NOR* and the one called *NAND*. These two typology of memory differ in the architecture they are designed and in the way they are programmed. In addition, there exists an hybrid kind of flash memory, the *AND* flash, which takes advantages of both the characteristics of NOR and NAND. The NAND flash memories are optimized for the fast update of data. It is to be considered that the smallest block that can be delated in a NAND is 8 Kb which is different from the 64Kb of the NOR.

When data in a memory flash are updated, it is not possible to program the same page due to the one-way programming peculiarity of flash technology, so the page containing the to-be-updated data is entirely rewritten to a new location either in the same block or not). In the spare area, the page with new data is marked as allocated, while the old one is marked as unallocated (deleted or obsolete) [7].

### C. Digital Evidence in Android

The partition `mtd5` maintains the third party applications as well as all the logs related to such applications or generated by the OS itself. In [8] are presented and analyzed some of the most useful tools in a digital forensics investigation on an Android OS device.

## III. Android Forensics

In this section the main tools for the forensic acquisition and analysis of devices equipped with the Android OS are described. The forensic acquisition can be performed using logical and/or physical operations. Logical operations are performed only on allocated data and are usually executed by accessing the file system. Allocated data is organized in the file system structure and is accessible through it. Physical operations, on the other hand, do not use the file system to access the data but interact directly with the physical storage media.

### A. Tools for the logical acquisition

There are essentially three different kind of tools for logical acquisition: Nandroid Backup is a set of tools and a script which enable to backup an Android device provided that the root privileges are granted. The backup can be considered as a logical copy of the flash memories involved (the whole internal memory and */sd-ext* of the microSD).

Third party applications are those developed by the developer community using the standard Application Programming

Interfaces (API) supplied by Google. Such APIs can access the entire device including digital evidence present on the `mtd5` partition. There are several applications able to extract useful logical information from the file system such as AFLogical Tool developed by ViaForensics [9].

There are a few commercial tools for the logical acquisition such as Oxygen Forensics Suite 2011 by Oxygen, Android Logical Plug-in by Paraben and Mobile Phone Examiner by Access data.

### *B. Tools for the physical acquisition*

Android is a Linux-based OS and makes available the same tools present on all the Linux boxes. In particular the command `dd` allows to make a bit-level copy both of the internal flash memory and of the microSD. The copy is stored on a partition located on the microSD. In order to properly use the `dd` command to perform a physical copy it is necessary to open a remote shell on the device by using a PC. It is important to avoid any modification to the files present on the device to be acquired. The Android Debugger tool (ADB) allows to remotely connect to the device and perform a bit-level copy while the device is in recovery mode.

For example, to acquire a bit-level copy of the partition `/dev/mtd/mtd5` and store its image on the microSD it is possible to use:

```
# dd if=/dev/mtd/mtd5 of=/sdcard/userdata.dd bs=4096
```

### *C. Tools for the analysis*

The forensic logical and physical copies of the device can be analyzed by using commercial and open-source tools originally designed for PCs, such as Encase, FTK, Oxygen, Autopsy, PhotoREC. A useful open-source tool is Unyaffs [10] which extracts the logical structure of the file system from the Nandroid logical acquisition. Least but not last, an hex-editor can be very handy, to check for and extract digital evidence which is not caught by the other tools.

## IV. The Android Anti-Forensics Tecnique

In this section the proposed technique is described. It allows to modify and delete in a secure and selective way the digital evidence in a device with Android OS, without using cryptographic additions or low-level modifications on the kernel.

Usually, Android does not grant users access to the root of the file system. To gain access to the `mtd5` partition on the internal flash memory and to */sd-ext* on the microSD, and then modify/delete digital evidence on the device it is essential to gain root privileges. There are a few guides on the Internet on how to gain root privileges for many Android devices. MoDaCo [11] and XDA-developers [12] are among the most active communities.

The anti-forensics techniques used for magnetic supports are difficult to apply on flash memories NANDs. NANDs differ from hard drives in the technology they use to store data and in the algorithms they use to manage and access data. NANDs maintain a layer of indirection between the logical block addresses that computer systems use to access data and the raw flash addresses which identify physical storage. The layer of indirection enhances NAND performance and reliability by hiding flash memory's idiosyncratic interface and managing its limited lifetime, but it can also produce copies of the data that are invisible to the user but that a sophisticated analysis can recover [7]. These differences between NANDs and hard drives make difficult to apply secure deletion and modification techniques used for magnetic supports on NANDs.

The authors did not find in literature any paper on the secure modification of flash memories. On the other hand, a few papers address the issue of secure deletion on flash memories. In [3] and [4], cryptographic primitives have been used to make difficult to recovery digital evidence. However, the presence of cryptographic primitives is suspect in a forensics analysis. A technique which exploits the Android security features to delete digital evidence is presented in [5]. Here a third party application, at install time, creates a private folder used to contain possible subsequent digital evidence. Such evidence are deleted during uninstalling process of the application owning the private folder involved. According to [5] 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.

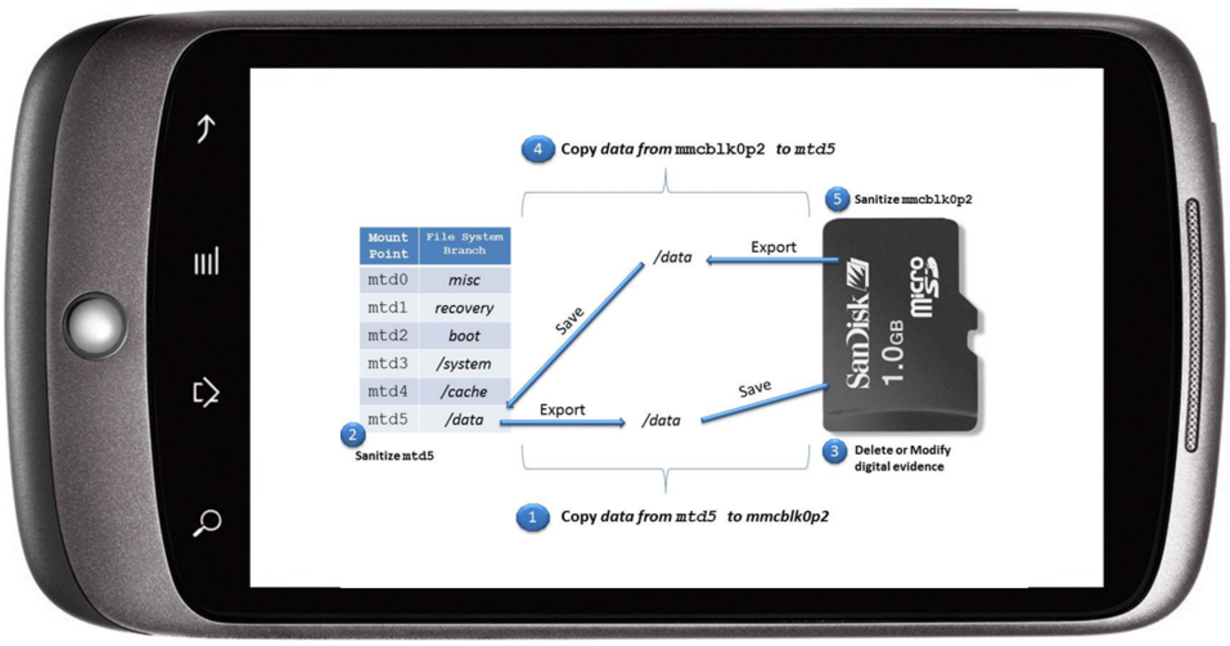
The proposed anti-forensics technique enjoys the following three characteristics:

- The use of only simple software tools which are quite commonly installed on devices running the Android OS. The use of cryptographic techniques and/or kernel modifications would have raised suspicion in a forensics analysis.
- The deletion is performed in a secure way by overwriting data. It is also possible to use different sanitization techniques, for example those described in NIST 800-88 [13].
- The modification is performed without leaving traces in the device. Also, the metadata and the attribute list (e.g., timestamp, owner, etc.) can be preserved or changed in a suitable way, so to not raise suspicion during a forensics analysis.

The modification and deletion operations are performed in such a way that previous copies of data cannot be recovered or detected.

The key idea of the technique is to reduce the selective and secure modification/deletion from a physical level on NAND to a logical level on microSD.

The microSD is used as a temporary memorization support. It holds data contained in the internal NAND to allow the subsequent sanitization of it. Afterwards it is possible to



| Mount Point | File System Branch |
|---|---|
| mtd0 | misc |
| mtd1 | recovery |
| mtd2 | boot |
| mtd3 | /system |
| mtd4 | /cache |
| mtd5 | /data |

4 Copy data from mmcblk0p2 to mtd5
5 Sanitize mmcblk0p2
/data
Export
Save
Export
/data
Save
2 Sanitize mtd5
3 Delete or Modify digital evidence
1 Copy data from mtd5 to mmcblk0p2
SanDisk
microSD
1.0GB

Fig. 1. Graphical rappresentation of the proposed technique.

selectively delete and modify data on microSD using logical commands available on the device. Finally, data on microSD is logically copied back to the internal flash memory. This copy acts only on the allocated files stored on the microSD, ignoring those present on pages marked as deleted.

Most of the Android devices collect in the branch */data* both logs generated by the OS and those generated by third party applications. However, the file system structure is not the same for all the devices. As an example, Android devices by Samsung collect such logs in the branch */dbdata*. Even though the branches of the file system containing such logs are different, they are generally located on the partition `mtd5`. Therefore, herein after the description of technique will refer to `mtd5` and not to */data* or */dbdata*. After a suitable *Setup* phase, depending on the device and on the used tools, it is possible to apply the proposed technique. Such a technique consists of five steps, as shown in Fig. 1:

1) Copy data from `mtd5` to `mmcblk0p2`: all data, including digital evidence, that has to be modified or deleted are copied to the partition `mmcblk0p2` located on the microSD. The copy is at logical level.
2) Sanitize `mtd5`: all data in the partition `mtd5` is erased using sanitization techniques.
3) Delete or modify digital evidence: the digital evidence which is located only in `mmcblk0p2`, is deleted or modified using logical operations.
4) Copy data from `mmcblk0p2` to `mtd5`: all data, including digital evidence that has been modified, are copied on the partition `mtd5`.
5) Sanitize `mmcblk0p2`: all data in the partition `mmcblk0p2` is erased using sanitization techniques.

At the end of the above five steps the partition `mtd5` contains the one and only that data which were present on pages marked as allocated on the partition `mmcblk0p2` located on the microSD. In such data there are the modified files targeted by the anti-forensic modifications. The digital evidence that was logically deleted in step 3) will not be present in the `mtd5` partition since the logical copy operation in step 4) does not copy any page marked as deleted. The partition `mmcblk0p2` will be empty after the five steps. This is not suspicious for the following reason. Starting from Android v2.2, an application ca be installed on the microSD. It is also possible to move a previously installed application from the internal memory to the microSD provided that it has been formatted with the EXT3/EXT4 file system type. Therefore, the presence of the partition `mmcblk0p2` does not raise suspicion in a digital forensic analysis. If the device is running an older version of the Android OS, it is preferable to delete such a partition since it can raise suspicion after a forensics analysis.

After the application of the proposed technique, in the partitions `mtd5` and `mmcblk0p2` there are no pages marked as deleted and containing duplicated blocks. Usually, in flash memories there are multiple copies of some blocks. Such a difference could rise suspicious after a forensics analysis. Hence, it is necessary to artfully produce deleted pages contain duplicated blocks. Since an anomaly detection analysis can distinguish the artful behaviour from the genuine one, it is advisable to generate these pages similarly to the normal Android OS and user behaviour. More generally, the content of the `mtd5` partition should have been changed to result undistinguishable by a anomaly detection analysis from a real one. However, such analysis is very difficult to perform because it should consider both the users behaviour and the Android OS functionalities.

To verify the correctness of the proposed technique one can perform a forensic investigation by first obtaining the logical and physical acquisitions and then analyzing them, as illustrated in Section III. The analysis includes searching for the digital evidence, which has been deleted by the technique. as well as the searching for traces left by the adopted technique itself. An unsuccessful search means a validation of the technique.

## V. Case Study

The focus of the case study is the application of the proposed technique to a specific device. This case study shows how the technique has been applied, and more generally gives useful advice on its application on different devices. A HTC Nexus One device equipped with the MIUI ROM has been used. The MIUI ROM is a modified Android v2.3.4 by the MIUI Team [14]. With this distribution the user gains the root privileges, the complete access to the entire file system and the chance of using the BusyBox tool. The BusyBox contains all the needed software modules required for the operations involved in the case study, such as copy, delete, modify, overwrite (i.e. the commands `cp`, `rm`, `touch`, `dd`). The presence of the BusyBox tool installed on the device does not rise suspicious since it is quite common to have it installed on an Android device. The structure of the internal flash memory, of the microSD and of the file system on the HTC Nexus One is summarized in Table I. The size column shows how big are the partitions on the internal flash memory and on the microSD.

### A. Applying the technique

The *Setup* phase includes the following two steps:

1) The reboot of the device in *recovery mode*. Indeed, the copy, delete, modify and overwrite operations cannot be executed while the OS is running.
2) The verification of the presence of the partition `mmcblk0p2` on the microSD. If this partition were not on the microSD it should have been created. This partition has to be empty, with the same file system type of the `mtd5` and with a no smaller size. For example, if the `mtd5` partition has been formatted with EXT3/EXT4 file system type, then also the `mmcblk0p2` partition has to be similarly formatted. The use of the same file system type allows to preserve metadata and data attribute list during the copy process.

| Partition | Mount Point | Size | |
|---|---|---|---|
| `/dev/mtd/mtd0` | - | - | Internal |
| `/dev/mtd/mtd1` | - | - | Internal |
| `/dev/mtd/mtd2` | - | - | Internal |
| `/dev/mtd/mtd3` | */system* | 145.0MB | Internal |
| `/dev/mtd/mtd4` | */cache* | 95.0MB | Internal |
| `/dev/mtd/mtd5` | */data* | 196.3MB | Internal |
| `/dev/block/mmcblk0p1` | */sdcard* | 458.5MB | External |
| `/dev/block/mmcblk0p2` | */sd-ext* | 3.2GB | External |

TABLE I

The description and the analysis of the anti-forensics technique, consisting of the five steps shown in Fig. 1, for the case study is reported below.

*Copy data from mtd5 to mmcblk0p2:* The logical copy of files from the partition `mtd5` to the partition `mmcblk0p2` has been implemented using the command: `# cp -a /data /sd-ext` This command performs the copy by accessing the file system, ignoring the files present on pages marked as deleted. The parameter `-a` allows the recursive copy and the preservation of the file attribute list (e.g., timestamp, metadata, etc.).

*Sanitize mtd5:* The secure deletion of the `mtd5` partition is accomplished by overwriting all pages, without considering whether they are marked allocated or deleted. The overwriting operation consist in setting each bit of the partition to "0". Afterwards the partition is formatted EXT3/EXT4 types. The commands used are the following:

```
# dd if=/dev/zero of=/dev/mtd/mtd5 bs=4096
# mke2fs -T ext4 -b 4096 -E stride=64, stripe-width=64
  -O extent,^huge_file -m 0 -L userdata /dev/mtd/mtd5
```

It is also possible to use more sophisticated media sanitization techniques, as suggested in [13], by implementing each overwriting operation with an ad-hoc `dd` command.

*Delete and modify digital evidence:* Files on the partition `mmcblk0p2` can be deleted and modified using logical functions of the BusyBox tool. To logical delete a file one can use the command `rm`. To modify files one can also use third party tools. For example, using the MMS.apk, which is a preinstalled application in the Android OS, it is possible to delete SMS sent or received. To modify the timestamp of the database in order to synchronize it with the time of the last SMS present in database, one can use the command:
`# touch -mt 201101052230.03 mmssms.db` The option `-m` allows to set the last modification time of the database mmssms.db.

*Copy data from mmcblk0p2 to mtd5:* This is the same as the step *Copy* `mtd5` *to* `mmcblk0p2` in which the input and output are exchanged. The command used is:

```
# cp -a /sd-ext /data
```

*Sanitize mmcblk0p2:* This is the same as the step *Sanitize* `mtd5` differing only on the input partition. The commands used are the following:

```
# dd if=/dev/zero of=/dev/block/mmcblk0p2 bs=4096
# mke2fs -T ext4 -b 4096 -E stride=64, stripe-width=64
  -O extent,^huge_file -m 0 -L userdata /dev/block/mmcblk0p2
```

After the five steps, to avoid the lack of duplicated pages in the `mtd5` partition, the authors have performed the following operations:

- send and receive emails;
- surfed on the Internet (this generate many cache files);
- send and receive SMS;
- originate some calls.

### *B. Technique Validation*

The technique described and implemented in the case study has been validated by running multiple experiments involving many modifications and/or deletions. Then by performing both physical and logical acquisition and finally a forensic analysis on them. During the verification phase all the open-source tools mentioned in Section III together with the demo version of Oxygen Forensics Suite 2011 have been used.

In order to check the effectiveness of the file deletion operation, several files containing predetermined short patterns have been created on the involved flash memories. After the application of the technique, the presence of such patterns has been searched for. To perform the search of the patterns, the forensic copies of the internal and external flash memories have been transferred to an external PC and inspected with an hex-editor. In addition, to further test the correctness of the deletion of the digital evidence, digital forensics open-source tools (Autopsy and PhotoREC) have been used.

In order to validate the anti-forensics technique used to modify files on the device, it has been applied several times on different files. All the modifications have been performed on the device without requiring any external hardware. Some modification were made using programs shipped with the equipment while others by using third party applications. Some of these tools have been mentioned in Section V. Most of the performed modifications have been focused on system applications such as logs, SMS, calls, contacts, emails, etc.. After having applied the modifications, all the involved files and applications have been checked in order to verify whether the changes have been applied or not and the timestamps consistency. Moreover, keeping in mind that flash memories may keep duplications of some parts of data, it has been checked whether any duplication of the modified information, referring to an unmodified version of it, was found on the device.

All forensics analysis performed have confirmed the validity of the described technique.

## VI. Conclusions and Future Works

A new anti-forensics technique for the Android OS has been proposed and analyzed. By using the latest digital forensics tools, and considering the specific characteristic of the flash memories used by the Android devices, the technique has been validated showing its effectiveness in a case study. Whereas in the case study the device adopted has been a specific mobile phone running the Android OS, the authors plan to further investigate their technique even on different kind of Android devices (tablet, TV, netbook). Moreover, considering that the type of memories used by the majority of the Android devices are the same used by most of all the other mobile devices, a future study could be performed on devices using similar types of memories (e.g., the Apple iPhone).

It could be interesting to characterize the distribution of the duplicated pages in order to simulate it and survive to any eventual anomaly detection inspection.

## References

[1] Gartner Inc. and/or its Affiliates, "Gartner Analysts," http://www.gartner.com/technology/analysts.jsp, 2011 (accessed May 09, 2011).

[2] R. Harris, "Arriving at an anti-forensics consensus: Examining how to define and control the anti-forensics problem," in *The 6th Annual Digital Forensic Research Workshop (DFRWS 2006)*, Aug. 2006. [Online]. Available: http://dfrws.org/2006/proceedings/6-Harris.pdf

[3] D. W. Byunghee Lee, Kyungho Son and S. Kim, "Secure data deletion for usb flash memory," *Journal of Information Science and Engineering*, pp. 1710–1714, 2011.

[4] J. Lee, J. Heo, Y. Cho, J. Hong, and S. Y. Shin, "Secure deletion for nand flash file system," in *ACM Symposium on Applied Computing*, 2008, pp. 1710–1714.

[5] A. Distefano, G. Me, and F. Pace, "Android anti-forensics through a local paradigm," *Digital Investigation*, vol. 7, pp. S83–S94, Aug. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2010.05.011

[6] Charles Manning, "YAFFS: the NAND-specific flash file system - Introductory Article," http://www.yaffs.net/yaffs-nand-specific-flash-file-system-introductory-article, 2011 (accessed May 27, 2011).

[7] M. Y. C. Wei, L. M. Grupp, F. E. Spada, and S. Swanson, "Reliably erasing data from flash-based solid state drives," in *FAST*, G. R. Ganger and J. Wilkes, Eds. USENIX, 2011, pp. 105–117.

[8] J. Lessard and G. C. Kessler, "Android forensics: Simplifying cell phone examinations," *Small scale digital device forensics journal*, Aug. 2010.

[9] viaForensics, "viaForensics' AFLogical Tool is Best for Android Forensic Investigations," http://viaforensics.com/viaforensics-articles/viaforensics-aflogical-tool-android-forensic-investigations.html, 2011 (accessed June 22, 2011).

[10] Kai.Wei.cn, "Unyaffs project," http://code.google.com/p/unyaffs/, 2011 (accessed May 27, 2011).

[11] P. O'Brien, "Android @ modaco.com," http://android.modaco.com/, 2011 (accessed May 27, 2011).

[12] XDAdevelopers, "Android forum & windows phone discussion," http://forum.xda-developers.com/index.php, 2011 (accessed May 27, 2011).

[13] NIST, "NIST 800-88, Guidelines for Media Sanitization," http://csrc.nist.gov/publications/nistpubs/800-88/NISTSP800-88_rev1.pdf, 2011 (accessed May 27, 2011).

[14] MIUI Team, "MIUI Android ROM," http://www.miui.com, 2011 (accessed June 20, 2011).