Evaluation Framework for Detecting Manipulated Smartphone Data

Heloise Pieterse, Martin Olivier, and Renier van Heerden

***Abstract*—Ever improving technology allows smartphones to become an integral part of people's lives. The reliance on and ubiquitous use of smartphones render these devices rich sources of data. This data becomes increasingly important when smartphones are linked to criminal or corporate investigations. To erase data and mislead digital forensic investigations, end-users can manipulate the data and change recorded events. This paper investigates the effects of manipulating smartphone data on both the Google Android and Apple iOS platforms. The deployed steps leads to the formulation of a generic process for smartphone data manipulation. To assist digital forensic professionals with the detection of such manipulated smartphone data, this paper introduces an evaluation framework for detecting manipulated smartphone data. The framework uses key traces left behind as a result of the manipulation of smartphone data to construct techniques to detect the changed data. Assessment of the evaluation framework involves three distinct theoretical scenarios that involve the deletion and modification of existing data, as well as a failed attempt to insert fabricated data. The results produced by the evaluation framework suggest the framework can assist with the detection of manipulated smartphone data. The purpose of this research study was to demonstrate the manipulation of smartphone data and present an evaluation framework to detect such manipulated data.**

***Index Terms*—Digital forensics, mobile forensics, manipulation, smartphone data, smartphones, Android, iOS.**

I. Introduction

The 21st century is witnessing the rapid development of smartphone technology. The current technological advancements equip smartphones with improved capabilities and functionality that nowadays closely resemble a personal computer. Existing smartphone models support different connectivity options, various communication channels, the installation of third-party applications, as well as a complete operating system. The leading smartphone operating systems of 2018 are Google Android and Apple iOS. The prominence of both Android (70.09% market share) and iOS (28.50% market share) platforms directly relates to their provided capabilities and popularity among users [1]. Although other smartphone operating systems do exist, the combined market share of 98.59% guided this study to only focus on these platforms.

From a mobile forensics' perspective, which forms a subdiscipline of digital forensics, the data collected by smartphones, called smartphone data, can become important sources of digital evidence. Smartphone data includes any data of probative value that is generated by an application or transferred to the smartphone by the user [2]. The extensive market share of both Android and iOS smartphones ensures the diverse usage of the devices, which eventually leads to rich collections of smartphone data [3]. Smartphone data describe events (for example sending a text message or browsing a website) that occurred on the smartphone. Valuable smartphone data, such as contacts, text messages, call lists, browsing history and e-mails, provides a well-defined snapshot of user events and support the chronological ordering of these events [4]. The exact events recorded by a smartphone depend on several internal and external factors, such as smartphone settings, operation by the user and installed applications [5]. Regardless of the availability, the produced smartphone data can still offer insight during digital forensic investigations and provide important digital evidence.

The value of smartphone data as a form of digital evidence has, however, raised suspicion among users. Data retrieved from smartphones can offer contextual clues about the end-user, who the owner and user of the smartphone is, as well as activities performed involving the smartphone. Such clues can reveal who the user knows and communicated with, locations visisted, highlight personality traits and pinpoint close associates [6]. The presence of such information can be a cause for concern [7], which can drive end-users to apply manipulative techniques to the data and eliminate or remove any potential value. The motivation for manipulating smartphone data is two-fold. Firstly, benign end-users can deploy certain techniques to manipulate smartphone data deemed private or sensitive and minimise the exposure of such data. Secondly, end-users can use similar techniques to intentionally make changes to smartphone data to hide their involvement in criminal activities and erase incriminating events. These techniques and tools are commonly referred to as anti-forensics and are primarily used to "compromise the availability or usefulness of evidence to the forensic process" [8]. Several recent research studies ([4], [9], [10], [11], [12], [13]) have investigated the effect and feasible use of anti-forensics in the smartphone environment. The first study [4] explored the possibility to create a false digital alibi on a smartphone and thwart investigations. The remaining studies focused on introducing new anti-forensic techniques to modify and erase digital evidence, manipulate existing data or thwart marking-leading digital forensic tools. More specifically, one research study investigated the viability of modifying the operating system (CyanogenMod) in an anti-forensics context to prevent data extractions, present false data and impede

Manuscript received November 1, 2018.

H. Pieterse is with the Defence, Peace, Safety and Security, Council for Scientific and Industrial Research and Department of Computer Science, University of Pretoria, Pretoria, South Africa, e-mail: hpieterse@csir.co.za.

M. Olivier is with the Department of Computer Science, University of Pretoria, Pretoria, South Africa, e-mail: molivier@cs.up.ac.za.

R. van Heerden is with the National Integrated Cyber Infrastructure System, Council for Scientific and Industrial Research, Pretoria, South Africa and the School of Information and Communication Technology, Nelson Mandela University, Port Elizabeth, South Africa, e-mail: rvheerden@csir.co.za.

Based on: *"Detecting Manipulated Smartphone Data on Android and iOS Devices"*, by H. Pieterse, M. Olivier, and R. van Heerden which appeared in Communications in Computer and Information Science (volume 973). © 2019 Springer International Publishing

digital forensic tools [12]. It is, therefore, possible for end-users to utilise anti-forensics to erase, manipulate or construct false data, ultimately misleading digital forensic investigations.

To counter and thwart the effects of anti-forensics, existing research presents several solutions. Verma et al. [14] preserve date and timestamps of Android smartphones by capturing system generated values and storing these values in a location beyond the smartphone. Govindaraj et al. [15] have designed a solution, called iSecureRing, which permits a jailbroken iPhone to be secure and ready for a digital forensic investigation by preserving timestamps in a secure location. Research conducted by Pieterse et al. [16] showcased the successful manipulation of timestamps stored in SQLite database on Android smartphones. The research also proposed the Authenticity Framework for Android Timestamps, which provides methods to identify manipulation timestamps. These solutions are, however, either platform-specific, require additional software to be installed on a smartphone prior to an investigation or only focus on a specific subset of smartphone data such as timestamps.

This paper attempts to establish an evaluation framework that assists with the identification of manipulated smartphone data on both Android and iOS platforms. To construct such a framework, it is necessary to determine what manipulative changes can be applied to smartphone data. This is possible by conducting exploratory experiments involving the manipulation of data on a Samsung Galaxy S5 Mini (Android version 6.0.1) and iPhone 7 (iOS version 10.0.1) smartphones. The steps followed to perform the manipulation form a generic process to generalise the manipulation techniques. Such manipulation of smartphone data is essentially an attack on the data's integrity and is best described using an attack tree. Using the attack tree, key traces left behind due to the manipulation of smartphone data leads to the formulation of the evaluation framework, which provides key indicators for digital forensics professionals to identify and pinpoint manipulated smartphone data. Weights assigned to the indicators allow for the detection of manipulated smartphone data with a certain probability. The immediate challenges to address in this paper are thus the following: (a) development of an effective and generic process to manipulate smartphone data on both Android and iOS platforms and (b) construct an evaluation framework capable of detecting manipulated smartphone data.

The remainder of this paper is structured as follows. Section II presents an overview of the Android and iOS platforms and discusses the structure of SQLite databases. The generic process for smartphone data manipulation, constructed using the results of the exploratory experiments, is discussed in Section III. Section IV presents an attack tree for smartphone data manipulation that encapsulates the available manipulation techniques. In Section V the evaluation framework for detecting manipulated smartphone data is introduced and three distinct scenarios are evaluated using the framework. Finally, Section VI discusses the findings while Section VII concludes the paper.

II. Background

With the continuous growth in functions and capabilities of smartphones supporting the Android and iOS platforms, valuable sources of smartphone data are collected on these devices. This section reviews the architecture and file system structure of the Android and iOS platforms. Attention is given to the storage location of the smartphone data on these platforms, as well as the accessibility of the data. Following the review of smartphone platforms is an overview of SQLite databases, which are a popular choice for persistent storage on smartphones.

A. Smartphone Platforms

Operating systems form the foundation of advanced capabilities and improved functionality showcased by smartphones today. They operate seamlessly and act as the intermediary layer between the user and the underlying hardware resources. High performance smartphone operating systems, which include Google Android and Apple iOS, are the current pace setters, as reflected by their combined market share of 98.59% in the 4th quarter of 2017.

The Google Android platform is an open source operating system provided by the Open Handset Alliance [17] and was officially announced in November 2007. The architecture of the platform is divided into six layers: system applications, Java API framework, native C/C++ libraries, Android runtime, hardware abstraction layer and Linux kernel [18]. This architecture ensures the effective operation of applications by allowing fluent communication between these applications and the lower layers. Until Android version 2.2 (Froyo), Android smartphones primarily used Yet Another Flash File System version 2 (YAFFS2) [19]. Android switched from YAFFS2 to Fourth Extended (EXT4) file system with the release of version 2.3 (Gingerbread) to more efficiently support multi-core chip sets [19]. The EXT4 file system also divides the disk space into logical storage units, which supports reduced management overhead and improves throughput [20]. With regards to digital forensic investigations, the logical storage units containing valuable smartphone data are the `/data` and `/system` partitions [21]. Access to these partitions is not permitted by default and is only accessible by rooting the Android smartphone. Rooting gives the user access to the root directory (/) and permits the execution of superuser privileges [17].

The Apple iOS platform is a proprietary and slimmed down version of the macOS [22] for Apple's mobile devices. The architecture of the iOS platform consists of five layers: applications, Cocoa touch, media, core services and core OS/kernel [23]. The iOS platform acts as an intermediary layer between the underlying hardware components and installed applications, causing the applications to interact with the hardware through a set of well-defined system interfaces [24]. A variation of the Hierarchical File System Plus (HFS+), called HFSX, was selected as the primary file system for iOS [25]. The single-threaded design and rigid data structures of HFSX struggled to keep pace with ever-improving technology. In 2016 Apple announced a new file system, called the Apple

File System (APFS), for all Apple's mobile operating systems, including iOS [25]. Similar to the Android platform, iOS also divides the logical storage space into partitions. Traditionally, iOS smartphones are configured with two partitions: system and data [26]. Access to smartphone data stored on these partitions is not allowed by default and users must jailbreak the iOS smartphone. The term jailbreak originates from a Unix practice of placing services in a restricted set of directories called a "jail" and breaking free from these restrictions [27]. By jailbreaking an iOS smartphone removes restrictions put in place by Apple and elevates the privileges to root access [28].

B. *SQLite Databases*

SQLite is best described as an efficient software library that implements a lightweight Structured Query Language (SQL) database engine [29]. The main database file (`.db`, `.db3` or `.sqlitedb`) contains a complete SQL structure that includes tables, indices, triggers and views [30]. The first page of the main database file is a 100-byte database header page. The remaining pages following the header page are structured as B-trees, where each page contains a B-tree index and B-tree table that holds the actual data [31].

During transactions, SQLite stores additional information in a secondary file called either a rollback journal or write-ahead log (WAL) file [32]. The purpose of this secondary file is to ensure the integrity of the data in the event of transaction failure. The WAL approach, which was introduced with version 3.7.0, preserves the original data in the main database file and appends changes to a separate WAL (.db-wal) file. The WAL file also contains a 32-byte file header and zero or more WAL frames. When a checkpoint occurs the updated or new records in the WAL file are written to the main database file. Once completed, the WAL file remains untouched and can be reused rather than deleted. Traditionally, SQLite performs an automatic checkpoint when the WAL file reaches a size of 1000 frames (approximately 4MB in file size) [33]. The number of WAL frames are calculated using the WAL file size (minus the WAL header size) divided by the combined size of the header and frame.

III. Generic Process for Smartphone Data Manipulation

The manipulation of smartphone data occurs for different reasons by applying various techniques. The available techniques to manipulate smartphone data are modification, fabrication or deletion. Modification of the smartphone data refers to tampering or altering of existing smartphone data. With modification, the existing data is updated to reflect changed data. Fabrication describes the creation of new but false smartphone data. The fabricated or counterfeited data is inserted to represent actual data. Finally, deletion of smartphone data removes the data.

To establish a generic process for smartphone data manipulation, exploratory experiments involving both the Android and iOS default messaging applications are performed. The purpose of these experiments is to obtain access to the persistent data of the applications and attempt the manipulation, which is either the modification, fabrication or deletion, of the data. While performing the exploratory experiments, the steps followed to manipulate the smartphone data are carefully documented. From the observations, similarities are identified and collected into a generic process.

A. *Manipulation of Android Smartphone Data*

The first exploratory experiment focuses on the Android platform and uses a Samsung Galaxy S5 Mini, running Android version 6.0.1 (Marshmallow), as the test smartphone. To manipulate the smartphone data of Android's default messaging application, access to the file(s) responsible for storing the data is required. The Android platform stores all application-related smartphone data in the `/data` folder and access is only possible on a rooted smartphone, as mentioned in Section 2.1. Root access on the Samsung Galaxy S5 Mini is obtained using the CF Auto Root and Odin tools.

Android's default messaging application uses an SQLite database for data storage and is located in the `/data/data/com.android.providers.telephony/database/` folder on the Android smartphone. At this point, manipulation in the form of deletion is possible by simply removing the SQLite database files (`.db` and `.db-wal`) and rebooting the smartphone. This deletes all of the smartphone data related to the application. Modification of existing data or adding newly fabricated data requires direct access to the SQLite database records. Applying changes directly to the data in the SQLite database files is not feasible due to the complex structure of the files and the possibility of applied changes being overwritten. It is, therefore, necessary to open and access the data in these file(s). Two approaches exist to access the SQLite database files: direct or off-device.

The direct approach involves the manipulation of the smartphone data by opening the SQLite database on the Android smartphone. This requires the use of an appropriate tool, such as the `sqlite3` command-line program, to manually enter and execute SQL statements [34]. This program provides access to the SQLite database records (using the `.open` command) and allows for the manipulation of the smartphone data using the appropriate SQL statements (INSERT, UPDATE or DELETE). Android smartphones do not ship with a pre-installed `sqlite3` command-line program. Absence of or failure to utilise the `sqlite3` command-line program necessitates the use of the off-device approach.

The off-device approach requires an established communication channel between the smartphone and a connected computer. Establishing such a channel relies on the USB debugging functionality, which is not visible by default. Although not visible, going to Settings, About phone and tapping multiple times on the build number will enable Developer mode. Selecting Developer options and touching the check box next to "USB debugging" will enable this feature. Following the enabling of the "USB debugging" feature, it is possible to create a communication channel using the Android Debug Bridge (ADB). ADB is a versatile command-line utility that communicates with a connected Android smartphone [35]. The communication channel is established using `adb shell`, followed immediately by the `su` command. Using the established communication channel, the SQLite database files are first transferred to the `/sdcard` folder before downloading the files to the connected computer. The `/sdcard` folder is found across all Android smartphones, regardless of make or model, and allows end-users to store additional files and data. Using an SQLite editor to open the `.db` file causes an automatic checkpoint to occur, ensuring all the records in the `.db-wal` file are transferred and visible in the editor. It is now possible to manipulate the smartphone data using the available SQL statements (INSERT, UPDATE or DELETE). After completing the manipulation of the smartphone data, the SQLite database is closed to ensure the changes are captured correctly.

To complete the manipulation of the smartphone data, the remaining `.db` file must be returned to the Android smartphone. Before this file can be returned to the `/data/data/com.android.providers.telephony` texttt/databases/ folder, the original SQLite database (`.db` and `.db-wal` files) must be removed using the `rm` command. The removal of the original SQLite database files prevents the manipulated data from being overwritten. Thereafter, the `.db` file can be returned to the `/data/data/com.android.providers.telephony` texttt/databases/ folder using the `mv` command. The only required file is the `.db-wal` file, which is generated following a smartphone reboot. The permission of the `.db` file must be changed using the `chmod a=rw` or `chmod 666` command to create a new `.db-wal` file. The reboot also ensures the manipulated data is visible on the Android smartphone.

This concludes the exploratory experiment of manipulating Android smartphone data. The following section attempts the manipulation of smartphone data residing on an iPhone 7.

B. *Manipulation of iOS Smartphone Data*

The second exploratory experiment focuses on the iOS platform and uses an Apple iPhone 7, running iOS version 10.0.1, to perform the experiments. To manipulate the smartphone data that forms part of the default messaging application on the iPhone 7, access to the file(s) storing the data is required. The iOS platform stores application smartphone data in the `/private/var/mobile/Library` folder. Access to this folder is only permitted on a jailbroken smartphone, which is achieved by using the `extra_recipe + yaluX` jailbreak application and Impactor to transfer the application to the iPhone 7. Upon installing the application, the jailbreak executes and immediately reboots. The jailbreak status is confirmed by verifying the automatic installation of the Cydia application, a package manager for jailbroken iPhones.

iPhone's default messaging application uses a SQLite database for storing data. The SQLite database is found in the `/private/var/mobile/Library/SMS/` folder on the iPhone 7. At this point, manipulation in the form of deletion is possible by removing the SQLite database files (`.db` and `.db-wal`) and performing a smartphone reboot. Again, this removes all of the smartphone data related to the application. Modification of existing or the creation of counterfeited data necessitates access to the SQLite database records. Access to the records is possible via one of the following two approaches: direct or off-device.

The direct approach involves the manipulation of the smartphone data by opening the SQLite database on the iPhone 7. This approach relies on the presence and availability of the `sqlite3` program on the iPhone 7. In contrast to Android, iOS comes pre-installed with the `sqlite3` program. The program provides direct access to the SQLite database and permits the modification of existing data, fabrication of new data, as well as the removal of all or specific data. Should the `sqlite3` program fail to effectively apply the changes to the smartphone data, it will be necessary to follow the off-device approach.

The off-device approach requires the transferral of the SQLite database (both the `.db` and `.db-wal` files) to a connected computer. A communication channel is established using the iFunbox and puTTy applications. Obtaining access to the iPhone 7 file system is possible using the standard iOS credentials, which is root (username) and alpine (password). Thereafter, the SQLite database files is first transferred to the `/var/mobile/Media` folder before downloading the files unto the connected computer. The `/private/var/mobile/Media` folder is similar to Android's `/sdcard` folder and allows users to store additional media and downloaded files. Using a SQLite editor, the `.db` file is opened and immediately causes an automatic checkpoint (see Section 2.2). It is now possible to manipulate the smartphone data using the available SQL statements (INSERT, UPDATE or DELETE). After completing the manipulation, the SQLite database is closed to ensure the changes are correctly captured.

For the manipulated data to reflect on the iPhone 7, it is necessary to return the `.db` file. Before returning the file to the `/private/var/mobile/Library/ SMS` folder, the existing SQLite database (`.db` and `.db-wal` files) must be removed using the `rm` command. These files, especially the `.db-wal` file, are removed to ensure the manipulated data is not overwritten. Thereafter, the `.db` file can be transferred to the `/private/var/mobile/Library/SMS` folder using the `mv` command. The only required file is the `.db-wal` file, which is created following a smartphone reboot. To generate the new and empty `.db-wal` file, the current permissions of the `.db` file must be changed using the `chmod a=rw` or `chmod 666` command. This ensures the `.db-wal` file is created and the manipulated data is visible on the iPhone 7.

The successful manipulation of iOS smartphone data concludes the exploratory experiment. The following section consolidates the findings found across both exploratory experiments and formulates a generic process for smartphone data manipulation.

C. *Generic Process*

The exploratory experiments performed in the previous sections confirm that it is indeed possible to manipulate smartphone data on both the Android and iOS platforms. Although the focus was on the manipulation of text messages of the default messaging applications, the same steps can be followed to manipulate any other smartphone data stored within SQLite structures. From these experiments, it is now possible to pinpoint various similarities among the steps followed to manipulate smartphone data. Using the collected similarities, a generic process is formulated that generalises the manipulation of smartphone data. The generic process consists of four distinct stages. Each individual stage describes the progression of the generic process to manipulate the smartphone data along with the requirements that must be met to successfully complete each stage, as well as the actual manipulation.

- **Phase 1**: ensures the selected smartphone is accessible by confirming the smartphone is either rooted (Android) or jailbroken (iOS).
- **Phase 2**: requires the selection of the application and identifying the location of the file(s), such as a SQLite database, storing the smartphone data. The data of the selected smartphone application must reside on the smartphone.
- **Phase 3**: identify the most appropriate approach to access the smartphone data: Direct or Off-device.
- **Phase 3.1**: the direct approach performs the manipulation of the smartphone data directly on the smartphone and relies on the presence of a program or utility to access the file(s).
- **Phase 3.2**: the off-device approach requires the transferral of the file(s) to the connected computer. Using the most appropriate program or utility, the contents of the file(s) is accessed and manipulated accordingly. Once completed, the file(s) is closed and returned to the smartphone to overwrite previous smartphone data. The returned file(s) is also assigned the necessary read/write permissions to ensure the smartphone application can interact with the manipulated smartphone data.
- **Phase 4**: requires a manual reboot of the smartphone.

This proposed generic process for smartphone data manipulation captures the steps to follow to modify, fabricate or delete smartphone data. The following section further investigates the manipulation of smartphone data by introducing an attack tree that encapsulates the various manipulation scenarios.

IV. Attack Tree for Smartphone Data Manipulation

The established generic process for smartphone data manipulation provides the steps to affect changes to data. Such changes are essentially an attack on the integrity, availability and authenticity of smartphone data and is best described using an attack tree. An attack tree provides a formal and methodical way to describe various attacks against a system [36]. The attacks are represented using a conceptual tree structure with the main goal of these attacks listed as the root node. The nodes following the root describes the different avenues of achieving the goal, constructed using OR (choice between alternative steps) and AND (represents different steps to achieve the same goal) nodes.

A. Construction of the Attack Tree

The goal of this attack tree is the "manipulation of smartphone data" and is denoted by G. The intermediate goals are: deletion (I_1), modification (I_2) or fabrication (I_3). Following the intermediate goals are the sub-goals that describes the required steps to accomplish each intermediate goal and ultimately complete the set goal. There are two options for deletion: removal of the files holding the data which deletes all of the data (S_1) or removing specific data such as individual records (S_2). Removal of the file(s) requires physical access to the smartphone(S_5) by either rooting (Android) or jailbreaking (iOS) the smartphone. Once access is acquired, it is necessary to locate and remove the file(s) (S_6). This is followed by a reboot of the smartphone (S_7) and ensures the deletion of all the smartphone data related to the smartphone application. The removal of individual records also requires physical access to the smartphone and locating the file(s) holding the data. Since this attack focuses on the manipulation of specific data, it is necessary to access and open the file(s) containing the data (S_8). Options to open the file(s) are either directly on the smartphone (S_9) or off-device on a connected computer (S_{10}). To open and view the data requires the use of an appropriate utility or program to access the data (S_{11}). Should such utility or program be unavailable or the approach not be feasible, access to the file(s) holding the data must occur off-device on a computer connected to the smartphone. Off-device manipulation requires the transferral of the file(s) (S_{12}), which relies on an established connection between the smartphone and the connected computer (S_{14}). After performing the manipulating, the file(s) are returned to the smartphone via the established connection (S_{13}). This is again followed by a smartphone reboot (S_7) to ensure the removed data reflects on the smartphone.

The remaining manipulation techniques, modification (I_2) and fabrication (I_3), follows similar attack paths. To either change existing data (S_3) or insert fabricated data (S_4), it is necessary to open and access the data in the file(s). Therefore, these manipulation techniques follows a path identical to the removal of individual records. According to the descriptions above, the attack tree is constructed and presented in Fig. 1. This attack tree forms the basis for deriving attack scenarios to manipulate smartphone data.

B. Derived Attack Scenarios

The presented attack tree provides four distinct techniques to manipulate smartphone data (deletion of all data, deletion of specific data, modification of data or fabricating data). The focus of this section is on three theoretical attack scenarios that involve the deletion and modification of existing data, as well as a failed attempt to insert fabricated data.

The first attack scenario illustrates the deletion of specific data stored by a smartphone application. For this attack scenario, the iOS operating system is identified as the target platform and involves a previously jailbroken iPhone 7. The availability of the `sqlite3` command-line program permits manipulation of the smartphone data directly on the smartphone by following the direct approach (see Section III-C).

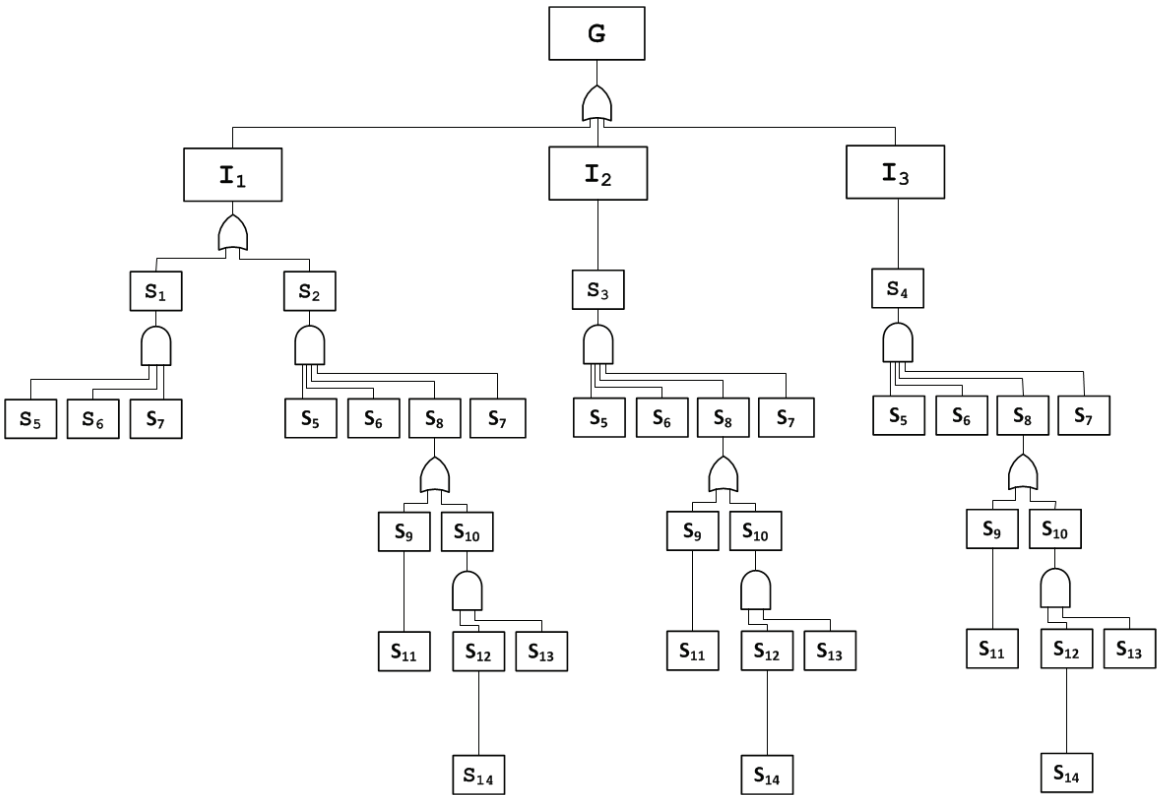


Fig. 1. Attack tree that illustrates the steps for smartphone data manipulation

Using the provided attack tree, the described attack scenario is denoted as follows: $I_1, S_2, S_5, S_6, S_8, S_9, S_{11}, S_7$.

The second attack scenario demonstrates the modification of existing data present on an Android smartphone. Access to the data necessitates rooting the Android smartphone. Due to the unavailability of the `sqlite3` command-line program on the Android platform, the modification of the data must occur using the off-device approach (see Section III-C). Again using the provided attack tree, the described attack scenario is denoted as follows: $I_2, S_3, S_5, S_6, S_8, S_{10}, S_{12}, S_{14}, S_{13}, S_7$.

Th final attack scenario attempts to insert newly fabricated data. For this attack scenario, the Android operating system is again selected as the target platform and rooted. Following the off-device approach, the SQLite database of the application selected to hold the fabricated smartphone data is transferred to the connected computer. However, failure to retrieve the SQLite database from the Android smartphone ultimately terminates the attack scenario.

The presented attack scenarios will have inherent side-effects that leaves various traces on the smartphones. Traces specific to each sub-goal are listed in Table I. Collectively, the traces provides evidence that can assist with the identification of manipulated smartphone data. The following section further explores these traces by extracting key indicators and using the indicators to construct an evaluation framework for smartphone data.

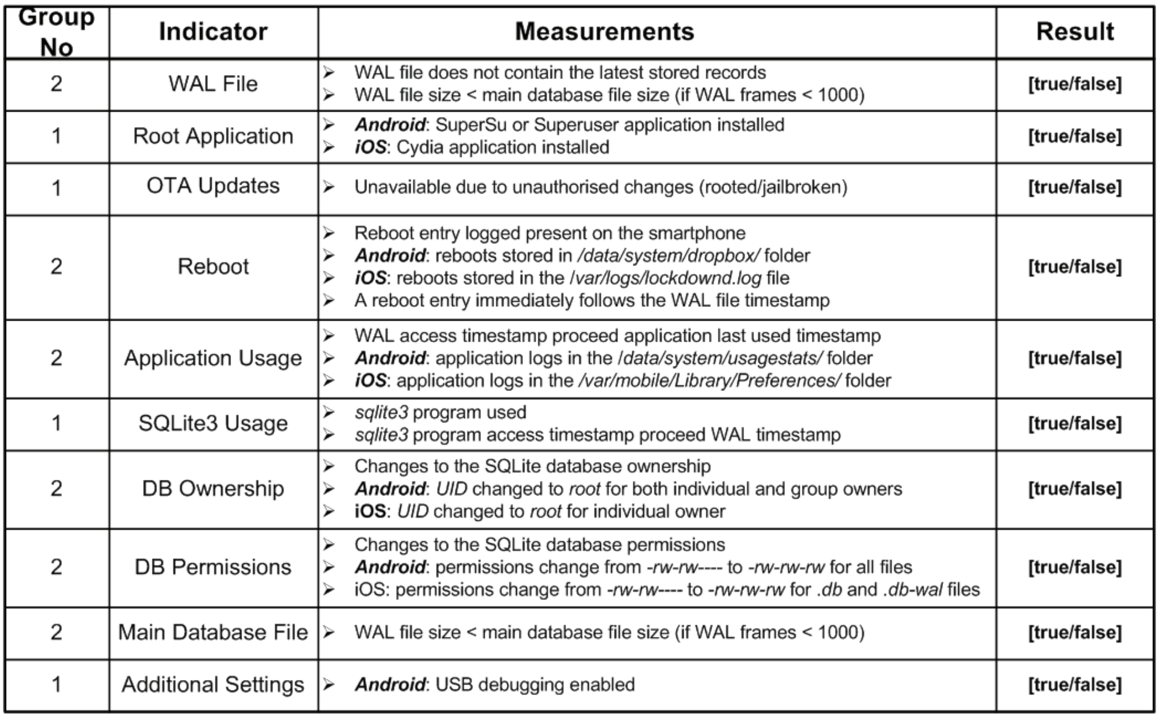
V. Evaluation Framework

The collection of traces deduced from the various manipulation techniques encapsulated in the attack tree equips digital forensic professionals with the necessary information to evaluate smartphone data. There is, however, no structure or order to these traces, which can impact the effective use of the traces to detect manipulated smartphone data. To assist digital forensic professionals, key indicators are extracted from these traces and captured in an evaluation framework. Fig. 2 presents the evaluation framework for detecting manipulated smartphone data.

From the collected traces 10 distinct indicators are identified, which are listed in the above evaluation framework. Each indicator is a possible side-effect that occurs due to the manipulation of the smartphone data. Certain indicators, such as the root status and OTA updates, are not a direct indication of the intentional manipulation of smartphone data. However, the manipulation necessitates the need for rooting/jailbreaking the smartphone, which also impacts the availability of OTA updates. Therefore, a larger collection of present indicators is

TABLE I
Traces created due to the manipulation of smartphone data

Sub-Goal	Trace Created
S_1, S_2, S_3, S_4	The presence of a new and clean WAL file
S_5	Automatic installation of a root application
S_5	Unavailability of over-the-air (OTA) updates
S_7	Creation of a new entry in the reboot log
S_8	Discrepancy between WAL and application timestamp
S_9, S_{11}	Use of the `sqlite3` command-line program
S_{10}, S_{12}	Change in ownership of the `.db` file
S_{10}, S_{12}	Change in permissions for the `.db` file
S_{10}, S_{13}	The `.db` file size larger than `.db-wal` file
S_{14}	Enabled settings (USB debugging)



Group No	Indicator	Measurements	Result
2	WAL File	➢ WAL file does not contain the latest stored records ➢ WAL file size < main database file size (if WAL frames < 1000)	**[true/false]**
1	Root Application	➢ ***Android***: SuperSu or Superuser application installed ➢ ***iOS***: Cydia application installed	**[true/false]**
1	OTA Updates	➢ Unavailable due to unauthorised changes (rooted/jailbroken)	**[true/false]**
2	Reboot	➢ Reboot entry logged present on the smartphone ➢ ***Android***: reboots stored in */data/system/dropbox/* folder ➢ ***iOS***: reboots stored in the */var/logs/lockdownd.log* file ➢ A reboot entry immediately follows the WAL file timestamp	**[true/false]**
2	Application Usage	➢ WAL access timestamp proceed application last used timestamp ➢ ***Android***: application logs in the */data/system/usagestats/* folder ➢ ***iOS***: application logs in the */var/mobile/Library/Preferences/* folder	**[true/false]**
1	SQLite3 Usage	➢ *sqlite3* program used ➢ *sqlite3* program access timestamp proceed WAL timestamp	**[true/false]**
2	DB Ownership	➢ Changes to the SQLite database ownership ➢ ***Android***: *UID* changed to *root* for both individual and group owners ➢ **iOS**: *UID* changed to *root* for individual owner	**[true/false]**
2	DB Permissions	➢ Changes to the SQLite database permissions ➢ ***Android***: permissions change from *-rw-rw----* to *-rw-rw-rw* for all files ➢ iOS: permissions change from *-rw-rw----* to *-rw-rw-rw* for *.db* and *.db-wal* files	**[true/false]**
2	Main Database File	➢ WAL file size < main database file size (if WAL frames < 1000)	**[true/false]**
1	Additional Settings	➢ ***Android***: USB debugging enabled	**[true/false]**

Fig. 2. Evaluation framework to identify manipulated smartphone data

a better reflection of the manipulation of smartphone data.

To pinpoint these indicators, specific measurements are presented to assist digital forensic professional. Where necessary, explicit measures are specified for the different smartphone platforms. Each evaluated indicator produces a binary result that reflects either a positive [**true**] or negative [**false**] result. A positive result indicates the evaluated measurement(s) are met while a negative result contradicts the indicator.

The presented indicators only stipulates how to evaluate smartphone data without providing an outcome regarding the potential manipulation of the data. The impact of each indicator on the evaluation of the smartphone data is, however, not equal since each indicator evaluates different aspects regarding the manipulation of smartphone data. Stemming from the provided descriptions are two distinct groups of indicators. The first group (1) contains indicators that merely confirms an opportunity existed for the smartphone data to be manipulated. Belonging to group 1 are the following indicators: Root Application, OTA Updates, SQLite3 Usage and Additional Settings. The second group (2) collects indicators that specifically focus on affirming changes to the smartphone and the application responsible for creating the smartphone data. The remaining indicators, namely WAL File, Reboot, Application Usage, DB Ownership, DB Permissions and Main Database File, pertain to group 2. The categorisation of the indicators according to these groups allows for a weighted calculation of a manipulation score.

Following a weighted approach to calculate a manipulation score (M_s) allows for better communication of the potential manipulation of the smartphone data. The weight assigned to each group reflects the impact the evaluated indicator will have on the final score. Two distinct groups for the available indicators have been identified. Since group 1 contains 40% of the available indicators, a weight of 0.4 is assigned to the group. The weight assigned to group 2 is the remainder, which is 0.6. The manipulation score (M_s) is then calculated using equation (1) by accumulating all of the positive results per group (pos_g) and then dividing by the number of indicators available per group (n_g). The result is then weighed using the assigned weight as specified above.

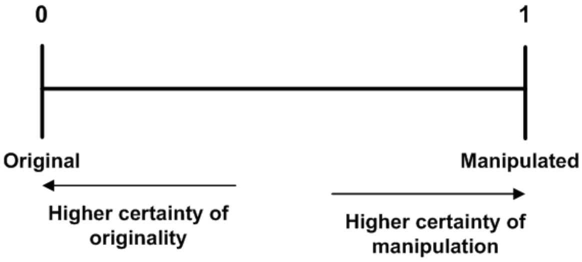
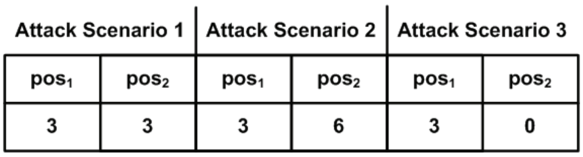

0
1
Original
Manipulated
Higher certainty of originality
Higher certainty of manipulation

Fig. 3. Probability scale to measure manipulation of smartphone data

$$M_s = \sum_{g=1}^{2} (w_g)(\frac{pos_g}{n_g}) \tag{1}$$

Using the probability scale shown in Fig. 3, the calculated manipulation score can be plotted to reflect the potential manipulation of the evaluated smartphone data. Confirming the effectiveness of the evaluation framework to identify manipulated smartphone data, necessitates the assessment of the derived attack scenarios introduced in Section IV-B using the framework.



Attack Scenario 1		Attack Scenario 2		Attack Scenario 3	
pos_1	pos_2	pos_1	pos_2	pos_1	pos_2
3	3	3	6	3	0

Fig. 4. Collection of positive results per group for each attack scenario

A. *Evaluation of Attack Scenarios*

The manipulation of smartphone data impacts and negatively influences the conclusion drawn from the analysed smartphone data. The constructed evaluation framework provides a suitable methodology to assess smartphone data and identify the manipulation of such data. Applying the attack scenarios (see Section IV-B) at a theoretical level and assessing the outcomes using the evaluation framework, which permits digital forensic professionals to pinpoint manipulated data.

The first attack scenario illustrated the deletion of specific data stored on an iPhone 7. Completion of the attack scenario caused the following indicators to present on the iPhone 7: WAL File, Root Application, OTA Updates, Application Usage, SQLite program and Reboot. Since the manipulation of the smartphone data occurred on the iPhone 7, fewer indicators are available and, therefore, the manipulation score is expected to be approximately 0.5. All of the indicators meet the provided measurements and using equation 1, the calculated manipulation score is 0.6.

The second attack scenario demonstrated the modification of existing data available on an Android smartphone. Completion of the attack scenario caused the following indicators to present on the Android smartphone: WAL File, Root Application, OTA Updates, Reboot, Application Usage, DB Ownership, DB Permissions, Main Database File and Additional Settings. Since the manipulation of the smartphone data followed the off-device approach (see Section III-C), more indicators are present on the Android smartphone. The manipulation score is, therefore, expected to be above 0.5. All of the indicators meet the provided measurements and using equation 1, the calculated manipulation score is 0.9.

The final attack scenario attempted the insertion of newly fabricated smartphone data on an Android smartphone. The attack scenario was, however, unsuccessful but still caused the following indicators to present: Root Application, OTA Updates and Additional Settings. The present indicators confirm that an opportunity existed for the smartphone data to be manipulated but since no manipulation occurred, the manipulation score is expected to be below 0.5. All of the indicators meet the provided measurements and using equation 1, the calculated manipulation score is 0.3.

Fig. 4 summarises the collection of positive results per group for each attack scenario. The results show that each attack scenario had equal opportunity to manipulate the smartphone data but only attack scenarios 1 and 2 successfully manipulated the data. Fig. 5 maps the final calculated manipulation scores on the probability scale. The results confirm that the evaluation framework along with the calculated manipulation score can assist digital forensic professionals with the identification of manipulated data.

VI. Discussion and Future Work

The purpose of the evaluation framework is to assist digital forensic professionals with the assessment of smartphone data and allow for the identification of possibly manipulated data. The evaluation framework consist of ten indicators and associated measurements to assess smartphone data. These indicators are divided into two distinct groups to better reflect both the opportunity to manipulate the smartphone data, as well as the actual manipulation of the data. Depending on the data collected from the smartphone, digital forensic professionals can either evaluate all or a subset of the indicators. The adjustable structure of the framework easily supports different collections of smartphone data. The easy to follow structure provides a step-by-step guide for evaluating the smartphone data, thus quickening the assessment of smartphone data while saving digital forensic professionals valuable time during examinations. The evaluation framework is simplistic yet comprehensive, providing digital forensic professionals with an easy to understand methodological approach to evaluate smartphone data.

The result(s) produced by the evaluation framework allows digital forensic professionals to make informed decisions regarding the inclusion or exclusion of smartphone data. Evaluation of the attack scenarios in Section V-A showed the evaluation framework can identify manipulated smartphone data. Furthermore, the grouping and subsequent weighing of the indicators describe the certainty of the manipulation of the smartphone data using the probability scale. Using the produced result(s), digital forensic professionals can eliminate unreliable smartphone data from being submitted as potential digital evidence and only use reliable data to formulate and draw accurate conclusions.

Future work can build on this research by expanding the existing evaluation framework and establishing other approaches to identify manipulated smartphone data. Firstly, the existing collection of indicators can be extended by reviewing other smartphone platforms and identifying additional indicators that can pinpoint the manipulation of smartphone data. Secondly, the current focus of the evaluation framework is only on determining the manipulation of smartphone data. Adapting the

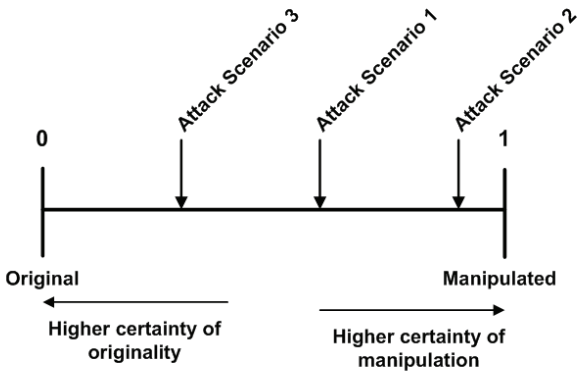


Fig. 5. Manipulation scores mapped on the probability scale

framework to also evaluate the accuracy and authenticity of the smartphone data provides a more comprehensive assessment of the data. Lastly, the provided evaluation framework must be continuously reviewed to ensure framework aligns with the technological improvements of smartphone platforms.

VII. Conclusion

Smartphone data found on both Android and iOS devices can form an important component of digital forensic investigations. Available smartphone data provides a well-defined snapshot of user events. To protect their privacy or hide incriminating events, users can deploy anti-forensics to manipulate smartphone data. To assist digital forensic professionals with the detection of such manipulated smartphone data, this paper introduces an evaluation framework for detecting manipulated smartphone data. The framework uses key traces left behind as a result of the manipulation of smartphone data to construct techniques to detect the changed data. The challenges addressed in this paper were to show (a) that smartphone data can be manipulated and (b) construct an evaluation framework to detect such manipulated data. Challenge (a) was addressed by formulating the generic process to manipulate smartphone data on both the Android and iOS platforms. Challenge (b) was concluded by introducing the evaluation framework for smartphone data and confirming the framework can assist with the identification of manipulated data. The current paper provides preliminary evidence that the suggested evaluation framework shows potential and future work will focus on expanding this research.

References

[1] (2018, Dec.) Operating system market share. netmarketshare. [Online]. Available: https://netmarketshare.com/operating-system-market-share.aspx

[2] H. Pieterse, M. Olivier, and R. van Heerden, "Evaluating the authenticity of smartphone evidence," in *Advances in Digital Forensics XIII*, vol. 511, 2017, pp. 41–51.

[3] R. Ayers, S. Brothers, and W. Jansen, "Guidelines on mobile device forensics (draft)," NIST, Tech. Rep. Special Publication 800-101, 2013.

[4] P. Albano, A. Castiglione, G. Cattaneo, G. D. Maio, and A. D. Santis, "On the construction of a false alibi on the android OS," in *Third International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, 2011, pp. 685–690.

[5] H. Pieterse, M. Olivier, and R. van Heerden, "Smartphones as distributed witnesses for digital forensics," in *Advances in Digital Forensics X*, vol. 433, 2014, pp. 237–251.

[6] M. Kala and R. Thilagaraj, "A framework for digital forensics in i-devices: Jailed and jail broken devices," *Journal of Advances in Library and Information Science*, vol. 22, no. 2, pp. 82–93, 2013.

[7] M. Tsavli, P. Efraimidis, and V. Katos, "Reengineering the user: privacy concerns about personal data on smartphones," *Information & Computer Security*, vol. 23, no. 4, pp. 394–405, 2015.

[8] R. Harris, "Arriving at an anti-forensics consensus: Examining how to define and control the anti-forensics problem," *Digital Investigation*, vol. 3, pp. 44–49, 2006.

[9] P. Albano, A. Castiglione, G. Cattaneo, and A. D. Santis, "A novel anti-forensics technique for the android OS," in *International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)*, 2011, pp. 380–385.

[10] S. Azedegan, W. Yu, H. Liu, M. Sistani, and S. Acharya, "Novel anti-forensics approaches for smart phones," in *45th Hawaii International Conference on System Sciences (HICSS)*, 2012, pp. 5424–5431.

[11] C. D'Orazio, A. Ariffin, and K. Choo, "iOS anti-forensics: How can we securely conceal, delete and insert data?" in *47th Hawaii International Conference o System Sciences (HICSS)*, 2014, pp. 4838–4847.

[12] K. Karlsson and W. Glisson, "Android anti-forensics: Modifying cyanogenmod," in *7th Hawaii International Conference of System Sciences (HICSS)*, 2014, pp. 4828–4837.

[13] J. Zheng, Y. Tan, X. Zhang, C. Liang, C. Zhang, and J. Zheng, "An anti-forensics method against memory acquiring for Android devices," in *International Conference on Computational Science and Engineering (CSE) and Embedded and Ubiquitous Computing (EUC)*, 2017, pp. 214–218.

[14] R. Verma, J. Govendaraj, and G. Gupta, "Preserving dates and timestamps for incident handling in android smartphones," in *Advances in Digital Forensics X*, vol. 433, 2014, pp. 209–225.

[15] J. Zheng, Y. Tan, X. Zhang, C. Liang, C. Zhang, and J. Zheng, "iSecureRing: Forensic ready secure iOS apps for jailbroken iPhones," in *35th IEEE Symposium on Security and Privacy*, 2014.

[16] H. Pieterse, M. Olivier, and R. van Heerden, "Playing hide-and-seek: Detecting the manipulation of android timestamps," in *Information Security for South Africa*, 2014, pp. 1–8.

[17] J. Lessard and G. Kessler, "Android forensics: Simplifying cell phone examinations," *Small Scale Digital Device Forensics Journal*, vol. 4, no. 1, pp. 1–12, 2010.

[18] (2017, Oct.) Platform architecture. android. [Online]. Available: http://developer.android.com/guide/platform/

[19] C. Zimmermann, M. Spreitzenbarth, S. Schmitt, and F. Freiling, "Forensic analysis of YAFFS2," in *Sicherheit*, 2012, pp. 59–69.

[20] H.-J. Kim and J.-S. Kim, "Tuning the EXT4 filesystem performance for android-based smartphones," in *Frontiers in Computer Education*, 2013, pp. 745–752.

[21] R. Tamma and D. Tindall, *Learning Android Forensics*. Birmingham, UK and Mumbai, India: Packt Publishing Ltd, 2015.

[22] K. Tracy, "Mobile application development experiences on Apple's iOS and Android OS," *IEEE Potentials*, vol. 31, no. 4, pp. 30–34, 2012.

[23] (2017, Oct.) iOS technology overview. apple. [Online]. Available: http://developer.apple.com/library/ content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview Introduction/Introduction.html

[24] M. Kanoi and Y. Jdiet, "Internal structure of iOS and building tools for iOS apps," *International Journal of Computer Science and Applications*, vol. 6, no. 2, pp. 220–225, 2013.

[25] E. Tamura and D. Giampaolo, "Introducing Apple file system," Apple, Inc., Tech. Rep., 2016.

[26] M. Epifani and P. Stirparo, *Learning iOS Forensics*. Birmingham, UK and Mumbai, India: Packt Publishing Ltd, 2016.

[27] J. Zdziarski, *iPhone forensics: Recovering evidence, personal data and corporate assets*. Sebastopol, California: O'Reilly Media, Inc., 2008.

[28] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, "PiOS: Detecting privacy leaks in iOS applications," in *NDSS*, 2011, pp. 177–183.

[29] M. Kanoi and Y. Jdiet, "A recovery method of deleted record for SQLite database," *Personal and Ubiquitous Computing*, vol. 16, no. 6, pp. 707–715, 2012.

[30] (2018, Apr.) About SQLite. SQLite. [Online]. Available: https://www.sqlite.org/about.html

[31] P. Patodi, "Database recovery mechanism for Android devices," Ph.D. dissertation, Indian Institute of Technology, Bombay, 2012.

[32] (2018, Apr.) Database file format. SQLite. [Online]. Available: https://www.sqlite.org/fileformat.html

[33] (2018, Apr.) Write-ahead logging. SQLite. [Online]. Available: https://www.sqlite.org/wal.html

[34] (2018, Apr.) Command line shell for SQLite. SQLite. [Online]. Available: https://www.sqlite.org/cli.html

[35] (2018, Jan.) Android debug bridge (adb). android studio. [Online]. Available: http://developer.android.com/studio/command-line/adb.html

[36] B. Scheier, "Attack trees," *Dr. Dobb's Journal*, vol. 24, no. 12, pp. 21–29, 1999.

Heloise Pieterse is a Senior Researcher in the Cyber Warfare research group at the Council for Scientific and Industrial Research, Pretoria, South Africa. Heloise holds a BSc, BSc (Honours) and MSc degrees in Computer Science and is currently pursuing a Ph.D. in Computer Science with the focus on the authenticity of smartphone data at the University of Pretoria, Pretoria, South Africa. Her research interests include digital forensics, especially smartphone forensics and mobile device security.

Martin Olivier is a professor at the Department of Computer Science in the School of Information Technology at the University of Pretoria. His current research interest is digital forensics. Previously he worked on privacy, database, application and system security. Prof Olivier started his career at the CSIR in Pretoria. In 1991 he joined the Department of Computer Science at the Rand Afrikaans University in Johannesburg as a lecturer. He has also lectured part-time at a number of tertiary institutions in South Africa. Prof Olivier holds a BSc degree in Mathematical Sciences, BSc (Honours), MSc and PhD degrees in Computer Science, a BA degree in Humanities, a BA (Honours) degree in Philosophy and an MPhil degree in Workplace Ethics. He is also a Certified Cyber Forensic Professional (CCFP).

Renier van Heerden is a senior researcher at Council for Scientific and Industrial Research (CSIR) in Pretoria, South Africa in the field of Cyber Security and currently Services Development and Incubation manager (acting) at the SANReN CA. The following domains are his interest: password security, network attack and network ontologies. Prior to joining the CSIR he worked as a software engineer in advanced optics applications for South African based Denel Optronics and as a Lecturer at the University of Pretoria. Renier obtained a degree in Electronic Engineering, a Masters in Computer Engineering at the University of Pretoria and PhD at Rhodes University.