Mobile Forensics Data Integrity Assessment by Event Monitoring

Alessandro Distefano, Antonio Grillo, Alessandro Lentini, Gianluigi Me, Davide Tulimiero
Computer Science, Systems and Production Department
University of Rome Tor Vergata, Via del Politecnico 1
00133, Rome, Italy
{distefano,grillo,lentini,me,tulimiero}@disp.uniroma2.it

***Abstract*—Mobile phones, due to the wide range of services they offer, increasingly represent a major tool, with crucial importance, in people daily life. Therefore, they could act as fundamental witnesses, or simply as sources to angle for, supporting investigations of a wide range of crimes, not limited to digital crime. Currently, the most forensic acquisition and analysis of these devices is based on tools implementing a remote paradigm, dealing with a forensic workstation used for seizure, via insertion of a piece of code in the mobile device; therefore, the characterization of integrity respect is still difficult and needs to be investigated in depth. In this paper, we present a new approach to assess the integrity respect regarding both the acquisition tools used and the occurring events. Experimental results show the suitability of the proposed strategy in order to characterize a full investigation workflow as well.**

***Index Terms*—Mobile Forensics, Integrity Respect, Forensic**

I. INTODUCTION

Mobile Phones (MPs) currently represent one of the most diffused technologies all over the world [1]; the number of subscribers overcame 2 billions of units and this number seems to follow an increasing trend. Hence, in the last four years, an interesting growth was registered in areas such as India, China and Latin America. The current capabilities of the last-generation MPs are extremely interesting from the forensic perspective [6], due to the large use of additional data services. In fact, MPs are equipped with a set of interfaces allowing both long-range (e.g., GSM, UMTS), and short-range (e.g., Bluetooth, WiFi) communications. Regarding the computational capabilities, they resemble general purpose computers more than just phones: in fact, applications with a wide set of functionalities (e.g., GPS, e-mail, chat, office automation, multimedia management) are provided in-bundle in many *off-the-shelf* MPs. Therefore, the use of these functionalities drives the quantity and quality of personal information stored in the MPs. In fact, since that information deal with all the activities related to the use of the MP, the data stored by a MP squarely describe habits and behaviors of its owner.

In this scenario, the number of MPs implicated in crime activities is relevant and it is growing up [9]. Hence, as MPs become a widely used tool even in the illegality, the ability to perform mobile forensic analysis to achieve results in investigation become every day more important. In [8], some issues about forensic data acquisition for MPs are outlined, while in [5] a more extended discussion has been provided.

In this paper, we present a new strategy to perform a stronger assessment of the integrity respect than the approach used in [8]. This approach uses an application which has been specifically designed to capture and to identify any change occurred in the MP file system, with special care to the internal memory. In this way, it is possible to correctly identify every modification and the related operating entity.

This paper is structured as follows. Section II introduces the importance of the integrity respect assessment in forensic fields, some common strategies are briefly described with relation to problems when applied to mobile devices. Section III briefly describes a new local paradigm to acquire data from internal memory of MPs, while Section IV describes the role that such paradigm can play regarding forensic properties assessment as well. Section V shows both the experiments and the related results and Section VI presents an example of a complete workflow coherent with the proposed approach.

II. STATE OF THE ART

A. *Background on integrity assessment*

As we are interested in forensic tools designed to acquire data from MPs, a rigorous method to assess the respect of some fundamental properties is mandatory: the required guarantees in the forensic environment are strict [7]. In particular, in order to avoid the evidence poisoning, the former requirement is represented by the (best) respect of the integrity of the stored data.

Although only in May 2007, the NIST published a document [6] covering the specific field of forensic data acquisition from MPs, a lot of effort was previously made in order to create a set of rules and guidelines to describe good practices for their management during the investigations. In fact, due to the internal memory inaccessibility, the classical computer forensics rules and guidelines cannot be merely shifted to mobile devices; probably, the main inhibitor to face is represented by the unavailability/heterogeneity of the direct access to all the storage volumes [5]. However, if these rules and guidelines can help in the filesystem corruption prevention, they are definitively weak when characterizing and studying this phenomenon. A further valuable aspect regarding mobile devices relies on data integrity, which could be threatened both by occurring events (e.g., incoming calls, incoming messages, alarms, reminders) (e.g. due to human errors in well-known procedures), and by operations made both by forensic operators and by forensic tools. All the aforementioned remarks seem to suggest that a complete respect of integrity, during forensic analysis of a MP using the current methodologies based on software logical acquisition, is quite unrealistic. Furthermore, the analysis workflow heavily depends on the initial state of the device; in such scenario, a minimum level of corruption seems impossible to be avoided. Hence, regarding mobile forensics, the current goal is to minimize the degree of corruption of the stored data; the basic proposed approach is to deeply analyze the corruption in order to isolate the original information from the poisoned one, with the intention to use in courts only the original data. This idea can lead to the definition of two regions of the acquired data, which can be identified, even in general, by the use of extended experimentations to mitigate the *per device peculiarities*. Furthermore, since the extraction of a raw image of data stored acting as reference is often unfeasible, an investigation to state the degree of corruption represents a relevant step to identify the acquisition reliability related to the integrity requirement.

Currently, as the validation of forensic tools is a complicated and expensive task, a lot of manufacturers seem to be more likely interested in functionalities rather than strong validation of their forensic tools [5].

Finally, although some authoritative analyses and reports have been performed by the US National Institute of Standards and Technology [12], a set of standard procedures for mobile forensic tools validation, currently, is still under construction.

B. Integrity assessment via code reading

This first strategy is widely used to perform generic software testing using Code Reading sessions; this technique can be viable in order to validate forensic tools as well, providing a fine grained analysis of the behavior of the tool. However, this approach presents three major issues:

1) The tool must be Open Source: for other tools, as the source code is not available, it is impossible to perform a code inspection [7];
2) The whole operational environment must be Open Source: actually, as the acquisition tool is just a part of the entire equipment used in acquisition, it is important to extend the code inspection to the entire environment;
3) The conclusions are only based on a static comprehension of the tools behaviour and they cannot take account of dynamic evolution of the acquisition task.

Unfortunately, regarding Mobile Forensics, this approach is extremely complex to apply and less used in practice, because the entire operational environment rarely is completely Open Source. Hence, the inspection is severely limited or can be performed on a reduced set of instruments; such approach, when feasible, is quite always combined with experimentation in order to draw more reliable conclusions.

C. Integrity assessment via experimentation

This second strategy performs the assessment following an orthogonal direction: only the dynamic evolution of the tool and the results are used. This way, more frequently used in practice, is based on the capability to extract a reference image of the data to be preserved. A second image of this data is made using the tool to be evaluated and then a cross-check is performed between the two images, in order to identify the differences. This approach presents many practical benefits with respect to the former, but the validity and generalization of conclusions could be threatened by the reduced sample size and its representativeness. In fact, most often, even the commercial forensic tools are tested over a restricted number of devices; furthermore, the access to devices that have been seized in real investigations is limited by law.

Furthermore, regarding mobile devices, it is not always possible to easily get the required reference image and, in many cases, the only tool able to get a reference image is the tool being investigated or a similar one; in this situation it is not trivial to state what kind and degree of corruption is caused by the acquisition tool or by additional events.

III. THE MIAT TOOL

The work described in [2], [4], [8] presents a new methodology to acquire data from smartphones, based on a new local paradigm dealing with a local connection established between the mobile device to be acquired, which become the forensic workstation, and its internal memory file system. The forensic tool underlying the new methodology (namely, MIAT - Mobile Internal Acquisition Tool) is a software application which can be installed directly on the smartphone using a removable memory card as an installation volume. During the execution, MIAT mirrors the internal memory file system into the removable memory card; in such a way, the whole forensic equipment required to perform an acquisition is a reduced set of memory cards.

A. MIAT forensic properties assessment

As MIAT aims to be a forensic tool, the assessment of its forensic properties is required. In [8], an overall assessment has been provided both for performances and for forensic properties together with a comparison between MIAT and the Paraben Device Seizure [13] commercial tool. Experiments show that both these forensic tools modify a reduced set of files; this set is composed by OS files, whose utility can be considered marginal in a forensic investigation. However, the approach presented in [8] has not the granularity required to assess if the corruption is due to the acquisition process or to another cause, because the compared images are collected using the same forensic tools which are under study.

For this reason, this paper aims at presenting a stronger analysis of the phenomenon of corruption; the new strategy allows the separation between the reference image creation and the acquisition performed with the tool being investigated. Furthermore, this strategy allows to easily investigate the corruption due to some occurring events such as incoming calls, text messages and installation of applications.

IV. A STRONGER STRATEGY

The strategy presented in this paper is based on a monitoring application called FSMon which has been designed and implemented exploiting the Symbian OS native APIs. This application is able to perform a fast image of the entire logical structure of the internal memory and to be notified of calls to the file system, in order to detect any modification which could occur.

Regarding the notification feature, the strong relation between the FSMon application and the Symbian OS can be considered as a drawback in terms of portability, however it is required in order to realize that feature. In fact, at the time of writing, the technologies supporting portable applications cannot provide a proper support to the realization of features which are strongly related to services provided by the OS. At the same time, the realization of the system call interception feature for mobile OSs, which is well known for conventional OSs, is interesting for a large number of applications (e.g., proactive malware detection [10], sandboxes [11]) and also other mobile OSs (e.g., Windows CE [3]).

A. The FSMon Application

This common application can be notified directly by the Symbian OS File Server when a configurable event occurs; regarding the objectives of this work, the events of interest are write operations, files creation and deletion when they happens on the internal memory file system. FSMon can be executed in the two different modes:

1) Notification Waiting: when FSMon is spawned in Notification Waiting mode it uses an endless loop. Each iteration waits for a notification and, as it occurs, collects last modification time data for each entry currently present in the file system. Among the collected data, FSMon searches for the entries which have been modified, created or deleted within a restricted time range across the notification time, in order to identify the file system entries affected by the last notification. The *Figure 1* shows an example of a possible notification time interval of 2 seconds. This check is suitable only for closed files: at notification time, if a selected file is still open, a different schema must be used. In this second case, the current effective file size is compared with the previous one; if a mismatch occurs, the file is confirmed. Experiments show that in the most frequent case, each notification identifies a single file entry affected. This identification schema is required because the target Symbian OS version does not offer a service to directly identify the last-modified entry in the FS.

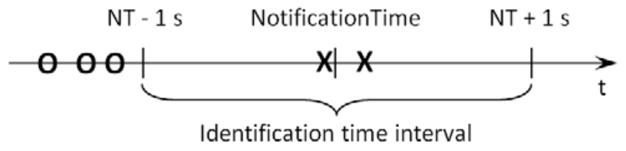

Figure 1: Entry identification example. The entries with lastmodification time within the identification time interval (**X**) are selected, the others (**O**) are discarded.

2) File System Structure Imaging: when FSMon is spawned in File System Structure Imaging mode, it performs a fast image of the file system tree, including last modification time for each file entry. This second mode is useful to investigate the corruptions due to event that FSMon cannot monitor (e.g., a device reboot).

In both modes, the results are stored into the same Removable Memory Card (RMC) used to store the FSMon executable, using a simple text file. In Notification Waiting mode, this text file contains the file entries affected by each managed notification, while in File System Structure Imaging mode , the file contains the collected image.

B. FSMon Implementation

From an implementation perspective, FSMon is a Symbian application realized using the native API offered by the OS. FSMon has two fundamental requirements: the integrity respect and the capability to manage all the raised notifications. Regarding the first requirement, we will show how each connection established between FSMon and the Symbian File Server is intended to respect the integrity. Regarding the second requirement, FSMon is implemented to quickly manage each notification; in this way, we minimize the probability of notifications lost. The notification management behavior is fully implemented by an `ActiveObject`: a read-only connection is established with the Symbian File Server, which supplies the notification service via the `NotifyChange` method in `RFs` class. This method is fully customizable, both for events to be notified and volumes to be monitored. The file system imaging behavior is implemented by a simple iterative scanning task: for each file system entry, a snapshot of its attributes is collected. In order to build such image, a read-only connection to Symbian File Server is established and each snapshot of file attributes is collected without opening the interested file.

FSMon must be able to perform write operations onto RMC volume, in order to store the results; however, this capability does not threat the internal memory integrity.

V. EXPERIMENTS

This section describes the definition of the experiments (Section V-A) and the related results (Section V-B).

A. Experiments definition

1) *Device state control:* First, we need to control the state of the used mobile device; the degree of control we need must ensure that each experiment starts and runs under given and constant conditions. The state of the device is controlled using four countermeasures:

- The device is hard-formatted before each experiment, in order to ensure the experiments work on the same file system image;
- The device is always booted in recovery mode, in order to ensure that only crucial tasks are executing after the device boot, while the execution of other tasks is

inhibited; this prevents corruption due to some local events (e.g., autorun applications);

- The device is booted in offline mode, removing the SIM card from its slot, in order to ensure the isolation between the mobile device and the communication networks; therefore, the environmental events (e.g., incoming calls or text messages) cannot corrupt the stored data.
- Battery charge level is at least 50% and the charger is not plugged, used to prevent the battery discharge during the experiments, which could generate unknown events.

Although the device state, using the countermeasures shown above, is not a typical operational state, it ensures that each experiment runs under the same conditions. Hence, during each experiment we have the same device state, the same file system structure and, finally, the same running processes list. Furthermore, in order to prevent any other source of corruption, we disabled the short-range connections (e.g., Bluetooth) to avoid any changes due to data transmission from/to other devices.

2) *Subject of the experiments:* All the experiments were performed on the same device, in this case a Nokia N70; this device is equipped with Symbian OS v8.1a.

3) *Treatments:* The treatments used during the experiments can be grouped in two several sets: Forensic tools (discussed in Section V-B1) and Device events (discussed in Section V-B2). Regarding the first set, we used both MIAT-S60 [8] and Paraben Device Seizure v1.3.2824.32812 [13]. Regarding the second set, we selected the following interesting events:

- Device reboot;
- SIM card removal;
- MIAT installation;
- RMC extraction and insertion;
- Environment related events;
- Events related to data acquisition.

4) *Replication:* Replication is one of the most crucial aspects in forensic experimentation; generally, the replication of experiments is strongly related to the validity and the generalization of the experimental results. However, when applied to forensics, experimentation has to face with legal issues and limitations. Often, the most representative devices are those seized in real investigations, but this kind of devices are not freely available. In our experiments, we used a single device; however, this item does not undermine the conclusions drawn because our objective is simply to characterize the behaviour of the tested forensic tools and of the occurring events. Obviously, only a more extended experimentation, including a wide set of different device models, can lead to more general conclusions.

5) *Experimental workflows:* In order to perform a right description of the performed experiments, we defined the following three workflows.

a) Characterization of forensic tool behaviour: Aiming at investigating the dynamic evolution of forensic tool execution. The measures are collected by FSMon and a cross-check with data acquired is performed. The workflow steps are the following:

1) Device format;
2) FSMon startup in mode 1;
3) Forensic tool execution;
4) FSMon block.

b) Characterization of simple events: Aiming at investigating the file system corruption due to the occurrence of events that FSMon is able to monitor. The workflow steps are the following:

1) Device format;
2) FSMon startup in mode 1;
3) Forensic tool execution;
4) FSMon block

c) Characterization of complex events: This kind of experiments has the same objective of the previous but some events (e.g., device reboot, SIM card removal) cannot be monitored by FSMon. In this case, the workflow steps are the following:

5) Device format;
6) File System structure image using FSMon in mode 2;
7) Event occurrence;
8) File System structure image using FSMon in mode 2.

6) *Corruption verification*: In Section IV-A, we stated that if a last-modification time change is discovered, a corruption is detected; however, the experiments shown that some files, although storing the same content, have different lastmodification times. This phenomenon could be motivated in several ways. It is possible that a sequence of operations does not alter the whole file content, it is possible that a file is fully rewritten; in these cases, the content does not change but the last-modification time does. Our approach to solve this problem is to perform a cross-check between the additional information collected by MIAT and the changes notified to FSMon; this check is performed using the MD5 hash-code in order to effectively identify the differences in the content of files. Although, the phenomenon of collisions in hash-codes is always possible, we consider the probability of such event as negligible; furthermore, as a reduced set of files seems to be modified by MIAT, we have not used these files in our cross-check.

B. Experimental results

The results are grouped according to the used treatment regarding the classification made in Section V-A3. Each experiment is summarized using a table, whose format is the same for each experiment; for practical reasons the name and path of the files are omitted. The first column shows the monitored operations, the second column shows the results obtained using only FSMon (FSMon), the third column shows, if applicable, the cross-check results (CC). Each inner cell contains two numbers separated by a colon: the first represents the number of generic files which has been changed, while the second represents the number of user-data files among generic ones which has been changed. Further tables are used to clarify some particular aspects.

1) *Forensic tools:*

a) MIAT acquisition: The workflow described in Section V-A5a was applied twice, in order to collect both write operations and files creation/deletion. As TABLE I (a) shows, FSMon reports 3 files modified during the acquisition process; however, two files of those are modified only after their acquisition is completed. In other words, the MIAT images of these two files contain the same data stored by the original files before the acquisition. TABLE I (b) shows time measures to ensure that the modification of these files occurs only after their acquisition. The third file is modified only right after a device format, while further MIAT acquisitions do not change that file.

TABLE I: MIAT Acquisition Experimental Results.

(a) Detected Corruption

Operation	FSMon	CC
Write	3:0	-:-
Creation	0:0	-:-
Deletion	0:0	-:-

(b) Interesting Times

Time	`CntModel.ini`	`nssvasdatabase.db`
Last-Modification	18:31:46	18:39:44
Acquisition begin	18:31:33	18:35:17
Acquisition end	18:31:33	18:35:17

TABLE II: Paraben Acquisition Experimental Results.

Operation	FSMon	CC
Write	8:0	5:0
Creation	1:0	1:0
Deletion	1:0	1:0

b) Paraben acquisition: The workflow described in Section V-A5a was applied twice, in order to collect both write operations and files creation/deletion. TABLE II summarizes the experimental results regarding Paraben acquisition. Furthermore, our experiments confirm what stated in [8]: Paraben misses the `‘‘Private’’` and `‘‘_PalbTN’’` folders.

2) *Device* events:

a) Device reboot: The workflow described in V-A5c was applied once. TABLE III (a) summarizes the results regarding the first reboot right after a device format, while TABLE III (b) describes further reboots. Comparing these two tables, we can state that the degree of corruption due to a device reboot is affected by a previous format process.

TABLE III: Device Reboot Experimental Results.

(a) First Reboot

Operation	FSMon	CC
Write	16:1	8:1
Creation	1:0	1:0
Deletion	0:0	0:0

(b) Further Reboot

Operation	FSMon	CC
Write	14:0	4:0
Creation	0:0	0:0
Deletion	0:0	0:0

TABLE IV: Sim Card Removal Experimental Results.

(a) First Removal

Operation	FSMon	CC
Write	21:1	8:1
Creation	2:0	2:0
Deletion	0:0	0:0

(b) Further Removal

Operation	FSMon	CC
Write	16:0	6:0
Creation	0:0	0:0
Deletion	0:0	0:0

b) SIM card removal: The workflow described in V-A5c was applied once. TABLE IV (a) summarizes the results of the first SIM card removal right after a device format, while TABLE IV (b) describes further SIM removals. Comparing these two tables, we can state that the degree of corruption due to a SIM removal is affected by a previous format process.

TABLE V: MIAT Installation Experimental Results.

(a) First Installation.

Operation	FSMon	CC
Write	3:0	-:-
Creation	2:0	-:-
Deletion	1:0	-:-

(b) Further Installations.

Operation	FSMon	CC
Write	3:0	-:-
Creation	1:0	-:-
Deletion	1:0	-:-

c) MIAT installation: In order to acquire data with MIAT, the operator should proceed, firstly, with installation; hence, we need to investigate the degree of corruption caused by this action. The workflow described in V-A5b was applied twice, in order to collect both write operations and files creation/deletion. In this case, we cannot apply the cross-check because MIAT cannot be used without being installed. TABLE V (a) summarizes the results right after a device format, while TABLE V (b) describes further installations; the additional file, which has been created during the first installation, is the log of the installed applications.

d) RMC extraction/insertion: In order to acquire data with MIAT, a memory card replacement is required; hence we need to investigate the degree of corruption caused by this action. The workflow described in V-A5b was applied twice, in order to collect both write operations and files creation/deletion. TABLE VI summarizes the experimental results related to this event. Our experiments state that a complete replacement implies the same corruption of a simple extraction or a simple insertion; it is worth noticing that the corruption due to this event can depend on the data stored by the RMC (e.g., backups of the phonebook).

TABLE VI: RMC Replacement Experimental Results.

Operation	**FSMon**	**CC**
Write	3:0	-:-
Creation	0:0	-:-
Deletion	0:0	-:-

e) Environment related events: This part exposes the results related to some interesting environmental events; for each of those, we applied twicethe workflow described in V-A5b, in order to collect both write operations and files creation/deletion. In particular, we focused on the following three events:

- Ingoing/outgoing Voice Call. TABLE VII (a) summarizes the related results. As expected, the detected write operation affects the log-file of the local events.
- Ingoing text message. TABLE VII (b) summarizes the related results. The detected file creation refers to a new file which is created for each received message and contains the text just received, while the write operation affects the index file for the stored messages.
- Ingoing E-Mail message. TABLE VII (c) summarizes the related results. The detected file creation refers to a new file which is created for each new message and contains the data just received, while the write operations affect the log-file of the local events (as the E-Mail was retrieved through a data call) and the index file for the stored messages.

f) Events related to data acquisition: This part exposes the results related to some events that can be strictly related to data acquisition; for each of those, we applied the workflow described in V-A5b twice, in order to collect both write operations and files creation/deletion. In particular, we focused on the following three events:

- Online/Offline mode software switch. In order to acquire data using MIAT, the phone should be isolated from the network; this can be achieved by a software switch.
- Battery charger plugging. The data acquisition can require a lot of time regardless of the used tools. In order to avoid this process to be stopped by low power, the device could be connected to the battery charger.
- USB cable plugging. In order to acquire data from a phone it could be connected to a forensic workstation by USB cables.

The experiments show that no files have been modified, nor created, nor deleted by none of the above events.

TABLE VII: Experimental Results Related to Environment Related Threats.

(a) Voice Call.

Operation	**FSMon**	**CC**
Write	3:0	-:-
Creation	0:0	-:-
Deletion	0:0	-:-

(b) Ingoing Text Message.

Operation	**FSMon**	**CC**
Write	3:0	-:-
Creation	0:0	-:-
Deletion	0:0	-:-

(c) Ingoing E-Mail.

Operation	**FSMon**	**CC**
Write	3:0	-:-
Creation	0:0	-:-
Deletion	0:0	-:-

VI. MIAT ACQUISITION WORKFLOW

The entire MIAT acquisition workflow can be characterized, from the perspective of the integrity respect, using the experimental results presented in Section V-B; our goal is to provide a description of how events can interoperate in each possible flow of events. As the state of the device to be acquired and the crime scene equipment cannot be known a-priori, we grouped all the possible flows in a single flow-chart; the Figure 2 shows such chart where each path from start to stop blocks represents a complete operational flow. The expected corruption of an entire flow can be obtained computing, for each dimension, the set-theory union over the events in the path.

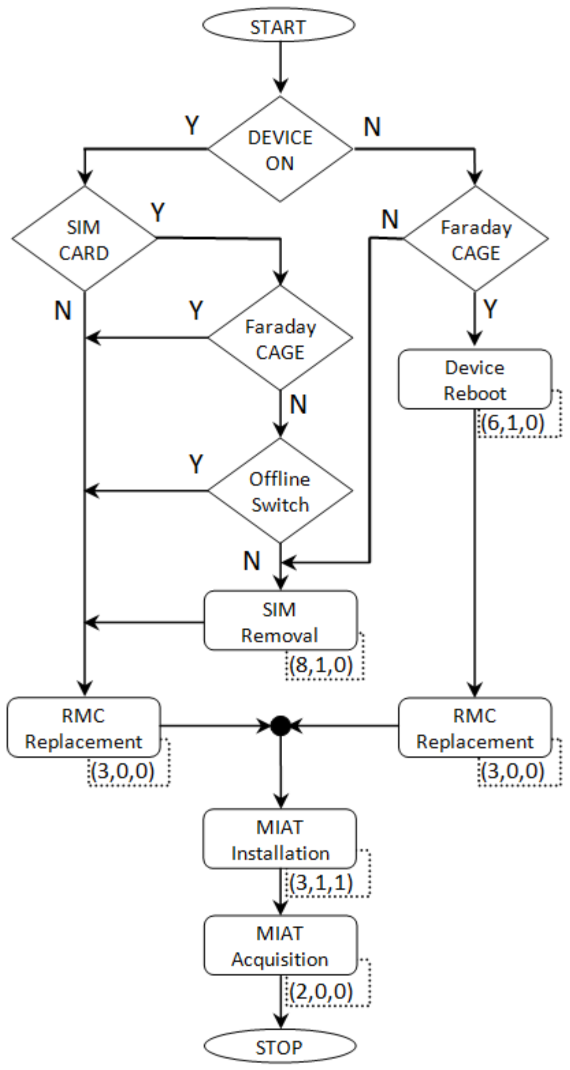

Figure 2: Corruption evaluation for the whole MIAT acquisition workflow. Each functional step is described by a triple whose dimensions, respectively, measure the number of files modified, created and deleted during the step.

VII. CONCLUSIONS

Strong assessment of properties for mobile forensics tools is a difficult task and, often, a rarely performed step during the design and the implementation of some tools. However, Open Source software and Open Source communities encourage the deep testing and assessment of general purpose applications; furthermore, by source code examination, a number of conclusions can be drawn. These advantages should be exploited especially for the assessment of forensic tools and to provide fairness between prosecutor and defense in the legal environment. At the same time, regarding mobile devices the corruption of stored data seems hard to be totally avoided; a current and realistic goal could be both to minimize and to characterize the corruption during the entire investigation workflow.

In this paper, we presented a new strategy to characterize the internal memory file system corruption due both to forensic tools and occurring events; our strategy is applicable to Open Source and to commercial tools as well. Further experimental results allowed performing a characterization of the entire MIAT collection workflow; such characterization could guide the forensic operators throughout the management of devices discovered in crime scenes, in order to minimize the degree of corruption.

REFERENCES

[1] Kalba, K.: The adoption of mobile phones in emerging markets: global diffusion and the rural challenge. International Journal of Communication vol. 2, pp. 631–661 (2008).

[2] Me, G., Rossi, M.: Internal forensic acquisition for mobile equipments. In: 4th International Workshop on Security in Systems and Networks, Proceedings of the International Parallel and Distributed Processing Symposium (2008).

[3] Becher, M., Hund, R.: Kernel-Level Interception and Applications on Mobile Devices. (2008).

[4] Dellutri, F., Ottaviani, V., Me, G.: MIAT-WM5: forensic acquisition for Windows mobile PocketPC. In: 2008 Workshop on Security and High Performance Computing Systems, part of HPCS (2008).

[5] Jansen, W., Delaitre, A., Moenner, L.: Overcoming Impediments to Cell Phone Forensics. In: 41th Annual Hawaii International Conference on System Sciences, part of HPCS (2008).

[6] Jansen, W., Ayers, R.: Guidelines on cell phone forensics recommendations of the National Institute of Standards and Technology. (2007).

[7] Carrier, B.: Open source digital forensics tools the legal argument. (2003).

[8] Distefano, A., Me, G.: An Overall Assessment of Mobile Internal Acquisition Tool. Journal of Digital Investigation vol. 5, pp. S121–S127 (2008).

[9] Williamson, B., Apledoorn, P., Cheam, B., McDonald, M.: Mobile forensics turns up heat on suspects. http://www.theregister.co.uk/2007/02/11/mobile forensics guidance/ (2007).

[10] Becher, M., Freiling, F.C.: Towards Dynamic Malware Analysis to Increase Mobile Device Security. In: SICHERHEIT (2008).

[11] Willems, C., Holz, T., Freiling, F.: Toward Automated Dynamic Malware Analysis Using CWSandbox. IEEE Security and Privacy, vol. 5, issue 2, pp. 32-39, (2007).

[12] National Institute of Standards and Technology - Information Technology Laboratory: Computer Forensics Tool Testing Program. http://www.cftt.nist.gov/mobile devices.htm

[13] Paraben Corporation: Paraben's Forensics Software. www.paraben.com