

Evaluation and Identification of Authentic Smartphone Data

by

Heloise Pieterse

Submitted in fulfilment of the requirements for the degree
Philosophiae Doctor (Computer Science)
in the Faculty of Engineering, Built Environment and Information Technology
University of Pretoria, Pretoria

July 2019

Publication data:

Heloïse Pieterse. Evaluation and Identification of Authentic Smartphone Data. Doctoral thesis, University of Pretoria, Department of Computer Science, Pretoria, South Africa, July 2019.

Electronic, hyperlinked versions of this thesis are available online, as Adobe PDF files, at:

<http://repository.up.ac.za>

Evaluation and Identification of Authentic Smartphone Data

by

Heloïse Pieterse

E-mail: heloisep085@gmail.com

Abstract

Mobile technology continues to evolve in the 21st century, providing end-users with mobile devices that support improved capabilities and advance functionality. This ever-improving technology allows smartphone platforms, such as Google Android and Apple iOS, to become prominent and popular among end-users. The reliance on and ubiquitous use of smartphones render these devices rich sources of digital data. This data becomes increasingly important when smartphones form part of regulatory matters, security incidents, criminal or civil cases. Digital data is, however, susceptible to change and can be altered intentionally or accidentally by end-users or installed applications. It becomes, therefore, essential to evaluate the authenticity of data residing on smartphones before submitting the data as potential digital evidence.

This thesis focuses on digital data found on smartphones that have been created by smartphone applications and the techniques that can be used to evaluate and identify authentic data. Identification of authentic smartphone data necessitates a better understanding of the smartphone, the related smartphone applications and the environment in which the smartphone operates. Derived from the conducted research and gathered knowledge are the requirements for authentic smartphone data. These requirements are captured in the smartphone data evaluation model to assist digital forensic professionals with the assessment of smartphone data. The smartphone data evaluation model, however, only stipulates how to evaluate the smartphone data and not what the outcome of the evaluation is. Therefore, a classification model is constructed using the identified requirements and the smartphone data evaluation model. The classification model presents a formal classification of the evaluated smartphone data, which is an

ordered pair of values. The first value represents the grade of the authenticity of the data and the second value describes the completeness of the evaluation. Collectively, these models form the basis for the developed Smartphone Application Data Authenticity Classifier (SADAC) tool, a proof of concept digital forensic tool that assists with the evaluation and classification of smartphone data.

To conclude, the evaluation and classification models are assessed to determine the effectiveness and efficiency of the models to evaluate and identify authentic smartphone data. The assessment involved two attack scenarios to manipulate smartphone data and the subsequent evaluation of the effects of these attack scenarios using the SADAC tool. The results produced by evaluating the smartphone data associated with each attack scenario confirmed the classification of the authenticity of smartphone data is feasible. Digital forensic professionals can use the provided models and developed SADAC tool to evaluate and identify authentic smartphone data.

The outcome of this thesis provides a scientific and strategic approach for evaluating and identifying authentic smartphone data, offering needed assistance to digital forensic professionals. This research also adds to the field of digital forensics by providing insights into smartphone forensics, architectural components of smartphone applications and the nature of authentic smartphone data.

Keywords: Digital forensics, Mobile forensics, Smartphones, Smartphone applications, Smartphone data, Authenticity, Evidence, Reference architecture, Android, iOS.

Supervisor : Prof. M. S. Olivier

Co-Supervisor : Dr. R. P. van Heerden

Department : Department of Computer Science

Degree : Philosophiae Doctor

To My Mother

For continuously being a guiding light in my life.

“The grey rain curtain of this world rolls back, and all turns to silver glass...and then you see it. White shores. And beyond...a far green country under a swift sunrise.”

Gandalf the White - The Lord of the Rings: The Return of the King (2003)

Acknowledgements

My sincere gratitude to God, the Almighty, for providing me with the skills, opportunity and determination to complete this work. You carried me during times of great difficulty and I am forever grateful to You.

I would also like to express my appreciation and gratitude to Professor Martin Olivier for his professional insight, guidance and support. Thank you for all your assistance and for helping me to complete this journey. I hope to continue collaborating in the near future.

My thanks to Dr. Renier van Heerden for his continuous guidance and inspiring suggestions. All your contributions have been extremely valuable and improved the quality of the thesis.

Words would not express my gratitude to my parents, Marius and Ria Pieterse, for their continued love and unwavering belief in my success. Thank you for all the words of encouragement and support when I needed it most.

Thanks to all my friends and family for their love and support during many years of hard work. A special word of thanks to Tielman Meyer and Mari Pieterse.

I would also like to take this opportunity to say a big thank you to all my colleagues at the Cyber Warfare Research Group, Defence and Security, CSIR, for their advice and ideas. A special word of thanks to Dr Jabu Mtsweni, for providing me with valuable time to complete my thesis and to Ivan Burke, for his mathematical support.

I am grateful for the financial support that I received from the Council for Scientific and Industrial Research and the University of Pretoria.

To everyone else that has played a part in the success of this research, thank you very much.

Contents

List of Figures	vi
List of Tables	viii
1 Introduction	1
1.1 Motivation	2
1.2 Problem Statement	4
1.3 Scope of the Research	5
1.4 Objectives	6
1.5 Research Methodology	7
1.6 Terminology	8
1.7 Thesis Layout	10
1.8 Summary	12
2 Mobile Device Forensics	13
2.1 Mobile Device Classification	14
2.2 Mobile Operating Systems	17
2.2.1 Active Mobile Operating Systems	17
2.2.2 Discontinued Mobile Operating Systems	19
2.3 Need for Mobile Device Forensics	20
2.4 Purpose of Mobile Device Forensics	22
2.5 Forensic Examination of Mobile Devices	22
2.5.1 Sources of Mobile Device Data	23
2.5.2 Classification of Acquisition Tools	24

2.5.3	Mobile Forensic Toolkits	28
2.5.4	Forensic Investigation Process Models for Mobile Devices	30
2.6	Mobile Device Forensic Challenges	33
2.6.1	Diversity of Mobile Devices	33
2.6.2	Mobile Device Design	34
2.6.3	Improving Mobile Device Technology	34
2.7	Summary	35
3	Smartphone Forensics	36
3.1	Smartphone Operating Systems	37
3.1.1	Google Android	37
3.1.2	Apple iOS	41
3.2	Forensic Examination of Smartphones	43
3.2.1	Android Forensics	44
3.2.2	iOS Forensics	51
3.3	Challenges for Smartphone Forensics	57
3.3.1	Acquisition Challenges	57
3.3.2	Malicious Applications	58
3.3.3	Anti-Forensics	59
3.4	Summary	61
4	Smartphone Data	62
4.1	What is Smartphone Data?	63
4.2	Sources of Smartphone Data	63
4.2.1	SIM Card Data	64
4.2.2	Device-specific Data	64
4.2.3	User-created Data	65
4.2.4	Sensor Data	66
4.2.5	Application-related Data	66
4.3	Storage Structures for Smartphone Data	67
4.3.1	Plain Text Files	67
4.3.2	Extensible Markup Language (XML)	68

4.3.3	Property Lists	68
4.3.4	SQLite Databases	70
4.4	Value of Analysed Smartphone Data	72
4.5	Measures to Ascertain the Authenticity of Smartphone Data	74
4.6	Limitations of Authenticity Measures	76
4.7	Summary	77
5	Reference Architecture for Smartphone Applications	78
5.1	Existing Reference Architectures	79
5.2	Reference Architecture Derivation Process	81
5.2.1	Conceptual Architecture of Android Applications	82
5.2.2	Conceptual Architecture of iOS applications	84
5.2.3	Architectural Similarities	86
5.3	Reference Architecture for Smartphone Applications	87
5.3.1	Components of Reference Architecture	89
5.3.2	Behavioural Models of Smartphone Applications	90
5.3.3	Smartphone Application Characteristics	97
5.4	Modelling an Application using Reference Architecture	98
5.4.1	Modelling Expected Application Behaviour	99
5.4.2	Confirming Application Behaviour	101
5.5	Summary	103
6	The Authenticity of Smartphone Data	104
6.1	Authenticity	105
6.2	Authentic Smartphone Data	106
6.3	Requirements for Authentic Smartphone Data	107
6.3.1	End-user Behaviour	107
6.3.2	Smartphone Operational State	109
6.3.3	Smartphone Application Behaviour	110
6.3.4	External Environment	113
6.4	Smartphone Data Evaluation Model	114
6.4.1	Pre-evaluation Phase	115

6.4.2	Smartphone Evaluation Phase	116
6.4.3	Documentation Phase	123
6.5	Model Qualities and Characteristics	123
6.6	Summary	125
7	Classification of Evaluated Smartphone Data	126
7.1	Classification of Digital Evidence	127
7.2	Classification Model for Smartphone Data	128
7.2.1	Categorisation of the Requirements	129
7.2.2	Authenticity Score Calculation	131
7.2.3	Authentic Grading Scale for Smartphone Data	133
7.2.4	Completeness	135
7.2.5	Authenticity Classification	136
7.3	Smartphone Application Data Authenticity Classifier (SADAC)	138
7.3.1	Functional Requirements	138
7.3.2	Architectural Design	138
7.3.3	Technical Platform Specification	139
7.3.4	Interface Layout	139
7.4	Summary	141
8	Smartphone Data Evaluation	143
8.1	Manipulation of Smartphone Data	144
8.2	Generic Process for Smartphone Data Manipulation	145
8.2.1	Manipulation of Android Smartphone Data	146
8.2.2	Manipulation of iOS Smartphone Data	149
8.2.3	Formulation of the Generic Process	154
8.3	Attack Tree for Smartphone Data Manipulation	155
8.3.1	Attack Tree Construction	156
8.3.2	Attack Scenarios	158
8.4	Evaluation of Smartphone Data	159
8.4.1	Attack Scenario 1	159
8.4.2	Attack Scenario 2	164

8.5	Summary	167
9	Conclusion	170
9.1	Revisiting the Problem Statement and Research Questions	171
9.2	Main Contributions	172
9.3	Future Work	174
A	Publications	176
B	Abbreviations	178
C	Smartphone Application Data Authenticity Classifier	182
C.1	Authenticity Score	182
C.2	Completeness Interval	184
C.3	Authentic Grading Scale Construction	186
C.4	Summary	188
	Bibliography	189

List of Figures

1.1	Scope delineation	6
2.1	Classification of acquisition tools (Ayers et al. 2013)	25
3.1	Architectural layers of the Android platform (<i>Android Developers</i> 2017) .	38
3.2	Architectural layers of the iOS platform	42
3.3	Available partitions on an Android smartphone	46
5.1	The conceptual architecture of Android applications	83
5.2	The conceptual architecture of iOS applications	85
5.3	Visualisation of architectural similarities between Android and iOS applications	86
5.4	Reference architecture for smartphone applications	88
5.5	Modelling the behaviour of smartphone applications using a data flow diagram	91
5.6	Modelling the behaviour of smartphone applications using a finite state machine	94
5.7	Reference architecture for Android’s default messaging application	98
5.8	Modelling the behaviour of Android’s default messaging application . . .	99
5.9	Snapshot of the newest log file in the <code>/data/system/recent_tasks</code> folder	102
5.10	Snapshot of the SQLite database of the default messaging application . .	102
5.11	SQLite database timestamps after sending the text message	102
6.1	Pre-evaluation phase	115

6.2	Smartphone evaluation phase	117
6.3	End-user behaviour evaluation	118
6.4	Smartphone operational state evaluation	119
6.5	Smartphone application behaviour evaluation	120
6.6	External environment evaluation	122
7.1	Completeness scale	136
7.2	Authenticity Classification Graph	137
7.3	SADAC architectural design	139
7.4	The SADAC tool interface layout	140
8.1	Confirmation of root access on the Android smartphone	146
8.2	Unavailability of the <code>sqlite3</code> command-line program	147
8.3	Changes to the file structure of the SQLite database files (Android)	149
8.4	Confirmed installation of the Cydia application on the iPhone 7	150
8.5	Confirmed presence of the <code>sqlite3</code> program on the iPhone 7	151
8.6	Changes to the file structure of the SQLite database files (iOS)	153
8.7	Generic process for smartphone data manipulation	154
8.8	Attack tree for smartphone data manipulation	157
8.9	Authenticity classification of Attack Scenario 1 smartphone data	163
8.10	Authenticity classification of Attack Scenario 2 smartphone data	168

List of Tables

2.1	Hardware characteristics of mobile devices	15
2.2	Software characteristics of mobile devices	16
3.1	Common exploits to gain root access on Android smartphones	48
3.2	Availability of untethered jailbreak tools across iOS versions	56
4.1	Plist files containing valuable information	69
4.2	SQLite write-ahead log (WAL) Header Format (<i>SQLite 2017a</i>)	70
4.3	SQLite databases found on Android and iOS platforms	71
5.1	Architectural similarities between Android and iOS applications	87
5.2	State transition table	96
6.1	SQL queries to identify inconsistent records	111
7.1	Proposed Scale for Classifying Levels of Certainty (Casey 2011)	128
7.2	Categorisation of assessment points (End-user Behaviour)	130
7.3	Categorisation of assessment points (Smartphone Operational State)	131
7.4	Categorisation of assessment points (Smartphone Application Behaviour)	131
7.5	Categorisation of assessment points (External Environment)	132
7.6	Weight assignments for the classes	133
7.7	Authentic grading scale for smartphone data	135
8.1	Traces created by Attack Scenario 1	160
8.2	Attack Scenario 1 evaluation (End-user Behaviour)	160
8.3	Attack Scenario 1 evaluation (Smartphone Operational State)	161

8.4	Attack Scenario 1 evaluation (Smartphone Application Behaviour)	161
8.5	Attack Scenario 1 evaluation (External Environment)	161
8.6	Traces created by Attack Scenario 2	164
8.7	Attack Scenario 2 evaluation (End-user Behaviour)	165
8.8	Attack Scenario 2 evaluation (Smartphone Operational State)	165
8.9	Attack Scenario 2 evaluation (Smartphone Application Behaviour)	166
8.10	Attack Scenario 2 evaluation (External Environment)	166

Chapter 1

Introduction

Mobility is the future and the people of the 21st century are witnessing the fast-paced and continuous growth of mobile technology. The evolution of mobile devices started during the 1940s when the first mobile phones appeared. The design of the mobile phones follows radio technology and related instruments. From a design perspective, these mobile phones more closely resembled two-way radios, were enormous and lacked portability (Farley 2005). The turn of the millennium, however, saw great leaps in the growth of mobile technology and mobile phone functionality. Mobile devices, such as mobile phones, satellite navigation systems and tablet computers formed the foundation for future mobile technology. These advances in mobile technology and related devices allowed basic mobile phones to evolve into smart, compact and lightweight devices called smartphones.

Smartphones combine traditional mobile phone features with personal computer functionality (Kubi et al. 2011), offering enhanced connectivity, communication, multimedia, gaming and imaging capabilities. Such capabilities are a direct result of the ever-improving hardware components, development of information-rich and visually attractive smartphone applications, as well as mobile operating systems. These mobile operating systems, such as Google Android and Apple iOS, are the drivers of the rapid evolution of smartphones and allow for increasing popularity among end-users. The capabilities of smartphones, in conjunction with their ever-increasing storage capacity, allow for large quantities of digital data to reside in internal or external storage. The

reliance on and ubiquitous use of smartphones in the daily activities of end-users cause these devices to collect rich sources of digital data. Such digital data includes contacts, text and instant messages, call history, geographical data, electronic mail, web browsing history and multimedia activities (Ayers et al. 2013).

The diversity and sophistication of smartphones create increasing challenges and difficulties during criminal or civil cases. Various manufacturers produce smartphones, which use different mobile operating systems and hardware components, as well as support a wide variety of smartphone applications (Mohtasebi and Dehghantanha 2013). Mobile device forensics, therefore, emerged as a branch of digital forensics to address the challenges and difficulties surrounding the examination of mobile devices, such as smartphones. The interdisciplinary field of mobile device forensics describes the best practices and acceptable methods to collect and analyse digital data from mobile devices (Barm-patsalou et al. 2013). Data retrieved from smartphones can offer contextual clues about the end-user, the owner and user of the smartphone. Such clues can reveal whom the end-user knows and communicated with, highlight personality traits and pinpoint close associates (Kala and Thilagaraj 2013). A thorough inspection of all the available digital data produces necessary intelligence that becomes valuable should the smartphones be linked to criminal and civil.

Digital data is, however, vulnerable to unintentional change or explicit tampering, which raises issues with regards to its authenticity and reliability as evidence (Losavio 2005). It becomes, therefore, essential for digital forensic professionals to ensure and confirm the authenticity of the data.

1.1 Motivation

Smartphones have become constant companions for people across the world. According to a Gartner press release (*Gartner* 2018), the global sales of smartphones will increase by 2.1% in 2018. Their popularity is the result of continuous enhancement in functionality and ever-improving mobile operating systems. The widespread use of these portable and small-scale smartphones increases the possibility of such a device forming part of criminal or illegal activities (Barm-patsalou et al. 2013, 2018). The capabilities provided

by smartphones allow criminals to use the technology to facilitate various crimes, namely identity theft, financial fraud or cyberbullying, and avoid apprehension by erasing or fabricating certain digital traces.

Individuals with malicious intent are becoming aware of the importance of digital data found on smartphones and may use various techniques to alter the data and thwart the examination of such data. A knowledgeable individual may tamper with the digital data stored on a smartphone. Tampering can involve creating false transactions, wiping data from memory, purposefully modifying file extensions to impede mobile forensic tools or altering timestamps to falsify recorded activities (Ayers et al. 2013, Casey 2013). Furthermore, to conceal fraudulent actions individuals can forge digital alibis by simulating specific user activities or fabricating specific events (Albano, Castiglione, Cattaneo, De Maio and De Santis 2011).

It becomes clear that digital data stored on smartphones, referred to as smartphone data, is susceptible to change. Smartphone data can be altered, manipulated, fabricated or hidden by either faulty or error-prone applications, malware, end-users with malicious intent or anti-forensic techniques. Unknown alterations to existing data and failure to detect fabricated data can cause misleading results when analysed. It is, therefore, essential for digital forensic professionals to be able to eliminate anti-forensics, faults or malicious actions and first establish the authenticity of the smartphone data before formulating any conclusions relating to that data. Authenticity refers to the preservation of the data from the time it was first generated and the ability to prove that the integrity of the data has been maintained over the entire period (Losavio 2005, Casey 2011, Cohen 2012).

Many software applications include safeguards, such as audit logs and integrity checks (Thomson 2013), to ensure the validity and reliability of the data. Such safeguards can assist digital forensic professionals to ascertain the authenticity of digital data. Smartphones and their applications generally do not include sophisticated audit logs or similar safeguards. Meanwhile, commercial mobile forensic tools, such as Cellebrite Universal Forensic Device and FTK Mobile Phone Examiner, provide limited support in establishing authenticity (Verma et al. 2014). Solutions proposed by Verma et al. (2014), Govindaraj et al. (2015) and Pieterse et al. (2015) can assist digital forensic profes-

sionals with the evaluation of smartphone data, especially with regards to authenticity. Research by Verma et al. (2014) illustrate the preservation of date and timestamps of Android smartphones by capturing system generated values and storing these values in a location beyond the smartphone. Govindaraj et al. (2015) designed a solution, called iSecureRing, which permits a jailbroken iPhone to be secure and ready for examination by preserving timestamps in a secure location. Finally, Pieterse et al. (2015) proposed an authenticity framework for Android timestamps.

However, these solutions are either platform-specific, require additional software to be installed on a smartphone before examination or only focus on a specific subset of digital data such as timestamps. Clearly, there is a need for additional solutions that can enable digital forensic professionals to determine the authenticity of smartphone data. These solutions should be platform independent, support larger subsets of available smartphone data and eliminate the need for additional installed software.

1.2 Problem Statement

The vulnerable nature of smartphone data necessitates the ability to evaluate the authenticity of such data. This allows digital forensic professionals to eliminate and disregard unreliable smartphone data during examination. The lack of comprehensive solutions that can evaluate the authenticity of smartphone data emphasises the need for new and sophisticated techniques to evaluate and identify authentic smartphone data. This thesis consequently examines the problem of establishing the authenticity of smartphone data. The problem, as stated above, can be resolved by addressing the following questions:

- How does smartphone data originate?
- What is authentic smartphone data?
- What are the requirements smartphone data must adhere to be deemed authentic?
- How is smartphone data evaluated using the requirements?
- How can authenticity be classified based on the evaluation of the smartphone data?

By addressing these questions, this thesis lays the foundation for digital forensic professionals to evaluate and identify authentic smartphone data. The following section delineates the scope of the research.

1.3 Scope of the Research

The evolution of mobile devices has led to the development of various mobile operating systems. Renowned mobile operating systems, such as Symbian, BlackBerry OS and Windows Phone, were once widespread and widely used but have been discontinued and no longer involve any active development. Advances in mobile technology during the past decade gave rise to new and innovative smartphone platforms. This thesis examines the following smartphone platforms: Google Android (69.87% market share) and Apple iOS (28.82% market share) (*NetMarketShare* 2018). Although the thesis can also focus on other smartphone operating systems, the combined 98.69% market share of Android and iOS smartphones justify the emphasis on these platforms in this research.

Smartphones, running the Android or iOS operating systems, hold extensive collections of data. Smartphone data includes pre-generated data, data created due to the operation of the smartphone, such as executing applications, and data transferred to the smartphone by the end-user. The application-driven nature of smartphone operating systems permit the installation of additional, third-party applications. Both third-party and pre-installed smartphone applications are responsible for the generation of extensive collections of valuable smartphone data.

Application-related smartphone data primarily resides on internal storage. Other locations, such as subscriber identity module (SIM) cards and external or portable storage (micro SD cards) (Curran et al. 2010, Al-Hadadi and AlShidhani 2013), are available on certain smartphones and also store valuable smartphone data. SIM cards, however, present digital forensic professional with a small collection of smartphone data and the inconsistencies of externally stored data can impede structured evaluation. Smartphone data residing on internal storage is readily available, and the consistent storage structure of the data simplifies evaluation.

The focus of this thesis, as visualised in Figure 1.1, is to explore the authenticity

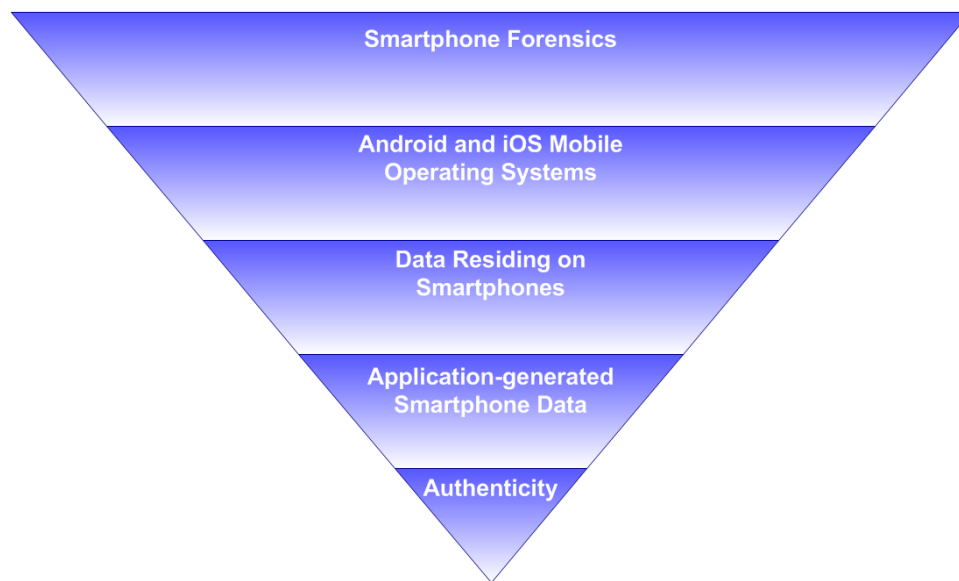


Figure 1.1: Scope delineation

of application-related smartphone data of Android and iOS smartphones that resides in internal storage kept directly on smartphones.

1.4 Objectives

The focus of this thesis is the evaluation of smartphone data and the identification of the authenticity of such data. Evaluation and identification of authentic smartphone data relies on the completion of several objectives. The primary objectives of this thesis are as follows:

- Model the structure and behaviour of smartphone applications for the Android and iOS platforms.
- Create a reference architecture for smartphone applications to capture expected behaviour.
- Establish requirements authentic smartphone data must adhere to based on the expected behaviour of smartphone applications.

- Capture the requirements in a model to assist digital forensic professionals with the evaluation of smartphone data.
- Design a classification model for evaluated smartphone data to determine the authenticity of the data.
- Development of a proof of concept software application that assists with the evaluation and classification of smartphone data.
- Confirm the practical use of the models to evaluate the authenticity of smartphone data.

To achieve the mentioned objectives, a specific research methodology is outlined in the following section.

1.5 Research Methodology

To address the questions raised in the problem statement (see Section 1.2) and meet the stated objectives (see Section 1.4), the thesis reviews various sources of literature that offers the necessary foundation for the presented research. The literature review captures the current state of mobile device forensics. This is followed by an overview of the concepts of smartphone forensics, focusing specifically on the Android and iOS platforms. Finally, emphasis is placed on the available sources of smartphone data and the structures responsible for storing the data.

The theoretical knowledge gained from the literature review forms the basis for modelling a reference architecture for smartphone applications. The reference architecture acts as an abstraction for smartphone applications developed for Android and iOS platforms by capturing the key components and behavioural interactions. The purpose of the reference architecture is to allow digital forensic professionals to more easily comprehend applications and understand how the associated data originated.

The central part of this thesis consists of the development of two models that allows for the evaluation and identification of authentic smartphone data. From the modelled reference architecture it is possible to infer key requirements and construct a model for

smartphone data evaluation. The smartphone data evaluation model provides digital forensic professionals with a guided approach for evaluating smartphone data. However, the smartphone data evaluation model only describes how to evaluate the data and not what the outcome of the evaluation is. Therefore, the thesis also formulates a classification model that uses mathematical formulas to assign an authenticity classification to the evaluated smartphone data based on the outcomes produced by the evaluation model.

The Smartphone Application Data Authenticity Classifier (SADAC) is a proof of concept software application build using the smartphone data evaluation and classification models. The purpose of SADAC is to demonstrate the practical use of the models to evaluate and classify smartphone data. Finally, the thesis validates the developed software program and created models by evaluating manipulated smartphone data. The manipulation of smartphone data involves exploratory experiments performed on both Android and iOS platforms to determine what form of manipulation is possible. Using the evaluation of the manipulated smartphone data as input for SADAC, the observed results confirm the competency of the models to establish the authenticity of smartphone data.

The outlined research methodology ensures the formulation of an appropriated solution for the established problem statement by resolving the key research questions and stated objectives.

1.6 Terminology

Most of the terms used in this thesis are defined in the relevant chapters. This section introduces a collection of specific terms used throughout the thesis, and the terms are defined in advance to avoid ambiguity.

Examination

The examination phase, which is one of seven digital forensic activities, involves the observation of traces and their characteristics, the recovery of information or content from data sources, as well as making the results available for the analysis phase. The

analysis phase will make effective use of the results produced by the examination phase and place the results in a technical context (Pollitt et al. 2018).

Classification

Classification, within the digital forensic realm, is one of the five core processes to address forensic questions and consists of two facets. The first is taxonomy, which is the scientific process that creates and defines classes. The second is ascription, which involves the process of recognising, with sufficient confidence, that digital data belongs to a specific class (Pollitt et al. 2018).

Evaluation

Within the digital forensic context, evaluation is best defined as a phase that produce a value that can be fed into a decision process. The provided value can then be used to establish the strength of evidence relating to a claim, or disprove the claim (Pollitt et al. 2018).

Digital Forensic Professional

The digital forensic professional is the individual responsible for examining digital devices (Ahmed et al. 2014, Hoelz and Maues 2017). Utilising the appropriate tools and techniques, the digital forensic professional must know how to effectively retrieve, analyse and interpret the collected digital data while also drawing accurate conclusions.

Smartphone End-User

The smartphone end-user is the owner or user of the smartphone and is primarily responsible for interacting with the device. Such interactions lead to the generation of smartphone data that can become valuable digital evidence should the smartphone form part of a criminal or civil case.

1.7 Thesis Layout

This thesis consists of 9 chapters, a bibliography and various appendices that supply additional material. The details of the remaining chapters and appendices are as follows:

- The purpose of **chapter 2** is to investigate and conduct an overview of the current state of mobile device forensics. Particular attention is given to the classification of mobile devices, as well as active and discontinued mobile operating systems functioning on these devices. The need for mobile device forensics is further emphasised, and the following aspects of mobile device forensics are discussed: sources of mobile device data, acquisition techniques, available mobile forensic tool kits and existing forensic investigation process models for mobile devices. The chapter concludes by presenting the current challenges still impacting mobile device forensics.
- **Chapter 3** lays the necessary foundation for understanding the concepts relating to smartphone forensics required in chapters **6**, **7** and **8**. The chapter describes the architectural design of the Android and iOS platforms and focuses on the key areas for each platform, namely the file system, partitions, device access (rooting/jailbreaking) and data acquisition. Existing challenges, such as anti-forensics and smartphone malware, as well as the purpose of smartphone forensics are also discussed.
- **Chapter 4** provides an overview of smartphone data, reviewing the various sources of smartphone data and the structures responsible for storing the data on smartphones. The forensic value obtained from analysed smartphone data is also presented. The chapter further discusses available techniques and solutions that can assist with the identification of authentic smartphone data. The limitations of these techniques and potential solutions are also highlighted.
- A new reference architecture for smartphone applications is proposed in **chapter 5**. The reference architecture is established by following the stipulated reference architecture derivation process and based on architectural similarities derived from conceptual architectures for Android and iOS applications. From the reference architecture, behaviour models are created that allow digital forensic professionals

to comprehend the creation of smartphone data. The chapter concludes with an evaluation of the reference architecture.

- The reference architecture presented in chapter 5 allows digital forensic professionals to have a better understanding of how application-related smartphone originates. **Chapter 6** focuses on the authenticity of such smartphone data. The chapter highlights authenticity and presents a formal description of authentic smartphone data. The requirements for authentic smartphone data are described and captured in a new smartphone data evaluation model. Finally, the chapter concludes by highlighting the benefits and limitations of the established model.
- The focus of **chapter 7** is on the classification of smartphone data and the formulation of a smartphone data classification model. The classification model is an extension of the smartphone data evaluation model introduced in the previous chapter and measures the authenticity of smartphone data. Also, the chapter introduces the newly developed proof of concept software application, called the Smartphone Application Data Authenticity Classifier, which automates the authenticity classification of smartphone data.
- **Chapter 8** assesses the effectiveness and efficiency of the models constructed in chapters 6 and 7 by evaluating manipulated smartphone data. The manipulation of smartphone data involves exploratory experiments conducted on different smartphone platforms (Android and iOS), which leads to the formulation of a generic process for smartphone data manipulation. Using the generic process, an attack tree is constructed, and various attack scenarios are deduced to manipulate smartphone data. A collection of these attack scenarios are applied at a theoretical level and the effects evaluated using the Smartphone Application Data Authenticity Classifier application.
- **Chapter 9** concludes the thesis by revisiting the problem statement and set research questions. The chapter also reflects on the main contributions of the research and highlights potential areas that can be considered for future research.

The following appendices are included:

- **Appendix A:** lists the conference and journal publications derived from this research.
- **Appendix B:** lists and defines all abbreviations used in this thesis.
- **Appendix C:** source code snippets of the Smartphone Application Data Authenticity Classifier application.

1.8 Summary

This chapter provided an introduction to this thesis and emphasised the importance of this research within the mobile device forensics field. Also, this chapter also presents the problem statement, delineates the scope of the research and stipulates the methodology to follow to meet set objectives. The chapter concluded with an outline of the thesis layout.

Chapter 2

Mobile Device Forensics

The fast-paced development and continuous growth of mobile technology support the evolution of mobile devices. The improvements of mobile device capabilities coupled with their popularity among end-users cause these devices to collect vast amounts of both personal and professional data (Barmpatsalou et al. 2018). The data collected by mobile devices can become invaluable sources of evidence, should the device be linked to criminal or illegal activities. Mobile device forensics, therefore, emerged as a branch of digital forensics to assist with the forensic investigation of mobile devices. Mobile device forensics is “the science of recovering digital evidence from a mobile device under forensically sound conditions using accepted methods” (Ayers et al. 2013). The interdisciplinary field of mobile device forensics consists of various techniques that apply to a wide range of devices (Barmpatsalou et al. 2013), including mobile phones, smartphones, tablet computers and satellite navigation systems. Although mobile device forensics often supports different mobile devices, the focus of this chapter is only on mobile phones and smartphones.

Sections 2.1 and 2.2 present the classification of mobile devices and describe the available mobile operating systems respectively. The necessity of mobile device forensics is discussed and emphasised in Section 2.3. Section 2.4 presents the purpose of mobile device forensics while Section 2.5 describes various aspects of the forensic examination of mobile devices. The latest challenges impacting mobile device forensics are highlighted in Section 2.6. Section 2.7 concludes the chapter.

2.1 Mobile Device Classification

Mobile devices are portable, lightweight, compact devices powered by rechargeable batteries. The basic set of features incorporated into mobile devices include a processor, read-only memory (ROM), random-access memory (RAM), microphone, speaker, digital signal sensor, interface and some hardware keys (Lohiya et al. 2015). Although all mobile devices tend to follow a similar blueprint, different mobile devices have different technical and physical characteristics (Ayers et al. 2013). Generally, mobile devices can be classified as basic mobile phones, feature mobile phones or smartphones.

Basic mobile phones are devices equipped with the necessary radio capabilities to support voice and text communication (Jansen and Ayers 2007, Kubi et al. 2011). These devices also provide a set of essential personal information management (PIM) applications, such as a phone book and calendar, to assist with both personal and professional activities (Kubi et al. 2011). Feature phones are advanced mobile phones that extend the capabilities offered by basic mobile phones and provide improved activities, such as browsing the Internet (Jansen and Ayers 2007, Ayers et al. 2013). Feature phones are also the first type of mobile devices to use NOR flash and RAM (Ayers et al. 2013). Smartphones, which are also called high-end mobile phones, merge the capabilities of feature mobile phones with those provided by a personal digital assistant (PDA) (Jansen and Ayers 2007). Besides the traditional telephone services (Casey 2011), smartphones offer functionality similar to that of personal computers, which includes the reviewing of electronic documents, enhanced Internet features and improved PIM functions (Jansen and Ayers 2007, Kubi et al. 2011). The functionality of smartphones is optimised to support a broad range of applications, reflected by the changes made to the user interface design, and a wide variety of connectivity methods, such as Wireless Fidelity (Wi-Fi), Bluetooth, Universal Serial Bus (USB) and near-field communication (NFC). Furthermore, smartphones also include additional sensors such as an accelerometer, magnetometer and gyroscope, to enhance user experience (Maus et al. 2011).

It is possible to further distinguish between the different classes of mobile devices by reviewing their hardware and software characteristics. A comparison of the hardware characteristics of mobile devices is presented in Table 2.1 (Jansen and Ayers 2007, Ayers et al. 2013, Lohiya et al. 2015).

Table 2.1: Hardware characteristics of mobile devices

	Basic Mobile Phone	Feature Phone	Smartphone
<i>Processor</i>	Speed limited	Speed improved	Speed superior
<i>Memory</i>	Capacity limited	Capacity improved	Capacity superior
<i>Display</i>	Grayscale	Colour	24-bit Colour High-Definition Display
<i>Camera</i>	None	Back Camera Still (photos)	Front/Back Still and Video Panoramic
<i>Text Input</i>	Numeric Keypad	Numeric Keypad Soft keyboard	Touch Screen QWERTY keyboard
<i>Wireless</i>	Infrared	Infrared Bluetooth	Bluetooth Wi-Fi NFC
<i>Location</i>	None	None	Global Positioning System

From a hardware perspective, basic mobile phones offer limited processor speed and memory capacity. The display of basic mobile phones is colourless, and interaction with the device occurs using a numeric keypad. Feature phones provide improved processor speed, increased memory and storage capacity. These devices also support a colour display and built-in camera. Smartphones are typically larger devices and provide a more prominent display, touch-sensitive screen and built-in wireless communication mechanisms (Ayers et al. 2013).

Table 2.2 compares the different software characteristics of mobile devices (Jansen and Ayers 2007, Ayers et al. 2013, Lohiya et al. 2015). Improvements of hardware components across the different classes of mobile devices lead to improved software that is available for these devices. Where basic mobile phones only supported, for example

Table 2.2: Software characteristics of mobile devices

	Basic Mobile Phone	Feature Phone	Smartphone
<i>Operating System</i>	Proprietary Closed	Proprietary Closed	Proprietary Open source
<i>PIM</i>	Simple phonebook	Phonebook Calendar	Enhanced phonebook and calendar Reminder list
<i>Applications</i>	None	Minimal (e.g., notepad and music player)	Advance (e.g., games and social media)
<i>Messaging</i>	Text messaging	Text messaging (images and sounds)	Text Messaging Multimedia Messaging
<i>Chat</i>	None	Chat via text messaging	Instant messaging
<i>E-mail</i>	None	Via text messaging	Via post office or Internet message access protocols
<i>Web</i>	None	Via wireless application protocol gateway	Direct HTTP

messaging and phonebook applications, smartphones provide advance PIM applications, e-mail functionality, instant messaging and web browsing via direct HyperText Transfer Protocol (HTTP) (Jansen and Ayers 2007, Ayers et al. 2013). These improvements also stem from the continuous growth of mobile operating systems.

2.2 Mobile Operating Systems

Mobile operating systems developed and evolved along with mobile devices, becoming an integral component. The core function of a mobile operating system is to act as the interface between the mobile applications and the underlying device hardware (Speckmann 2008). Various mobile operating systems have been developed to accommodate mobile devices and ensure fluent interaction between applications and device hardware. This section highlights active, as well as discontinued mobile operating systems.

2.2.1 Active Mobile Operating Systems

Active mobile operating systems are platforms that involve continuous development. New and improved versions of these mobile operating systems are regularly released, providing new functionality to remain competitive. Currently, the available and most used mobile operating systems are Android, iOS, Windows 10 Mobile and Tizen.

Android

Android is an open source, Linux-based mobile operating system (OS) developed by Google Inc. in conjunction with the Open Handset Alliance (OHA), which is a collaboration among mobile technology companies (Lee et al. 2009, Hoog 2011). The architecture of Android consists of six layers: Linux kernel, hardware abstraction layer, Android runtime, native C/C++ libraries, Java application program interface (API) framework and system applications (Lee et al. 2009). Android supports the installation of additional applications (delivered as Android package (APK) files), usually written in Java, and distributed by the Google Play Store (previously called the Android Market) (Holzer and Ondrus 2011, Vidas et al. 2011). It is also possible to distribute Android applications beyond the Play Store using third-party application markets (Holzer and Ondrus 2011). These applications further extend the functionality provided by the Android mobile operating system (Hoog 2011).

The openness of Google Android is due to the release of the source code through the Android Open Source Project (AOSP) on October 21, 2008 (Hoog 2011). The AOSP has, therefore, become the foundation of various other, lesser-known, mobile operating

systems, such as BlackBerry Secure, ColorOS, CyanogenMod, Emotion User Interface, FlymeOS, LineageOS, Indus OS and OxygenOS.

iOS

Apple Inc. designed and developed the iOS platform for their range of mobile devices, such as the iPhone, iPod Touch and iPad (Egele et al. 2011). This single mobile operating system for all Apples mobile devices allows users to easily migrate applications between these devices (Barrera and Van Oorschot 2011). The iOS platform is a proprietary OS (Tracy 2012), derived from OS X (Kanoi and Ingole 2013), and responsible for managing the device hardware, as well as providing fundamental technologies for native applications. To ensure a stable platform, the architecture of iOS comprises four layers: core-OS/kernel, core services, media support and cocoa touch (*Apple Developer 2017a*). The architecture supports the installation of additional applications, with the preferred development environment being Xcode and the Swift programming language (Tracy 2012). Developed iOS applications are bundled as IPA files and published in Apple's App Store, accessible via iTunes (Egele et al. 2011). Only accepted and signed iOS applications are made available on the Apple App Store.

Windows 10 Mobile

Microsoft introduced the Windows Phone mobile OS to regain the market share from the leading competitors, mainly Android and iOS (Gronli et al. 2014). In 2015, Microsoft released the latest edition of the Windows family: Windows 10 (Tidrow et al. 2015). The most significant change introduced by Windows 10 is the cross-platform model. This cross-platform model allows the same code base and user experience to be shared across a wide range of Windows devices including desktop computers, tablets and smartphones (Tidrow et al. 2015). Windows mobile devices run a modified version of Windows 10, called Windows 10 Mobile (Halsey 2015). Windows 10 Mobile support universal applications, allowing desktop applications also to be supported by Windows mobile devices (Tidrow et al. 2015). End-users can download applications for Windows mobile devices from the Windows Store (Halsey 2015). There is, however, no support provided by Windows 10 Mobile for classic Win32/.NET applications (Halsey 2015, Tidrow et al. 2015).

Windows 10 Mobile also offers an enterprise edition that provides additional device management and security capabilities (Tidrow et al. 2015). Since October 2017, Microsoft only supports the development of maintenance releases and patches for Windows 10 Mobile.

Tizen

Tizen is an open source Linux-based mobile operating supported by the Linux Foundation, Intel, Samsung, Huawei and Fujitsu. Tizen targets various mobile device categories, such as smartphones, tablets, netbooks and in-vehicle infotainment systems (Gadyatskaya et al. 2014, Vashisht and Vashisht 2014). To accommodate such a vast collection of mobile devices, Tizen follows a simple architecture. The lowest layer represents the Linux kernel and device drivers. The middle layer contains the Tizen code, which provides the basic functionality required by the web and native frameworks. The top layer captures the user applications (Gadyatskaya et al. 2014). Tizen supports native, web and hybrid (a combination of native and web) developed applications (Gadyatskaya et al. 2014). The drive behind the development of the Tizen mobile operating system is to create an alternative but less expensive mobile platform (Vashisht and Vashisht 2014).

2.2.2 Discontinued Mobile Operating Systems

An extensive collection of once popular and widely distributed mobile operating systems have been discontinued. These discontinued platforms no longer involve any development and ceased releasing new versions. Renowned discontinued mobile operating systems include Symbian, BlackBerry OS, Bada, Palm OS and Firefox OS.

Symbian is proprietary software that was developed in 1998 (Vaughan-Nicholas 2003) for Nokia's collection of mobile devices. Design as a multitasking operating system, Symbian delivered advanced features and supported the complex requirements of network protocols worldwide (Babin 2008). Symbian was discontinued when the platform was replaced by Windows Phone after Microsoft acquired Nokia. BlackBerry OS is a proprietary mobile operating system developed by Research In Motion (Lin and Ye 2009). This

platform specifically targeted enterprise customers and corporate professionals, providing features such as push mail, instant messaging and file sharing (Lohiya et al. 2015). Organisational changes led to the replacement of BlackBerry OS by BlackBerry 10. Bada was developed by Samsung Electronics for smartphones and tablet computers (Nosrati et al. 2012). The architecture of Bada is based on the open source software of Linux (Yun et al. 2016) and native applications for the platform were developed using C++ (Nosrati et al. 2012). Samsung ceased development of Bada, favouring the Tizen platform instead. Palm OS, also known as Garnet OS, is a proprietary mobile operating system developed by Palm, Inc. in 1996 (Nosrati et al. 2012). The platform was designed to allow natural interaction using a touchscreen interface and provided a suite of necessary PIM applications (Nosrati et al. 2012). Firefox OS was an open source mobile operating system released by Mozilla (Gadyatskaya et al. 2014). The architectural design of Firefox OS is based on a modified Android stack and, therefore, Android mobile devices can support a ported version of Firefox OS (Gadyatskaya et al. 2014). All development of Firefox OS was, however, stopped in 2016.

Although the above-mentioned mobile operating systems have been discontinued, end-users may still use mobile devices supporting these operating systems. It is, therefore, necessary to remain vigilant of the potential presence of these mobile operating systems during the examination of mobile devices.

2.3 Need for Mobile Device Forensics

Advances in mobile technology and related mobile operating systems ensure the popularity and relevance of mobile devices in contemporary societies. Mobile devices form an essential part of people's lives and are used for both personal and professional purposes (Jansen and Ayers 2007, Casey 2011, Barmpatsalou et al. 2018). Mobile devices can also assist with the execution of criminal activities, such as credit card fraud, spam distribution or child pornography (Farjamfar et al. 2014). This close relationship between an individual and their mobile device often leads to a rich collection of valuable digital data (Abalenkovs et al. 2012), which can reveal far more details about the end-user than traditional desktop or laptop computers (Barmpatsalou et al. 2018). The collected data

can provide a digital forensic professional with a good source of evidence (Adams et al. 2008, Yates 2010), should the mobile device be linked to criminal events (Casey 2011).

The data stored on and associated with mobile devices can help digital forensic professionals to form hypotheses and answer crucial questions during examinations (Casey 2011). Extracting contacts, call history (dialled, received and missed calls) and messages (text, multimedia and instant) provide an overview of an individual's social, work and family networks (Casey 2011). Further analysis of text messages, as well as e-mails and electronic documents, offer context and can reveal the main topic of discussion between individuals. Available geographical information stored on mobile devices, such as exchangeable image file format (EXIF) data embedded in digital photographs, can be used to pinpoint the location of the mobile device during a period of interest (Casey 2011). Location data also highlights the mobile device user's travel habits and reveal potential places of interest. Timestamps associated with all forms of data stored on mobile devices support the creation of timelines and allow digital forensic professionals to infer the time when events took place (Casey 2011). The ability to capture and analyse the data available on mobile devices provides forensic professionals with a powerful investigative capability.

Based on the variety of available mobile device data, it is possible to confirm that these devices may contain data of potential forensic value (Abalenkovs et al. 2012). Collection and presentation of this data allow digital forensic professionals to prove or disprove the occurrence of illegal events, pinpoint the location of suspects/victims (Yates 2010) or assist with the prosecution of individuals responsible for criminal activities (Curran et al. 2010). The existing generation of mobile devices is, however, sophisticated and increasingly tricky to examine (Curran et al. 2010). Mobile devices are portable and, therefore, require specialised hardware components to sustain the mobility aspect of these devices (Jansen and Ayers 2007). There is currently no de facto design for mobile devices, which causes hardware architectures to be as diverse as there are different mobile operating systems (Osho and Ohida 2016). The proliferation of mobile devices and the diversity of both software and hardware components for these devices produce difficulties during examinations. These difficulties led to the establishment of a specialised field, called mobile device forensics.

2.4 Purpose of Mobile Device Forensics

Mobile device forensics, a branch of digital forensics, emerged to address the difficulties surrounding the extraction, analysis and presentation of data retrieved from mobile devices. The purpose of mobile device forensics is to support the technical examination of mobile devices and the retrieval of data from these devices under forensically sound procedures (Curran et al. 2010).

The extraction and analysis of such a collection of data lead ultimately to the identification and presentation of digital evidence. Köhn (2012) defines digital evidence as “data or information of probative value stored on or transmitted by a digital device that supports or refutes a hypothesis formulated during a digital forensic investigation relied upon in court to determine the outcome of a legal question”. Digital forensic professionals can use the collected digital evidence to evaluate the following objectives (Jansen and Ayers 2007, Ayers et al. 2013):

- Gather information about the individuals involved in events (Who?)
- Create context and determine the exact nature of the events that occurred (What?)
- Construct a timeline of events (When?)
- Uncover information that presents a motive for events (Why?)
- Determine the location of the events (Where?)
- Identify how events unfolded (How?)

The gathered digital evidence and the formulation of answers to the specified objectives above all aid in solving criminal or civil cases (Al-Hadadi and AlShidhani 2013).

2.5 Forensic Examination of Mobile Devices

The primary purpose of mobile device forensics is to simplify the examination of mobile devices and assist digital forensic professionals with the extraction, preservation and analysis of data acquired from mobile devices. The following sections identify the sources

of data available on mobile devices and the tools that can be used to extract the data from these devices. Furthermore, these following sections also present existing mobile forensic toolkits that can assist with the extraction of mobile device data and describes recommended process models to follow when mobile devices form part of regulatory matters, security incidents, civil or criminal cases.

2.5.1 Sources of Mobile Device Data

The content-rich features and increasing storage capacity of modern mobile devices allow for the accumulation of vast sources of data, which can become valuable evidence for a criminal or civil case. Mobile devices collect and store these sources of data in three areas: SIM card, internal (device or handset) memory and external (portable or removable) memory (Punja and Mislán 2008, Curran et al. 2010, Casey 2011, Kubi et al. 2011, Al-Hadadi and AlShidhani 2013).

A SIM card is a smart card that is synonymous with mobile devices and responsible for storing subscriber and network-related information (Curran et al. 2010). Traditionally, a SIM card contains a processor and electronic erasable programmable read-only memory (EEPROM) with an encryption key to store and protect the data (Omeleze and Venter 2013). Stored data on SIM cards include phonebook/contacts, text messages, last dialled number (LDN), the position of last usage and service-related information (Punja and Mislán 2008, Kubi et al. 2011).

The limited storage capacity of SIM cards restricts the availability of collected data. Advances in mobile device technology have, however, led to an increase in the storage capacity of handset/device memory. Handset/device memory, more commonly referred to as internal memory, is the internal storage space of a mobile device and holds the operating system, pre-installed applications and other standalone files. Potential evidence that can be retrieved from the internal memory of mobile devices includes: call history, messages (text, multimedia and instant), e-mails, Internet history (browsing records and bookmarks) and files related to third-party applications (Jansen and Ayers 2007, Punja and Mislán 2008, Kubi et al. 2011). Besides user-related evidence, the internal memory also stores system-related information (device model, OS version and International Mobile Equipment Identity (IMEI) number) and telecommunication settings (mobile

network type and mobile network state) (Punja and Mislán 2008).

The final storage area for mobile device data is the external memory. External memory is removable memory that extends the internal storage capacity of a mobile device. External memory, such as Micro SD cards, primarily holds large amounts of user-related data, such as images audio files, videos and electronic documents (Punja and Mislán 2008, Al-Hadadi and AlShidhani 2013). Specific third-party applications also use external memory to store application-related information. It is important to note that external memory is, however, not supported by all mobile devices.

The variety, as well as distribution of data stored on mobile devices, can complicate the proper retrieval of the data as potential evidence. Acquisition tools, therefore, emerged to assist with and simplify the process of retrieving digital data from mobile devices.

2.5.2 Classification of Acquisition Tools

Extracting the available data stored on mobile devices and using the extracted data as evidence is critical to the success of a criminal or civil case. Mobile device data is obtained and collected from mobile devices using various acquisition tools. Figure 2.1 illustrates the categorisation of acquisition tools available for mobile device data retrieval. As the pyramid is traversed from the bottom level to the top, the acquisition tools become more technical, time-consuming and expensive (Ayers et al. 2013).

Level 1, manual acquisition, involves the retrieval of data directly from the mobile device by interacting with the device using the keyboard or touchscreen (Lohiya et al. 2015). Although not the preferred acquisition methodology to collect data, it remains the only method to acquire data when there is either no compatible software available or damage to the external interfaces (USB or wireless) (Jansen and Ayers 2007, Lohiya et al. 2015). Manual acquisition of mobile device data is a time-consuming process, prone to human error and provides no option for the retrieval of deleted data (Barmpatsalou et al. 2018). The use of this approach is, therefore, only recommended for examinations involving damaged or unfamiliar mobile devices.

Level 2, logical acquisition, is the extraction of data collected in the logical file allocation storage area (file system partition) of mobile devices (Jansen and Ayers 2007,

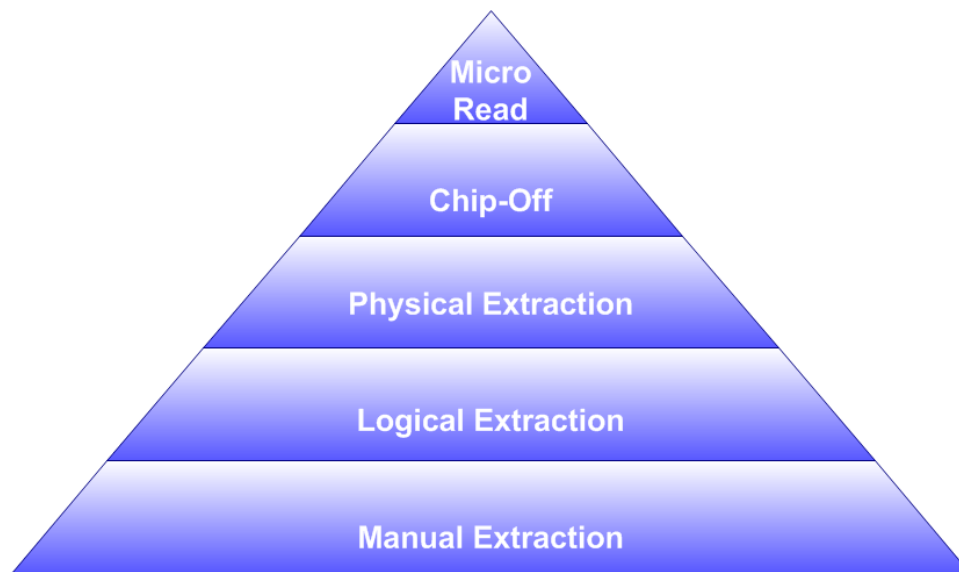


Figure 2.1: Classification of acquisition tools (Ayers et al. 2013)

Omeleze and Venter 2013). During the logical acquisition of mobile device data, a bit-by-bit copy of logical stored objects, such as directories and various file types, is acquired (Jansen and Ayers 2007, Bader and Baggili 2010). The logical image is created using a series of commands (backup or debugging functions) that are exchanged over an interface established between a computer and the mobile device (Park et al. 2012, Lohiya et al. 2015). The interface usually involves either a wired (USB) or wireless (infrared, Bluetooth or Wi-Fi) connection to access and retrieve the logical stored objects (Lohiya et al. 2015). Apple mobile devices and specific Android-supported Sony device models support logical acquisition using offline backups collected via installed software applications (Apple iTunes and Sony PC Companion respectively). Logical acquisition is easy to perform, requires no particular expertise to accomplish and is a non-destructive process since disassembly of the mobile device is not required. However, logical acquisition only retrieves a limited collection of existing files and directories from a mobile device and does not recover previously deleted data (Cheema et al. 2014, Afonin and Katalov 2016). Logical acquisition is, therefore, only the preferred method to acquire data when physical acquisition is not feasible.

Over-the-air or cloud acquisition is a form of logical acquisition and retrieves data

from stored backups in the cloud environment. Although cloud backups tend to contain limited information, over-the-air acquisition is the only available acquisition method for mobile devices with full-disk encryption or hardware locks, such as 64-bit Apple iPhones running iOS version 8 or higher. Over-the-air acquisition is easy to use, does not require physical access to the mobile device and operates independently of device models or operating system versions. Acquisition from the cloud environment, however, requires access to the end-user's authentication credentials. Needing access to the necessary credentials and the limited data retrieved by over-the-air acquisition impact the feasibility of this acquisition method (Afonin and Katalov 2016).

Level 3, physical acquisition, implies a bit-by-bit copy of the entire physical store (raw disk image) of a mobile device (Jansen and Ayers 2007, Bader and Baggili 2010). Physical acquisition collects a complete copy of all the data stored on the mobile device, including the recovery of previously deleted or lost data (Omeleze and Venter 2013). Physical acquisition tools directly deal with the raw data stored in flash memory on a mobile device (Lohiya et al. 2015). Physical acquisition provides the best balance between extraction speed, ease of use and amount of data acquired (Afonin and Katalov 2016). Therefore, physical acquisition tools remain the preferred method to obtain data from a mobile device since the method can extract the maximum amount of data from the device. Physical access to the mobile devices may not always be feasible (inability to root or jailbreak) and devices locked by an unknown pin or passcode can impede physical acquisition.

A commonly used physical acquisition tool connects via the Joint Test Action Group (JTAG) interface (Ayers et al. 2013). JTAG was first established in the 1980s, and the interface was created to assist with the wiring and interconnects on printed circuit boards (Abalenkovs et al. 2012). This interface provides an advance acquisition procedure utilising the access ports to acquire raw data stored on the mobile device. The acquisition procedure requires specialised equipment and a matching device-specific JTAG cable to retrieve available data from flash memory. Although technically complex, JTAG acquisition can collect data from both locked or damaged mobile devices and permits the collection of data from mobile devices running proprietary operating systems (Afonin and Katalov 2016). Another form of physical acquisition is Hex Dumping, which in-

volves the use of a flasher box to create an image of the RAM in hexadecimal format (Luttenberger and Creutzburg 2011).

Level 4, Chip-off, involves the physical removal (desoldering) of the flash memory chip from the devices' central processing unit (CPU) (Jansen and Ayers 2007, Omeleze and Venter 2013, Afonin and Katalov 2016). This method is achieved by physically removing the NAND logic gate and viewing the NAND memory using an appropriate reader (Jansen and Ayers 2007, Omeleze and Venter 2013). Chip-off provides high acquisition speed, acquires data from locked or damaged mobile devices and supports devices without JTAG ports (Abalenkovs et al. 2012). Chip-off acquisition provides best results when using unencrypted mobile devices but remains a destructive process that requires the physical disassembly of the device (Afonin and Katalov 2016). In-System Programming (ISP) is an advanced, non-destructive variation of the Chip-off acquisition. During the acquisition process, ISP attempts to acquire the data of embedded Multimedia Card (eMMC) memory without physically removing the chip. This acquisition method is only available for mobile devices utilising the eMMC or embedded multi-chip package ball grid array chips. Similar to Chip-off acquisition, ISP can also acquire data from locked mobile devices at high speed and supports devices without JTAG ports. ISP cannot, however, bypass encryption, remains a labour-intensive acquisition process and also requires disassembling of the mobile device to access the chip (Afonin and Katalov 2016).

Level 5, micro read, presents the final classification of acquisition tools available for mobile devices. Micro read records the physical observation of the NAND or NOR gates on the chip using an electron microscope (Ayers et al. 2013, Lohiya et al. 2015). The extreme technicalities required to successfully perform micro read cause this particular level of acquisition to only be attempted for high profile cases when all other acquisition methods have been exhausted (Ayers et al. 2013).

The acquisition tools mentioned above are commonly used to perform postmortem or dead forensics. Another option to acquire and extract the data stored on mobile devices is live acquisition. Live acquisition involves the near real-time extraction of content using either volatile memory or network data (Barmpatsalou et al. 2018). The key disadvantage of live acquisition is the potential manipulation of data due to the

direct interaction with an active mobile device (Luttenberger and Creutzburg 2011).

All of the presented acquisition tools quicken and simplify the retrieval of data from mobile devices. Acquiring the available mobile device data is vital for the overall success of a digital forensics examination. Therefore, one or more of the discussed acquisition tools usually forms part of mobile forensic toolkits.

2.5.3 Mobile Forensic Toolkits

The primary purpose of any mobile forensic toolkit is to assist digital forensic professionals with the extraction, recovery, analysis and examination of mobile device data. Due to the diversity of mobile devices and mobile operating systems, it is not feasible to design a mobile forensic toolkit that is applicable across all mobile devices. The toolkits are usually narrowed to specific models, operating systems or hardware architectures (Jansen and Ayers 2007). Mobile forensic toolkits must conform to the following prerequisites (Mokhonoana and Olivier 2007):

- The toolkit must have a minimal impact on the stored data and change the data as little as possible.
- Extract all of the data available on the mobile device.
- Minimise the interaction between the digital forensic professional and the mobile device.

There exists a wide variety of commercial and free mobile forensic toolkits. This section briefly discusses the most prevalent and widely used toolkits.

Elcomsoft Mobile Forensic Bundle is a complete mobile forensic toolkit that includes support for physical, logical and over-the-air acquisition tools. The toolkit supports iOS, Windows Phone 8-10 and BlackBerry 10 mobile devices. The toolkit allows for the recovery of backup passwords, as well as decrypting the encrypted backups. Furthermore, Elcomsoft Mobile Forensic Bundle can also decrypt and display WhatsApp communication histories, acquire information from Google accounts and view deleted messages (Elcomsoft 2017).

Mobile Phone Examiner Plus (MPE+) from AccessData is a standalone mobile device investigation solution that provides enhanced examination and analysis capabilities. MPE+ supports a vast collection of mobile devices: Android, Windows Mobile, BlackBerry, iOS (up to version 8.0), legacy phones and Chinese embedded devices. Acquisition tools to acquire the available mobile device data do not form part of the MPE+ toolkit but are available in a separate AccessData solution called nFIELD. Some of the features of MPE+ include visualisation tools, SQLite browser, plist viewer, Hex interpreter and SIM card support (*AccessData* 2017).

Oxygen Forensic Analyst is a single component of the Oxygen Forensic Suite and can retrieve data from iOS, Android, BlackBerry 10 and Windows Phone 8 mobile devices. The functionality provided by Oxygen Forensic Analyst includes the construction of timelines and social graphs, viewing all messages (text, multimedia, e-mails and iMessages), extracting EXIF data and displaying device-related information. Oxygen Forensic Analyst also provides a file browser that can be used to access and analyse photos, videos, documents and application-related databases (*Oxygen Forensics* 2017).

Cellebrite offers a collection of cross-platform mobile device solutions. The most prominent solutions are Universal Forensic Extraction Device (UFED) Pro Series, Field Series and Analytics. Cellebrite's UFED analytics automates data analysis (such as call history, messages, applications and standalone files) and case management tasks while supporting both smartphones and legacy devices (*Cellebrite* 2017).

MSAB's XRY product is a mobile forensic toolkit for smartphones, 3G modems, music players and satellite navigation units. XRY supports the logical and physical acquisition of data from various smartphones such as Android, iPhone and BlackBerry devices. The data retrieved by XRY includes, for example, Voice over Internet Protocol (VoIP) calls, Global Positioning System (GPS) mapping information, instant messages, call history and standalone files (*MSAB* 2017).

MOBILedit allows digital forensic professionals to view, search and retrieve mobile device data. The data retrieved by MOBILedit includes user created (for example call history, phonebook, messages and application data) and device-related information. MOBILedit supports various smartphones (Android, iPhone, Windows Phone and BlackBerry) and can bypass authentication mechanisms and backup encryption. In addition,

MOBILedit also includes functionality to retrieve Skype, Gmail and Facebook contacts without knowing the account passwords (*MOBILedit* 2017).

Finally, Paraben's E3 Device Seizure is the only mobile forensic toolkit that supports all mobile operating systems. E3 Device Seizure can retrieve various forms of mobile device data, offers JTAG capabilities and supports both logical and physical acquisition tools. Also, E3 Device Seizure supports the analysis and cloning of SIM card data (*Paraben* 2017).

The mobile forensic toolkits described in this section provide digital forensic professionals with the necessary support to retrieve and analyse the rich sources of digital data stored on mobile devices. To assist with the adequate collection of digital data from mobile devices according to sound principles, researchers have proposed various forensic investigation process models.

2.5.4 Forensic Investigation Process Models for Mobile Devices

Process models for mobile devices attempt to bring structure to and simplify the forensic investigation of such devices. This section describes existing process models that offer guidance to digital forensic professionals during the forensic investigation of mobile devices.

Ramabhadran (2007) proposed a forensic investigation process model specifically for Windows mobile devices. The process model was developed to assist both digital forensic professionals and organisations during investigations by establishing the necessary policies and procedures to follow. The process model consists of 12 phases: preparation, securing the scene, survey and recognition, documenting the scene, communication shielding, volatile evidence collection, non-volatile evidence collection, preservation, examination, analysis, presentation and review. The phases of this process model incorporate standard practices and techniques, which makes the model applicable to both corporate and law enforcement investigations. Although specifically aimed at Windows mobile devices, applying minimal changes will make this process model suitable for other mobile devices (Mohtasebi and Dehghantanha 2013).

Previously designed process models, such as the model for Windows mobile devices, were not well-suited for Symbian smartphones. Yu et al. (2009), therefore, proposed a

Symbian-specific forensic investigation process model that consists of five stages: preparation and version identification, remote evidence acquisition, internal evidence acquisition, analysis and presentation, and review. Absent from this process model was essential steps such as the preservation or transferral of collected evidence.

Husain et al. (2010) introduced a simple and cost-effective forensic analysis framework for iPhone devices. The framework consists of three phases: forensic data acquisition, data analysis and data reporting. The forensic data acquisition phase collects backup data using the iTunes backup utility while the data analysis phase parses the binary backup files into plist files and databases (see Section 4.3). Finally, the data reporting phase collects and presents the appropriate findings.

With the release of the Samsung Star 3G phones, Parvez et al. (2011) designed a framework to conduct the forensic investigation of the Star 3G phones. The framework formulates various procedures to allow a digital forensic professional to perform a full investigation of Samsung Star 3G phones. The procedural components included in the framework are authorisation, first response, device transportation, live acquisition, maintenance and evidence analysis. Although specifically designed for Star 3G phones, certain procedures can assist with the forensic investigation of other mobile devices.

A non-platform specific smartphone forensic investigation process model was introduced by Goel et al. (2012). This process model serves as a benchmark for investigating smartphones and consists of 14 stages: preparation & securing the scene, documentation, PDA mode, communication shielding, cell site analysis, evidence collection (volatile memory, non-volatile memory and offset/cloud memory), preservation, examination, analysis, presentation and review. The cell site analysis stage collects the geographical location of the mobile device based on previously made calls, messages or downloads. The offset/cloud memory analysis collects data stored online in the cloud environment. The smartphone forensic investigation process model provides a more comprehensive process model.

The harmonised digital forensic investigation process model, designed by Valjarevic and Venter (2012), is a standardised model aimed at harmonising existing process models. It forms a comprehensive model that incorporates all the benefits of current models and comprises of 12 phases: incident detection, first response, planning, prepara-

tion, incident scene documentation, potential evidence identification, potential evidence transportation, potential evidence storage, potential evidence analysis, presentation and conclusion. The process model also includes various parallel actions that are performed simultaneously to an investigation to ensure documentation and preservation of evidence. The harmonised digital forensic investigation process model was used to investigate BlackBerry (Mumba and Venter 2012) and Android (Omeleze and Venter 2013) mobile devices, which showed the adequate accommodation of mobile devices from different manufacturers.

Dancer et al. (2013) proposed a theoretical process model for smartphones based on an abstract model. The process model includes five phases: transportation, classification, analysis, interpretation and retention. The purpose and aim of this particular process are to be cross-platform and applicable to all smartphones, regardless of the underlying operating system.

A conceptual evidence collection and analysis methodology for Android devices was proposed by Martini et al. (2015). The methodology comprises eight steps: set-up a bootloader for live operating system, set-up a live operating system in memory, collect a physical image of device partitions, examine application files in private storage, examine application files on external storage, examine application databases, examine and analyse accounts data and analyse the application. The steps included in the methodology provides techniques to bypass device/operating system security features and collect a physical bit-by-bit forensic image. The structure and collection of steps allow the methodology to adhere to the following characteristics: forensic sound principles and practical device agnostic processes.

Besides the traditional forensic investigation process models, researchers also proposed standard operating procedures to assist with the digital forensic investigation of mobile devices. Lin et al. (2011) proposed the smartphone digital evidence forensics standard operating procedure that consists of the following four phases: conception, preparation, operation and reporting. Jansen and Ayers (2007) proposed the National Institute of Standards and Technology (NIST) smartphone standard operating procedure, which also consists of four phases: preservation, acquisition, examination and analysis (Jansen and Ayers 2007).

From the descriptions above, there is no standard or widely accepted forensic investigation process model for mobile devices, which is one of the many challenges influencing mobile device forensics. The diversity of process models is a direct result of the range of mobile devices, as well as the fast growing pace of mobile technology. It is, therefore, the responsibility of the digital forensic professional to select the most appropriate process model to conduct the digital forensic investigation of a mobile device.

2.6 Mobile Device Forensic Challenges

The field of mobile device forensics is still a relatively new and evolving branch of digital forensics, which causes the field to continuously face various challenges such as the diversity of process models. This section captures multiple other and current difficulties influencing mobile device forensics. It is possible to organise these difficulties according to the following categories: diversity of mobile devices, mobile device design and improving mobile device technology.

2.6.1 Diversity of Mobile Devices

The focus of Section 2.2 was on the diversity of mobile operating systems. The discussions in this section revealed that there is currently no de facto operating system for mobile devices. This lack of standard practice leads to the development of various operating systems for different mobile devices (Jansen and Ayers 2007, Yates 2010). There is also no standard that defines the default storage format and locations for storing different types of data on mobile devices (Osho and Ohida 2016). The lack of standardised methods to store data allows mobile device manufacturers to use different file systems, data formats and storage locations to store mobile device data (Brothers 2009, Casey 2011). Such diversity of both mobile operating systems and file system varieties complicates the acquisition of mobile device data.

2.6.2 Mobile Device Design

The portability of mobile devices is key to the overall physical design, which causes the devices to rely on a battery to supply continuous power to the device. As emphasised by the various process models discussed in Section 2.5.4, it is necessary to block radio signals to a mobile device during examinations to perform forensic analysis (Brothers 2009, Osho and Ohida 2016). Signal blockage prevents users from remotely changing or destroying mobile device data but quickly drains battery power (Brothers 2009, Osho and Ohida 2016). Should a mobile device shutdown, there is a possibility of data loss, as well as the activation of security mechanisms when restarted (Osho and Ohida 2016). Security mechanisms (mobile device locks or encryption) form part of mobile devices to supply data protection. Activation of such security mechanisms confines access to mobile device data (Brothers 2009) and also prevents the logical acquisition of the data (Osho and Ohida 2016). These challenges are due to the sophisticated design of mobile devices. Digital forensic professionals must always be aware of the impact these challenges can have during examinations.

2.6.3 Improving Mobile Device Technology

The rapid rate of change and enhancement of mobile device technology (Barmpatzidou et al. 2013) increases the sophistication of mobile devices. This is especially true for mobile operating systems, which have short product cycles and release new versions with substantial changes annually (Osho and Ohida 2016). Between these releases, mobile operating system vendors also periodically release new patches and minor upgrades (Osho and Ohida 2016). Besides the improvements of mobile operating systems, manufacturers also annually release new mobile devices that incorporate improved hardware components. Hardware improvement includes optimised processors, growing storage capacity (both internal and external memory) and various connectivity options, such as NFC, Wi-Fi, Bluetooth and Long-Term Evolution (LTE) (Goel et al. 2012, Osho and Ohida 2016). Improvements in mobile device technology affect the functionality and efficiency of existing mobile forensic toolkits since these toolkits struggle to remain up to date with the constant improvements.

2.7 Summary

This chapter reviewed the current state of mobile device forensics, which provides an updated and modernised view of mobile device forensics. The first part of the chapter gave particular attention to the classification of mobile devices and the current state of both active and discontinued mobile operating systems. The popularity and improved functionality of mobile devices allow the devices to form part of criminal activities and, thus, become a valuable container of potential evidence. The presence of such digital evidence established the need for and the motivation behind the emergence of the mobile device forensic discipline. With regards to the forensic examination of mobile devices, this chapter focused on the familiar sources of mobile device data and the available acquisition tools, as well as mobile forensic toolkits that can assist with the collection of relevant digital data. Furthermore, the varied collection of current forensic investigation process models, designed specifically for mobile devices, were also presented. The diversity of these process models, along with the continuous improvements of mobile device technology and design, creates constant challenges for mobile device forensics. Smartphone forensics, therefore, originated as a sub-discipline to address these challenges by exclusively focusing on the forensic investigation of smartphones.

Chapter 3

Smartphone Forensics

The past decade witnessed the rapid development of mobile technology, with smartphones being the current trendsetters. Their current prominence is directly related to the provided capabilities and functionality, which closely resemble a personal computer. Bundled with a complete operating system, improved connectivity and communication functions, and the option of installing additional applications, smartphones have become powerful mobile devices. The popularity and consumerisation of smartphones have alerted the forensic's community to focus on the technical capabilities of these devices, as well as the collection of the available digital data (Mylonas et al. 2013). Smartphone forensics is an emerging discipline, currently still in its infancy, but a growing field due to the rapid development of smartphone technology (Alghaffi et al. 2011). The focus of smartphone forensics is to collect the available data using the most appropriate acquisition method and examine the collected data in an efficient manner (Mohtasebi and Dehghantanha 2013). Recently, smartphone forensics have been receiving more attention because of the importance and impact of digital data during criminal and civil cases (Park et al. 2012).

This chapter focuses extensively on the current and most prevalent smartphone platforms, namely Google Android and Apple iOS, and the forensic examination of these platforms. Section 3.1 presents a brief overview of Google Android and Apple iOS operating systems, highlighting the architectural structure of these platforms. Key components (file system, partitions, and data acquisition) in the forensic examination of Android and

iOS smartphones are discussed in Section 3.2. Section 3.3 describes the challenges currently impeding the forensic examination of smartphones while Section 3.4 concludes the chapter.

3.1 Smartphone Operating Systems

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 end-user and the underlying hardware resources. The building blocks of modern day smartphone operating systems are platforms such as Symbian, Windows Mobile and BlackBerry OS. The success of these platforms is the result of interactive user interfaces, improved multimedia features and advance device-to-device communication structures. Although these are older mobile operating systems and no longer involve active development, they still established an acceptable standard followed by current smartphone operating systems. High-performance smartphone operating systems, namely Google Android and Apple iOS, are the current pacesetters, as reflected by their market share (69.87% and 28.82% respectively) in the 2nd quarter of 2017 (*NetMarketShare* 2018). The prevalence and domination of these platforms directed the focus only on Google Android and Apple iOS, as well as influence the remainder of this thesis to only concentrate on these platforms.

3.1.1 Google Android

Google Android is an open source operating system developed by the OHA for modern smartphones providing advanced capabilities (Lessard and Kessler 2010, Albano, Castiglione, Cattaneo, De Maio and De Santis 2011, Yang and Lai 2012). Officially announced in November 2007 (Hoog 2011), the popularity of platform caused the market share to grow exponentially over the past decade. Currently, Google Android is the dominant operating system in the smartphone market and used by various smartphone models, including Samsung, Sony, HTC, Huawei and Nokia. The Android platform is built using a combination of C/C++/Java programming languages (Padhya et al. 2016) and consists of the following major architectural components captured in Figure 3.1:

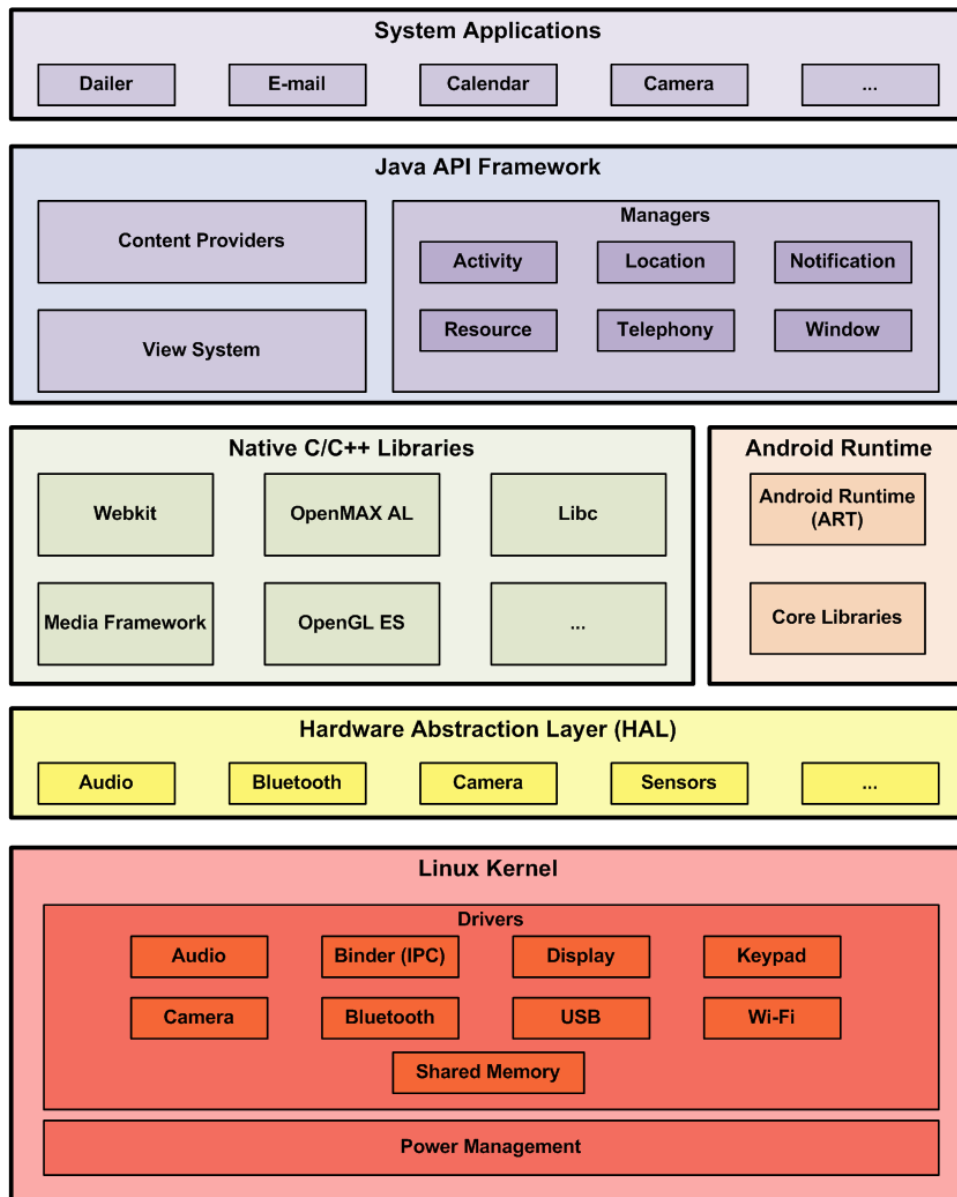


Figure 3.1: Architectural layers of the Android platform (*Android Developers 2017*)

system applications, Java API framework, native C/C++ libraries, Android runtime, hardware abstraction layer (HAL) and Linux kernel (*Android Developers 2017*).

The Android platform includes a collection of system applications, such as e-mail, text messenger, camera and calendar, developed using the Java programming language (*Android Developers 2017*). These applications are pre-installed by device manufacturers

(Chin et al. 2011) and reside in a location (`/system/app`) separate from user installed third-party applications (Zheng et al. 2014). Upon installation, the Android platform assigns each system application a distinct user and group identifier (ID) that remains constant during the application's lifetime (Albano, Castiglione, Cattaneo, De Maio and De Santis 2011). The purpose of system applications is to provide essential functionality to end-users and offers the necessary support for third-party applications (*Android Developers* 2017, Danielsson 2017).

Development of system and third-party applications rely on the Java API framework to present a set of programming interfaces that expose the Android OS to developers (Hasan and Haque 2016, *Android Developers* 2017). The framework simplifies the re-use of the core, modular components responsible for managing activities, creating services, implementing data structures and ultimately leads to the construction of powerful applications (Danielsson 2017). Components exposed by the Java API framework are (Vaibhav Kumar 2013, *Android Developers* 2017):

- View System: allow developers to build visually attractive, as well as complex user interfaces for applications using lists, grids, tables and buttons.
- Content Providers: enable applications to manage their data, access data stored by other applications and share data between various applications.
- Managers: offer support structures for applications, such as non-code resources, custom alerts, application life-cycle management, as well as location and telephony services.

Many of Android's lower level architectural components, such as Android Runtime (ART) and HAL, are built using native code that requires access to native libraries written in C and C++. These native libraries include but are not limited to, the Webkit (open source web browser), OpenGL ES (2D and 3D graphics), Media Framework (media playback) and Libc (standard C library). Developed applications that use C or C++ libraries can access these libraries via the Android Native Development Kit (NDK) (*Android Developers* 2017).

Before the release of Android version 5.0, the platform established a stable and secure execution environment for applications via the Dalvik Virtual Machine (DVM) (Barm-patsalou et al. 2013). DVM allows each application to execute in a virtual machine or sandboxed environment and translates Java code to a language that the Android OS can interpret (Simão et al. 2011). Several limitations of DVM led to the development of the new Android Runtime (ART) environment. Officially shipped with Android version 5.0, ART replaced DVM to ensure improved performance, optimised garbage collection and better debugging support. Similar to DVM, each application still runs in a virtual machine and supports a separate instance of ART. ART is specifically designed to run multiple virtual machines and manage the compilation of DEX files, a bytecode format uniquely associated with the Android platform. Compilation of DEX files occurs using ahead-of-time technology, which allows ART to perform the compilation of an application during installation (*Android Developers* 2017). ART significantly improves the performance of Android applications and the usability of Android smartphones.

The HAL further enhances the usability and overall functionality of the Android platform. Also, HAL is responsible for exposing device hardware capabilities to the higher-level Java API framework via standard interfaces. These interfaces allow developers from different hardware vendors to implement functionality without affecting or changing the higher-level architectural components. The HAL packages the functionality into multiple shared library modules, each of which adds support for a certain hardware component. Generally, the HAL supports the following hardware components (audio, Bluetooth and camera), as well as hardware sensors.

The final architectural component of the Android platform is the Linux kernel, which acts as an abstraction layer between the underlying physical hardware and the higher-level components. Android relies on Linux as the base operating system (Abalenkovs et al. 2012) and forms the foundation for functionalities such as threading, sandboxing, and low-level memory management (Albano, Castiglione, Cattaneo, De Maio and De Santis 2011, *Android Developers* 2017). Drivers and power management facilities further extend the kernel layer (Vaibhav Kumar 2013). Available drivers included are audio, binder (IPC), display, keypad, Bluetooth, camera, USB and Wi-Fi.

This presented architectural design of the Android platform creates an efficient and

effective operating system for smartphones. Evolving alongside Android is Apple iOS, a modern operating system developed specifically for Apple Inc. mobile devices.

3.1.2 Apple iOS

The first generation iPhone, released in June 2008 (Morrissey 2010), proved to be a fascinating piece of engineering that changed the mobile device landscape (Kala and Thilagaraj 2013). Along with the release of the iPhone came a new operating system called iOS, formerly called iPhone OS (Morrissey 2010). Developed and distributed by Apple Inc. (Kanoi and Ingole 2013), iOS is a proprietary OS and slimmed down version of the greater macOS (Tracy 2012). The iOS platform is constructed using a combination of C/C++/Swift/Objective-C programming languages (Padhya et al. 2016) and currently is the primary OS for the iPhone, iPad, iPad Mini and iPod Touch devices (Epifani and Stirparo 2016). Similar to the Android platform, iOS also 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 (Kanoi and Ingole 2013).

The iOS platform consists of the following major architectural components: applications, Cocoa Touch, Media, Core Services and Core OS/kernel (Ahmad et al. 2013, Kanoi and Ingole 2013). Figure 3.2 visualises the order of the iOS architectural components. The lower layers of the iOS platform contain all the fundamental services and technologies required by iOS applications. The higher-level layers include more sophisticated functions and provide object-oriented abstractions for the lower-level constructs (Kanoi and Ingole 2013).

The iOS platform usually ships with a collection of system applications that provide standard system services to the end-user. System applications pre-installed on an iPhone include but are not limited to the Phone, Mail and Safari applications (*Apple Developer* 2017b). Also, third-party applications approved by Apple can also be installed and executed on the iOS platform (Egele et al. 2011). iOS applications do not directly communicate with the underlying hardware, but instead, communicate through a set of well-defined system interfaces that protect applications from hardware changes. This abstraction makes it easy to develop and design applications that work consistently on

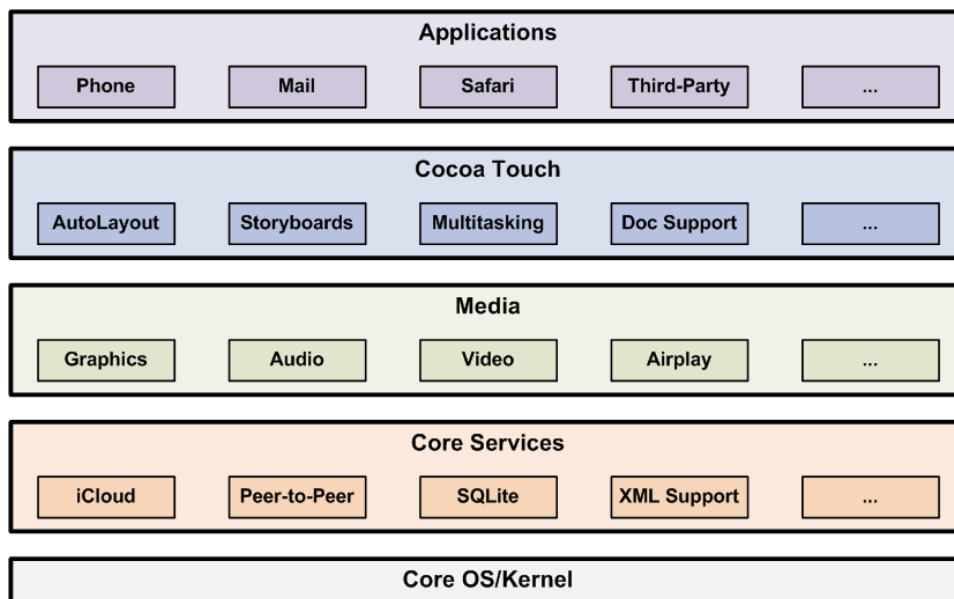


Figure 3.2: Architectural layers of the iOS platform

devices with different hardware capabilities (Kanoi and Ingole 2013).

The Cocoa Touch layer contains the necessary building blocks required to develop high quality and visually attractive iOS applications. This layer defines the necessary application infrastructure, supports user interface (UI) design, and offers necessary access to various fundamental technologies and high-level system services. High-level features available in the Cocoa Touch layer allow developers to define the visual construction of iOS applications (Auto Layout, UIKit Dynamics and Storyboards) and manage iOS applications' transition from and to the background state (Multitasking and UI State Preservation). Furthermore, the features also permit the sharing of documents and multimedia content (Document Picker and AirDrop) while supporting touch-based input (Gesture Recognizers) (*Apple Developer 2017b*).

Below the Cocoa Touch layer follows the Media layer, which contains the necessary graphics, audio, and video technologies to implement multimedia features in iOS applications. Technologies, such as UIKit, Core Graphics (Quartz), OpenGL ES, GLKit and Metal, provides high-quality graphics and custom art. Audio technologies include the Media Player Framework, AV Foundation, OpenAL and Core Audio, which allow developers of iOS applications to present end-users with a rich audio experience. AVKit

and AV Foundation technologies support the incorporation of video-based content in iOS applications. Finally, Airplay allows iOS applications to stream audio and audio using a wireless connection to the Apple TV, as well as other third-party AirPlay components (*Apple Developer 2017b*).

The third layer of the iOS architecture, called the Core Services layer, contains the fundamental system services for iOS applications. Fundamental among these system services are the iCloud storage, Peer-to-Peer services, Data Protection, SQLite and eXtensible Markup Language (XML) support. iCloud storage allows iOS applications to write and access data from a central location while Peer-to-Peer services support connectivity via Bluetooth to other nearby devices. SQLite and XML present iOS applications with structured storage for persistent data and encrypt the data using the Data Protection service (*Apple Developer 2017b*).

The final layer of the iOS architecture is the Core OS/kernel layer, which provides a range of services. These services include low-level network access to external accessories and vital OS services, such as memory management, file system support and thread handling. Interaction with the underlying hardware components also occurs via the Core OS/kernel layer (*Apple Developer 2017b*).

Similar to the architectural design of the Android platform, the iOS platform also provides end-users with another powerful smartphone operating system. The detailed overview of both Google Android and Apple iOS architectures confirms the complex structures of these platforms. Retrieving and analysing digital data from smartphones running these operating systems can be difficult and time-consuming. Therefore, the field of smartphone forensics attempts to overcome such challenges streamlining the forensic examination of smartphones.

3.2 Forensic Examination of Smartphones

Smartphone forensics forms part of mobile device forensics but focuses exclusively on collecting and analysing digital data retrieved from smartphones. To assist digital forensic professionals, the field of smartphone forensics attempts to standardise the examination of smartphones. Due to the complex architectural structures of smartphone operating

systems, smartphone forensics tends to be platform specific. The remainder of this section further explores smartphone forensics by focusing on each smartphone platform, namely Android and iOS, individually.

3.2.1 Android Forensics

Android forensics is a sub-discipline of smartphone forensics with a specific focus on the extraction, recovering and analysis of data present on Android smartphones (Hoog 2011). Having sound knowledge of Android's internal file structure, available partitions, rooting process and most appropriate acquisition technique will assist digital forensic professionals during collection and analysis of potential digital evidence (Tamma and Tindall 2015).

File System

The first generation of Android devices included a relatively unknown file system called the Yet Another Flash File System (YAFFS) (Hoog 2011). YAFFS was developed in 2002 (Vidas et al. 2011) as an open-source file system specifically for Not-AND (NAND) flash memory and is licensed under both the GNU Public License and a commercial license (Hoog 2011). YAFFS version 2 (YAFFS2) was released in 2004 (Vidas et al. 2011) to accommodate the availability of larger sized NAND devices (Vidas et al. 2011, Hoog 2011). YAFFS2 followed more strict NAND flash guidelines that enhanced the endurance of the NAND flash while also being optimised to run on low memory mobile or embedded devices. Furthermore, YAFFS2 also introduced essential features such as a log structure, built-in wear-levelling and error correcting code algorithms. YAFFS2 is fast, holds a small footprint in RAM and provides the capability to handle bad blocks (Hoog 2011). These features made YAFFS2 the ideal file system for early Android devices.

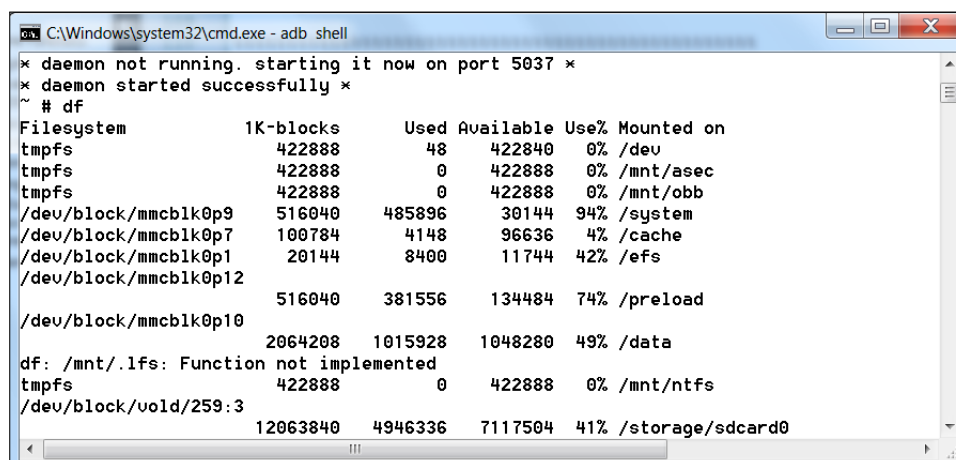
YAFFS2 was the primary file system for Android devices until Android version 2.2 (Froyo) (Vidas et al. 2011, Zimmermann et al. 2012, Barmptsalou et al. 2013). The single-threaded design of YAFFS2 caused bottlenecks in devices released with dual-core or multi-core chipset systems (Tamma and Tindall 2015, Kim and Kim 2012). With the release of Android version 2.3 (Gingerbread), Android switched from YAFFS2 to the

extended file system (EXT) version 4 (Vidas et al. 2011, Zimmermann et al. 2012). The EXT file system was first introduced in 1992 (Tamma and Tindall 2015) as the de facto file system for the Linux operating system (Hoog 2011, Park et al. 2016). Since the first release of EXT, the file system has evolved tremendously (EXT2, EXT3 and EXT4) with EXT gaining significance with mobile devices that include dual-core processors. The EXT4 file system runs smoothly on dual-core or multi-core mobile devices while providing stable performance (Hoog 2011, Kim and Kim 2012). Furthermore, the EXT4 file system also divides the disk space into logical blocks or partitions, which reduces overhead and improves throughput (Kim and Kim 2012). These key features of EXT4 make the file system ideal for newly released Android devices.

Partitions

Partitioning of the file system allows the logical storage blocks, or partitions, to be accessed independently of each other and organises the stored data in a structured manner. Due to the various manufacturers of Android devices, partitioning can differ across different device models. The most common partitions of Android devices are the following (Tamma and Tindall 2015, Vidas et al. 2011):

- `/boot`: holds all the information and files required by the boot process of the Android device.
- `/recovery`: allows the Android device to boot into the recovery console and perform updates or maintenance operations.
- `/data`: contains the bulk of user data (settings and customisations), as well as installed applications and their related data.
- `/system`: holds the Android operating system, which includes libraries, system binaries and pre-installed system applications.
- `/cache`: stores recovery logs, downloaded update packages and frequently accessed application data.
- `/sdcard`: is found across all Android devices regardless of make or model and allows end-users to store additional files and data.



```
C:\Windows\system32\cmd.exe - adb shell
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
~ # df
Filesystem          1K-blocks    Used Available Use% Mounted on
tmpfs                422888         48   422840    0% /dev
tmpfs                422888         0   422888    0% /mnt/asec
tmpfs                422888         0   422888    0% /mnt/obb
/dev/block/mmcblk0p9 516040   485896   30144   94% /system
/dev/block/mmcblk0p7 100784    4148   96636    4% /cache
/dev/block/mmcblk0p1 20144     8400  11744   42% /efs
/dev/block/mmcblk0p12
                    516040   381556  134484   74% /preload
/dev/block/mmcblk0p10
                    2064208 1015928 1048280   49% /data
df: /mnt/.1fs: Function not implemented
tmpfs                422888         0   422888    0% /mnt/ntfs
/dev/block/vold/259:3
                    12063840 4946336 7117504  41% /storage/sdcard0
```

Figure 3.3: Available partitions on an Android smartphone

It is necessary to connect the Android smartphone to a workstation and enable USB debugging to view the available partitions. Once connected, digital forensic professionals can use one of the following commands to list the partitions: `cat /proc/mounts`, `cat /proc/partitions` or `df`. Figure 3.3 shows the partitions of a Samsung Galaxy S2 Android smartphone running OS version 4.1.2 (Jelly Bean).

Although all partitions can hold data that may be of value during the examination, digital forensic professionals usually focus on the `/data`, `/system` and `/sdcard` partitions. Access to the `/data` and `/system` partitions is not permitted by default and requires root access.

Rooting

At the core of the Android platform is the Linux kernel 3.1.1, which is a monolithic Unix-like kernel. Unix-like systems offer support for multiple users via different accounts, such as regular user and superuser accounts (Tamma and Tindall 2015). The superuser account, commonly called the root account, is used for system administration and has the necessary privileges to control existing files and programs, as well as the OS (Drake et al. 2014, Tamma and Tindall 2015). The process of obtaining superuser privileges and accessing the root account on an Android device is called rooting (Drake et al. 2014).

Rooting may be seen as an adverse action, but it merely gives the end-user access to

the root directory (/) and permits the execution of root actions (Lessard and Kessler 2010). Rooting an Android smartphone enables the end-user to execute higher privileged functions, which are generally unavailable to the end-user (Grover 2013). The purpose of rooting is to overcome the restrictions imposed by mobile manufacturers (Tamma and Tindall 2015) and as a result, gain complete control of the Android smartphone. The restrictions removed by rooting allows the end-user to modify the look-and-feel with custom themes, alter or replace system applications and settings, perform full backups and restores, as well as install specialised applications (such as ad-blocker, over-clocking or tethering applications) that require administrator-level permissions (Drake et al. 2014, Tamma and Tindall 2015). Besides these advantages, several implications must be taken into consideration when rooting an Android smartphone. Firstly, rooting compromises the security of the device, simplifying the installation and propagation of malicious applications, as well as exposing the stored user data. Secondly, unsuccessful rooting attempts may “brick” the device, making the device inoperable. Finally, rooting the Android smartphone voids the warranty provided by the manufacturer (Tamma and Tindall 2015). It is, therefore, essential to understand the risks associated with rooting before commencing with the process.

The process of rooting depends on the current state of the bootloader of an Android smartphone. An unlocked bootloader allows the rooting process to copy the superuser (su) binary to the /system/xbin/su location in the current process' path and to grant the binary the necessary executable permissions using the chmod command (Tamma and Tindall 2015). Accompanied along with the su binary is an Android application, such as Superuser, which provides the end-user with a graphical interface to manage, request and grant root access. A locked bootloader requires the rooting process to find and exploit a vulnerability to unlock the bootloader before transferring the su binary (Tamma and Tindall 2015). Table 3.1 captures the most popular vulnerabilities and related exploits that are used to gain root access on an Android smartphone with a locked bootloader (Drake et al. 2014, Tamma and Tindall 2015, *AndroidVulnerabilities.org* 2018).

Rooting Android devices have become a desired activity and, therefore, digital forensic professionals can encounter such smartphones during examinations (Tamma and Tin-

Table 3.1: Common exploits to gain root access on Android smartphones

Exploit Name	Vulnerability	Description
RageAgainstTheCage	CVE-2010-EASY	The exploit targets the addb service.
psneuter	CVE-2011-1149	The exploit restricts permissions to the system properties space.
GingerBreak	CVE-2011-1823	Overwrites an entry in Global Offset Table to execute GingerBreak with root privileges.
Levigator	CVE-2011-1350 CVE-2011-1352	Exploit corrupts the kernel memory, which leads to privilege escalation.
zergRush	CVE-2011-3874	The exploit uses the Volume Manager daemon to trigger the vulnerability and gain root privileges.
mempodroid	CVE-2012-0056	The exploit writes directly to the code segment of the run-as program.
Qualcomm acdb audio buffer overflow	CVE-2013-2597	An application with access to the /dev/msm_acdb device file can use this flaw to escalate privileges.
WeakSauce	CVE-2014-3847	An exploit for HTC One m7 and m7 on Verizon.
Keystore buffer	CVE-2014-3100	Stack-based overflow in the encode_key function in /system/bin/keystore service in Android 4.3
Stagefright	CVE-2015-1538 CVE-2015-1539 CVE-2015-3824	A vulnerability exploited by sending a single multimedia message to an unpatched Android smartphone.
One class to rule them all	CVE-2015-3837 CVE-2015-3825	This vulnerability allows for arbitrary code execution via a specially crafted intent and affects Android versions 4.3-6.0

dall 2015). Investigations involving rooted Android devices allow digital forensic professionals to gain access to all of the digital data stored in the root directory. Access to the root directory, however, requires the USB debugging functionality to be enabled on the Android device (Lessard and Kessler 2010). Should digital forensic professionals encounter a locked Android device with USB debugging disabled, collecting the digital data from the root directory will not be feasible (Lessard and Kessler 2010). It is more likely that digital forensic professionals will encounter non-rooted Android smartphones during examinations and may require root access to collect the necessary data as digital evidence. Although rooting remains an option for digital forensic professionals to extract the necessary data from an Android smartphone (Grover 2013), the practice is not deemed ideal for several reasons. Rooting an Android smartphone often relies on a software vulnerability flaw that is usually specific to a particular device model or operating system version (Vidas et al. 2011). Due to such requirements, rooting may not always be feasible and, therefore, digital forensic professionals may need to proceed without accessing the root directory. Rooting the smartphone can also alter the partition storing user data (`/data/data` partition) (Vidas et al. 2011, Grover 2013). Such changes to the data can impact the validity of the digital data collected as evidence. Finally, rooting undermines the security model of the Android smartphone (Vidas et al. 2011), which makes the stored data more vulnerable to change.

The necessity to root an Android smartphone during examination directly depends on the need to retrieve specific data. The decision lies with the digital forensic professionals to weight the gains and risks before deciding to root the device. The decision regarding the rooting of an Android smartphone will also impact the acquisition of the stored data.

Data Acquisition

The most common acquisition techniques to retrieve data from Android smartphones are physical or logical acquisition. A rooted Android smartphone offers superuser privileges and provides full access to the Android Debug Bridge (ADB) shell (Afonin and Katalov 2016). The ADB shell permits a digital forensic professional to remotely connect to the Android smartphone and perform a bit-level copy (Albano, Castiglione, Cattaneo and De Santis 2011). Since Android is a Linux-based operating system, the platform includes

the built-in command line tool called `dd` (Albano, Castiglione, Cattaneo and De Santis 2011, Afonin and Katalov 2016). The `dd` command can perform a bit-level copy from a source to a specific destination, block-by-block, regardless of file system or operating system type (Albano, Castiglione, Cattaneo and De Santis 2011, Afonin and Katalov 2016). Digital forensic professionals can use the `dd` command to obtain a physical image of the internal memory of an Android smartphone. The example below illustrates the `dd` command:

```
dd if=/dev/block/mmcblk0p21 of=/sdcard/blk0.img bs=4096
    conv=notrunc,noerror, sync
```

This particular command creates a physical image using block sizes of 4096 bytes of the `mmcblk0p10 (/data)` partition as present on the Samsung Galaxy S2 smartphone. The command stores the image in the `/sdcard` location and the option, `noerror`, specifies the imaging of the partition must continue even if errors occur.

Rooted Android smartphones with pre-installed custom recoveries, such as CMW or TWRP, allow digital forensic professionals to create so-called Nandroid backups. Nandroid backups include a full image of the smartphones' file system and related data. This backup is created by booting into recovery mode and launching the backup command (Afonin and Katalov 2016).

As physical acquisition is reliant on a rooted Android smartphone, it may not be the most feasible technique to obtain data from the device. Digital forensic professionals encountering non-rooted Android smartphones must revert to logical acquisition techniques. Although Android does not include iTunes-like tools to make full device backups, particular manufacturers, such as Sony's PC Companion for Xperia devices, do provide software applications that support the creation of offline backups (Afonin and Katalov 2016). For smartphone manufacturers that do not support off-line backups, digital forensic professionals can use the built-in ADB functionality to obtain a logical image of an Android smartphone (Quick and Alzaabi 2011, Afonin and Katalov 2016). The `adb backup` command can create an off-line backup of a limited collection of data stored on the Android smartphone (Afonin and Katalov 2016). The backup process is limited explicitly due to Android fragmentation, which causes various device manufacturers of

Android smartphones to control what data forms part of the backup (Afonin and Katalov 2016). Should the backup fail to acquire the necessary data, digital forensic professionals can opt to use the `adb pull` command. The `adb pull` command can copy the specified file from the Android smartphone to the connected computer, as shown below (Quick and Alzaabi 2011):

```
adb pull /sdcard/<file>
C:\<local_folder>
```

Successful use of the `adb pull` command may be impeded by the current privileges of the Android smartphone as access to specific files may require superuser privileges. Digital forensic professionals must decide which acquisition technique will fit their needs best based on their initial assessment of the Android smartphone.

3.2.2 iOS Forensics

iOS forensics is another sub-discipline of smartphone forensics, which specifically emerged to assist digital forensic professionals with the extraction and analysis of data present on iOS smartphones. Having sound knowledge of iOS's internal file structure, available partitions, jailbreaking process and most appropriate acquisition technique will assist digital forensic professionals during collection and analysis of digital data. As mentioned in Section 3.1.2, iOS is primary OS for various Apple products, but the focus here strictly falls on the examination of iOS smartphones such as iPhones.

File System

Apple Inc. developed the Hierarchical File System (HFS) as the primary file system for Macintosh computers. HFS utilises a catalogue or B*tree structure to organise files and order data within the file system (Morrissey 2010). Several limitations of HFS, relating specifically to performance and reliability, led to the development of an improved version of HFS, called Hierarchical File System Plus (HFS+) (Epifani and Stirparo 2016). Enhancements of HFS+ included better disk space efficiency, international friendly file names using Unicode and support for other operating systems (Hoog and Strzempka

2011, Epifani and Stirparo 2016). Key files contained in HFS+ volume are (D’Orazio 2013)

- Catalogue file: organised as a B-tree and describes the volume’s folder and file hierarchy.
- Extents overflow: maintains a list of fork extents (up to eight).
- Allocation file: specifies if an allocation block is free or in use.
- Attributes file: contains additional metadata regarding files and folders.
- Startup file: facilitates the boot of non-macOS devices from HFS+ volumes.

A variation of HFS+, called HFSX, was selected as the primary file system for Apple’s mobile devices. The only difference between HFS+ and HFSX is case sensitivity (Morrissey 2010, Iqbal et al. 2012). HFS+, as well as the HFSX variation, proved to be valuable file systems for Apple computers and mobile devices for the past 30 years (Tamura and Giampaolo 2016). The single-threaded design and rigid data structures of these file systems, however, struggle to keep pace with ever-improving technology. In 2016, Apple Inc. announced a new file system, called the Apple File System (APFS), for all Apple’s mobile operating systems (watchOS, macOS, tvOS and iOS). The design of APFS is specific to the Apple ecosystem and related products while attempting to take advantage of flash solid-state drive storage structures. Enhancements introduced by APFS includes improved file system fundamentals, space sharing, HFS compatibility and better security capabilities such as encryption (Tamura and Giampaolo 2016). Apple Inc. officially shipped the APFS with iOS version 10.3.

Partitions

Similar to Android smartphones, smartphones running iOS also divides the logical storage space into partitions. Traditionally, iOS smartphones consist of two partitions: system and data. The system or root partition (/) is approximately 500MB in size, house the operating system and all preloaded applications. By default, the system partition

is read-only and not accessible to the end-user to modify. The system partition is expected to remain in factory state and is only modified when an update of the operating system occurs (Iqbal et al. 2012, Epifani and Stirparo 2016). Therefore, the system partition usually does not contain data of much forensic value but could still be examined if deemed necessary (Morrissey 2010).

The data partition (`/var`) of an iOS smartphone is, however, mounted with read/write access and occupies most of the available space (Iqbal et al. 2012, Epifani and Stirparo 2016). The data partition holds the bulk of data, which includes all forms of user data, as well as user-installed applications (Morrissey 2010, Epifani and Stirparo 2016). User data of value is generally found in the following locations (Iqbal et al. 2012):

- `/private/var/mobile/Applications`: contains all user-installed applications and their related data.
- `/private/var/logs`: holds crash and reboot logs.
- `/private/var/mobile/Media`: stores user photos, videos, books, music files and other downloads.
- `/private/var/mobile/Library`: holds iOS settings and system applications.
- `/private/var/mobile/Documents`: stores user files and documents.
- `/private/var/mobile/Downloads`: collects all downloaded media and files.

Accessing and obtaining the data available in both the data and system partitions is not permitted by default. End-users or digital forensic professionals wishing to access and view the data in these partitions must jailbreak the iOS smartphone.

Jailbreaking

The term “jailbreaking” originates from a Unix practice of placing services in a restricted set of directories called a “jail” (Zdziarski 2008) and breaking free from these restrictions. Jailbreaking is possible by exploiting a specific flaw in either the hardware or software components of the iOS platform, which unlocks the smartphone (Epifani and Stirparo

2016, Egele et al. 2011). Once unlocked, the jailbreak removes certain restrictions and limitations put in place by Apple (Tamma and Tindall 2015, *Managing Jailbreak Threats on iOS* 2016), allowing privileges to be elevated to obtain system-level or root access (Egele et al. 2011). This form of privilege escalation breaks almost all protection iOS offers (Miller 2011) and modifies the system kernels to permit read and write access to the file system (*Managing Jailbreak Threats on iOS* 2016).

Jailbreaking iOS smartphones is also a popular activity and motivated by several possible outcomes. Besides removing known restrictions, jailbreaking also modifies the system kernel to accept non-signed binaries (Egele et al. 2011, D’Orazio 2013). The removal of the restrictions allows for the installation and execution of applications on the Apple smartphone generally not found in the Apple App Store (Bader and Baggili 2010, Morrissey 2010, Miller 2011). Furthermore, jailbreaking also permits the customisation of the smartphone and support carrier-unlocking, allowing end-users to use the Apple smartphone on different carriers. Regardless of these advantages, jailbreaking is known to impair functionality and reduce the performance of the smartphone (Morrissey 2010, *Managing Jailbreak Threats on iOS* 2016). The removal of Apple’s protection measures also introduces significant security risks since access to confidential, as well as decrypted information is provided (D’Orazio 2013). Simultaneously, jailbreaking also opens the smartphone to the installation of malware (Miller 2011, *Managing Jailbreak Threats on iOS* 2016). Jailbreaking is an invasive activity that modifies the system partition (Gomez-Miralles and Arnedo-Moreno 2011, Epifani and Stirparo 2016) but leaves the data partition unaltered (Gomez-Miralles and Arnedo-Moreno 2011). The limited impact of the jailbreak process on the data partition makes jailbreaking a possible solution to obtain access to the required data stored on iOS smartphones.

All known iOS jailbreaks exploit a specific vulnerability, and the quality of the jailbreak depends on the exploitation of the vulnerability to break the enforced restrictions (Miller 2011). To simplify the process of jailbreaking, researchers usually automate the process by developing a software tool (*Managing Jailbreak Threats on iOS* 2016). Each jailbreak tool is specific to a specific range of iOS versions and device models (*Managing Jailbreak Threats on iOS* 2016) since Apple Inc. usually patches the exploited vulnerabilities in subsequent releases of new iOS versions (Miller 2011). It becomes, therefore, a

prevalent practice to continuously update or create new jailbreak tools with the release of new iOS versions or device models.

The first generation of jailbreak tools was called “tethered jailbreaks”, meaning the jailbreak status of the iOS smartphone is only maintained as long as the operating system is running and will disappear after a reboot (Gomez-Miralles and Arnedo-Moreno 2011, Miller 2011, *Managing Jailbreak Threats on iOS* 2016). The end-user must then perform a re-jailbreak after the reboot by connecting to a computer and visiting either a certain website or executing a specific application (Miller 2011). Tethered jailbreaks have a lesser impact on iOS smartphones and leave minimal traces on the device (Belenko and Sklyarov 2011). Modern jailbreak tools are “untethered” and allow iOS smartphones to independently reboot into a jailbroken state (*Managing Jailbreak Threats on iOS* 2016). Untethered jailbreaks remain after a reboot by capitalising on a persistent vulnerability, which is harder to accomplish and, therefore, are not found regularly (Miller 2011). The reliance on a persistent vulnerability causes untethered jailbreaks to have a more significant impact and leave permanent traces on iOS smartphones (Belenko and Sklyarov 2011). Table 3.2 lists the untethered jailbreaks that exist for iOS since the release of version 4.0.

Similar to the root status of Android smartphones, the jailbreak status of iOS smartphones will also impact the acquisition of digital data from these devices. It is crucial for digital forensic professionals to first confirm the jailbreak status of iOS smartphones before continuing with the acquisition of the available data.

Data Acquisition

Similar to Android smartphones, the most common techniques to retrieve data from iOS smartphones also include physical and logical acquisition. Physical acquisition is fully available for 32-bit iOS smartphones while only limited support exists for 64-bit devices (Afonin and Katalov 2016). The requirements to perform physical acquisition is an iOS smartphone with a jailbroken state and unlocked passcode (Afonin and Katalov 2016). With a functioning jailbreak state, the necessary tools to capture a physical image, such as Secure Shell (SSH) and Terminal, are available to digital forensic professionals. SSH is used to connect to the iOS smartphone and the Terminal tool to launch the (dd)

Table 3.2: Availability of untethered jailbreak tools across iOS versions

iOS version	Available Jailbreak Tools
4.0	ipwndfu, limera1n, sn0wbreeze, redsn0w PwnageTool, Star (JailbreakMe 2.0)
4.1	greenpois0n
4.2	Saffron (JailbreakMe 3.0), unthredek4il
4.3	unthredera1n
5.0	Absinthe
5.1	cinject
6.0	evasiOn
6.1	P0sixspwn
7.1	Pangu
8.0	Pangu8, PPJailbreak, TaiG
8.4	EtasonJB, Home Depot
9.0	Pangu9, Phoenix
10.0	extra_recipe + yaluX, yalu102
10.1	yalu + mach_portal

command line utility. The `(dd)` command captures an image of the data partition of the jailbroken iOS smartphone. With only limited support available for 64-bit iOS smartphones, physical acquisition of the available data may not always be feasible.

The limitations associated with physical acquisition of iOS smartphones cause digital forensic professionals to more often use logical acquisition. Logical acquisition of an iOS smartphone is generally achieved using the iTunes backup feature. The iTunes backup feature utilises Apple's synchronisation protocol to create a backup of the data stored on the iOS smartphone during the backup process. The backup file stores personal data (address book, calendar, images, text messages, e-mails, and web history), device configuration settings and application preferences. The created backup files are usually encoded but can be parsed and viewed using the following software applications: plist editor, SQLite database browser and iPhone backup extractor (Bader and Baggili 2010).

By default, iTunes stores the created backup files in a preconfigured directory (Bader and Baggili 2010):

- Macintosh: `/Library/Application/Support/MobileSync/Backup`
- Windows 7: `C:\Users\\AppData\Roaming\Apple Computer\MobileSync\Backup`

Digital forensic professionals must use their initial assessment of the iOS smartphone and the data required to decide which acquisition technique will suit their needs best. Traditionally, data from iOS smartphones are collected using logical acquisition (Barm-patsalou et al. 2013).

3.3 Challenges for Smartphone Forensics

The previous section discussed in depth Android and iOS forensics, both of which are sub-disciplines of smartphone forensics. The discussions showed that smartphone forensics is still a very new and immature branch of digital forensics, even more so than mobile device forensics. Therefore, the same challenges currently affecting mobile device forensics also impact the effectiveness of smartphone forensics. Furthermore, the continuously improved functionality and advanced features of smartphones create new and more complex challenges for digital forensic professionals. The current challenges influencing smartphone forensics are collected in the following categories: acquisition challenges, malicious applications and anti-forensics.

3.3.1 Acquisition Challenges

Similar to mobile device forensics, the diversity, design and improving technology of smartphones also continuously create challenges for smartphone forensics. Rapid changes in smartphone technology, as well as the heterogeneity of current smartphone operating systems and device models (Barm-patsalou et al. 2013) increase the complexity of establishing and maintaining standards in smartphone forensics (Alghaffi et al. 2011, Freiling et al. 2011). Furthermore, the non-existence of a widely accepted investigation process

model (Mohtasebi and Dehghantanha 2013) and the lack of a scientifically sound method for acquiring data (Alghaffi et al. 2011) impact the analysis and examination of recovered digital data. Frequent adaptations in smartphone security measures, such as device locks (pin, password, passcode and gesture) and data protection measures (encryption) can also influence the availability of data during examinations. Without the necessary standards and processes in place for smartphone forensics, proper and efficient acquisition of the available data on smartphones will remain challenging for digital forensic professionals.

3.3.2 Malicious Applications

Advances in smartphone technology and the popularity of smartphone applications among end-users create new opportunities for the dissemination of malware. Malware any hostile, intrusive or annoying software application, especially designed to manipulate a device without the end-user's consent (La Polla et al. 2013). With regards to smartphones, malware is usually delivered in the form of malicious applications. Malware, such as mobile botnets and ransomware, can impact the examination of smartphones and limit the availability of digital data. A mobile botnet is a collection of compromised mobile devices, controlled by a botmaster through a command and control network for a malicious purpose (Geng et al. 2012). Primarily, mobile botnets act as a delivery platform for the distribution of malicious applications. Should a smartphone form part of a mobile botnet and participate in malicious activities, a digital forensic professional may link such activities incorrectly to the smartphone end-user. While smartphones infected with mobile botnet malware may present digital forensic professionals with valuable data that can indicate attribution, ransomware actively attempts to prevent access to such data. Ransomware is malware that either encrypts the available data or locks the smartphone (Richardson and North 2017). Ransomware is becoming more prominent in the smartphone domain (Mercaldo et al. 2016) and can severely influence the availability of digital data. It becomes, therefore, necessary for digital forensic professionals to be aware, as well as detect the presence and impact of malware on smartphones before concluding an examination.

3.3.3 Anti-Forensics

Anti-forensics attempts to compromise the availability or usefulness of digital data during examinations (Harris 2006). Individuals can use a collection of tools, methods or processes to negatively impact the existence, quantity or quality of digital data, which can hinder subsequent analysis and examination of data (Kessler 2007). Application of anti-forensic techniques is not only intended for illegitimate purposes but can also be used by individuals who wish to protect their privacy (Kessler 2007). Although anti-forensics is currently quite a young and immature discipline, especially when contextualised in the smartphone environment (Distefano et al. 2010), it remains important for digital forensic professionals to be aware of anti-forensics and mitigate such actions when analysing digital data. The available methods and techniques to conduct anti-forensics fall into the following categories: (i) data destruction, (ii) data source elimination, (iii) data hiding, (iv) data counterfeiting and (v) attacks against forensic tools (Harris 2006, Conlan et al. 2016).

Data destruction attempts to destroy, dismantle or otherwise make digital data unusable during examinations (Harris 2006, Distefano et al. 2010). The methods available to perform data destruction include data deletion or artefact wiping, which is the act of overwriting data so that it is impossible to recover or restore the data (Garfinkel 2007, Karlsson and Glisson 2014). Overwritten data is irrevocably destroyed (Karlsson and Glisson 2014), and often not detected during the examination of smartphones. Data destruction is accomplished using file shredding tools, such as ProtectStar's iShredder Pro, to render existing files unrecoverable (Sporea et al. 2012). There are, however, two weaknesses associated with data destruction. Firstly, data destruction tools or operations may fail to destroy all data and allow digital forensic professionals to collect limited sources of digital data. Secondly, the presence and operation of data destruction tools can also leave notable traces behind. Regardless of presented weaknesses, data destruction is the preferred anti-forensic technique to eliminate digital data available on smartphones (Karlsson and Glisson 2014).

Data source elimination primarily prevents or neutralises the creation of data (Harris 2006, Distefano et al. 2010). Therefore, there is no need to apply data destruction operations since the data is never created. Data source elimination occurs by disabling

logs to prevent the recording of performed activities (Harris 2006). Essentially, the use of data source elimination techniques will minimise the footprint created by end-user activities (Garfinkel 2007). Similar to data destruction, the actual process of data source elimination also leaves notable traces behind (Harris 2006). Furthermore, the very lack of specific data sources can also provide valuable insight.

Data hiding is the act of removing data from view so that the data is less likely to be detected during examination of the smartphone. The data is, however, not destroyed or manipulated but instead just made less visible (Harris 2006, Distefano et al. 2010). Data hiding transpires by moving data into unallocated or otherwise unreachable storage locations. Other examples of data hiding techniques include cryptography and steganography (Garfinkel 2007, Kessler 2007). Cryptography, also known as hidden writing, is the ultimate anti-forensic tool (Kessler 2007) and refers to the use of encryption to conceal data (Pfleeger and Pfleeger 2007). While cryptography is very useful for hiding data, the encrypted data can still easily be detected (Garfinkel 2007). Steganography, which is the art of concealing data in clear sight (Pfleeger and Pfleeger 2007), can be used to embed the encrypted data in cover text or images to avoid detection (Garfinkel 2007). Steganography tools, such as StegDroid and MobiStego, allow end-users of Android smartphones to easily hide data within images (Sporea et al. 2012). Similar to data destruction, the sheer presence of data hiding tools installed on a smartphone may provide valuable insight.

Data fabrication, which is the act of creating fake or false data that appear to be something legitimate (Distefano et al. 2010). Data fabrication involves the selective manipulation of existing data or creating new data to negatively impact the validity of the existing data (Harris 2006). An example of a data fabrication application is Fake Location, which allows end-users of iOS smartphones to set fake locations for desired applications (Sporea et al. 2012). The presence of such manipulated data can incorrectly divert the examination of the smartphone (Distefano et al. 2010).

The final anti-forensic category involves attacks against forensic tools. These attacks often demonstrate the most devastating anti-forensic activities by attempting to impede either the acquisition of the digital data or the entire examination phase. Typical attacks include program packers that attempt to compress or encrypt digital data that

forensic tools attempt to acquire, anti-reverse engineering, which prevents the extraction of relevant data by the forensic tools, as well as attacks against the integrity of digital forensic professionals by means of smear campaigns or malicious activities (Conlan et al. 2016).

Smartphone end-users with malicious intent are becoming aware of the importance of digital data present on smartphones. To avoid detection and protect their privacy, such end-users may be interested in deploying the described anti-forensics techniques to alter data and thwart examinations. As smartphone anti-forensic techniques mature and grow in popularity, mitigating anti-forensic actions will become digital forensic professionals most significant challenge in the near future.

3.4 Summary

The growth and recent improvements in smartphone technology emphasised the need for smartphone forensics. As a discipline, smartphone forensics provide digital forensic professionals with the necessary concepts, processes and techniques to acquire and examine digital data from smartphones. The current popularity of Android and iOS mobile operating systems shift the current focus of smartphone forensics to these platforms. It is, therefore, necessary to have a good understanding of the file system and partitions of these platforms to know where the available data resides and how to acquire the data effectively. Acquisition of smartphone data is often impacted by various challenges, which can impede examinations by limiting the available data. It is important for digital forensic professionals to rise above such challenges since the acquired digital data from smartphones can play an essential role in the outcome of a criminal or civil case.

Chapter 4

Smartphone Data

The availability of forensic processes and techniques, as well as the improvement of related technologies, assist digital forensic professionals with the collection of digital data from smartphones. Data collected and stored on smartphones, called smartphone data, include but are not limited to pre-generated data, application-related data and externally created data transferred to smartphones by end-users. Improvements of smartphone forensics are of great importance since the ubiquitous use of smartphones during end-user's daily activities has rendered these devices rich sources of smartphone data. The adequate collection and analysis of such data are needed, especially when smartphones form part of criminal or civil cases.

This chapter further explores smartphone data, providing a formal description in Section 4.1 and identifying the various sources of smartphones data in Section 4.2. Section 4.3 presents the different storage structures present on smartphones to hold smartphone data. The value of analysed smartphone data is highlighted in Section 4.4. Finally, existing measures available to protect smartphone data and the limitations of those protection measures are discussed in Section 4.5 and 4.6 respectively. Section 4.7 concludes the chapter.

4.1 What is Smartphone Data?

The operation of smartphones and the execution of installed applications as end-users interact with the devices ensure smartphone data is generated continuously in a deterministic and undisturbed way (Mylonas et al. 2012). Smartphone data generally includes pre-generated data, data created due to the usage of the smartphone and data transferred to the device by the end-users. Pre-generated data is readily available when the smartphone is acquired and consists mostly of device information such as hardware and software identifiers. The operation of the smartphone and installed applications create large volumes of smartphone data, which can provide a clear snapshot of the end-user's actions within a specific period. The evaluation of an end-user's actions is further supported and substantiated by the analysis of data transferred to smartphones, which can include multimedia files and documents. Within the context of this thesis, smartphone data is defined as any form of digital data that resides directly on smartphones.

Pre-generated, created or transferred smartphone data primarily resides in three locations: (i) SIM card, (ii) internal storage and (iii) external or portable storage such as micro SD cards (Curran et al. 2010, Al-Hadadi and AlShidhani 2013). While all locations can contain smartphone data, all three storage locations may not always be present on smartphones. The availability of storage locations, as well as the presence of specific sources of smartphone data, depend on the smartphone manufacturer and the end-user's usage of the smartphone.

4.2 Sources of Smartphone Data

Various sources of smartphone data are present in different locations and include both static and dynamic forms of data. Regardless of location or nature of the smartphone data, the data can become valuable digital evidence should the smartphone be linked to criminal or civil cases. The primary sources of smartphone data are organised into the following categories: SIM card data, device-specific data, user-created data, sensor data and application-related data.

4.2.1 SIM Card Data

A SIM is an integrated circuit that functions as part of a universal integrated circuit card (UICC) physical smart card. Traditionally referred to as a SIM card, the identity module is a removable component that contains and securely stores essential information on extended non-volatile storage (Ayers et al. 2013). Information collected on SIM cards include unique identifies, subscriber-related information and authentication keys (Curran et al. 2010). The unique identifiers consist of the integrated circuit card identifier (ICCID) and the International Mobile Subscriber Identity (IMSI) numbers, which are used to uniquely identify the end-user of the smartphone on the mobile network (Mylonas et al. 2012). The repository of subscriber data available on SIM cards include phonebook entries or contacts, text messages, call information (last numbers dialed), location area identity and service-related information (Ayers et al. 2013). Finally, authentication keys, such as personal identification number (PIN) and personal unblocking key (PUK), protect the data stored on SIM cards (Brothers 2009, Ayers et al. 2013).

The diminished storage capacity of SIM cards limits the amount of data that resides on the smart cards. Improvements of smartphone technology led to an increase of storage capacity of smartphones, allowing the devices to hold larger volumes of data on internal or external storage locations. Although SIM card data will always form a critical component of digital evidence, it is crucial for digital forensic professionals to also collect data from the internal or external memory of smartphones.

4.2.2 Device-specific Data

Data accumulated in the internal storage volume of smartphones that relates specifically to smartphones is defined as device-specific data (Mylonas et al. 2012). The data is generally created during the smartphone manufacturing process, while the installation of required software components, such as the operating system, also imprints unique data relating to the smartphone. The primary form of device-specific data is the IMEI number, used to uniquely identify the smartphone (Aloul et al. 2009). The Global System for Mobile communications (GSM) network relies on the presence of the IMEI number to identify valid smartphones and prevent unauthorised smartphones, such as blacklisted

devices, from accessing the network. Secondary forms of device-specific data include various hardware (model number, serial number, manufacturer, brand and CPU model) and software (OS version, baseband version, kernel version and build number) identifiers (Mylonas et al. 2012, Schölzel et al. 2015, 2016). Finally, statistics gathered from system resources (CPU load, RAM usage, network traffic and battery level) also form part of device-specific data (Jeon et al. 2011). Device-specific data, such as hardware identifiers, remains unchanged but the application of software updates create changes to software identifiers.

Although device-specific data is often not included as digital evidence, the available data still offers value during examinations. The exclusive nature of device-specific data allows digital forensic professionals to uniquely identify smartphones that may form part of an extensive investigation. However, smartphone data reflecting the end-users' interaction with the device is usually of more value during criminal or civil cases.

4.2.3 User-created Data

User-created data strictly refers to data created externally to the smartphone that the end-user transfers to the device using either a wired (USB) or wireless (Bluetooth) connection. However, user-created data excludes any data created or generated due to the use of smartphone applications by the end-users. User-created data usually resides in internal or external (i.e. micro SD card) storage locations and can easily be added or removed by the end-users. User-created data will most likely comprise of pictures, photographs, videos, audio and document files (i.e. presentations or spreadsheets) (Dlamini et al. 2016, Quick and Choo 2016). Other forms of user-created data also include personally identifiable information (phone number and e-mail address), accounts (Google or Apple, social media and bank) and personalised configurations (smartphone settings and themes).

Data created by end-users is essential digital evidence since the data offers valuable information about the end-user. The dynamic nature of user-created data, however, influences the availability of the data. In contrast to SIM card and device-specific data, user-created data will differ between smartphones since each end-user's interaction with their smartphone is unique. It is, therefore, not possible for digital forensic profession-

als to pre-determine the presence and location of user-created data. Regardless of the challenges associated with user-created data, digital forensic professionals must review such data during the examination. End-users are, however, not alone responsible for the creation of smartphone data as data can also be generated by smartphone sensors.

4.2.4 Sensor Data

Current smartphone technology equips smartphones with various sensors that create and collect smartphone data (Mylonas et al. 2012, 2013). Hardware sensors present on most modern smartphones include multimedia, location, motion and environmental sensors (Mylonas et al. 2013). Multimedia sensors, such as the microphone and the camera (often a pair of lenses on the front and back of a smartphone) can be used to record audio and capture photographs or videos respectively. The location sensor refers to the GPS receiver available on smartphones that can calculate and track the current position of the device. Motion sensors, such as the accelerometer and gyroscope, observe movement and orientation. Finally, environmental sensors (magnetometer, proximity, light, temperature and pressure) capture data regarding the immediate environment. These sensors continuously observe and collect data that can offer context awareness (Mylonas et al. 2012, 2013).

Although smartphones sensors create and collect valuable sources of data, the volatile nature and time-sensitivity of the data influence the practical use of sensor data as digital evidence. Furthermore, sensor data cannot be collected during post-mortem investigations and not all collected sensor data are easily comprehensible (Mylonas et al. 2013). Therefore, it is also necessary to include data collected by software sensors such as user and system applications.

4.2.5 Application-related Data

The final source of smartphone data includes any data created or generated by smartphone applications. Smartphone applications, which consist of user (third-party) and system (pre-installed) applications, create both permanent and temporal data as the application executes (Mylonas et al. 2012). Application generated data is specific to an

application and the application generally collects the data in flat files or databases (Mylonas et al. 2012). Examples of application-generated data include but are not limited to contacts, text messages, instant messages, call history and websites visited. These applications can act as witnesses and by analysing the collected data will provide digital forensic professionals with better context regarding the use of smartphones by end-users.

The improvements of smartphone technology along with the popularity of applications among end-users cause application-generated data to form the most extensive collection of smartphone data. Both the value and volume of application-generated data emphasise the importance of such data to form part of criminal or civil cases.

4.3 Storage Structures for Smartphone Data

These various sources of smartphone data described in Section 4.2 play a critical role in the ecosystem of smartphones. Practical storage of such data is necessary to allow smartphones and the installed applications fluent and easy access to the data. The primary storage structures for smartphone data include plain text, XML and plist files, as well as SQLite databases.

4.3.1 Plain Text Files

Plain text files are a lightweight format and only store textual data. These files contain no formal structure, such as tags or tables, and are recognisable as files with either specific (`.txt` or `.log`) or no extension. Generally, plain text files are used by both Android and iOS mobile operating systems to store specific sources of smartphone data. Such sources can include smartphone configuration information and system logs, namely, reboot and error/crash log files (Abalenkovs et al. 2012). Although the structureless nature of plain text files can increase the complexity of retrieving and analysing the available data, digital forensic professionals must still evaluate these files for potential digital evidence.

4.3.2 Extensible Markup Language (XML)

XML was first specified, standardised and consolidated in 1998 as a standard language to describe raw data (Lee et al. 2010). XML, which is a markup meta-language (Kotze 2015), forms a subset of the Standard Generalised Markup Language and is specifically designed to support ease of implementation, readability, scalability, as well as interoperability (Bray et al. 2006). The structure and overall design of XML documents support these features. Usually, XML documents are composed of elements, declarations, comments, character references, processing instructions, as well as the raw data (Bray et al. 2006). Each XML document starts with an XML declaration and is followed by a root element, which contains the remainder of elements, instructions and references. This formal structure makes XML documents easily readable by both humans and electronic devices, allowing for accurate information classification and data exchange (Kotze 2015).

With regards to the Android platform, XML documents play a critical role in the development of Android applications. XML documents assist with the UI design, defining components used in the interface such as layouts, styles, strings, colours and dimensions. Furthermore, Android applications also rely on XML documents to contain required data, such as the `AndroidManifest.xml` file that captures configurations required by the application to function optimally. The iOS platform and related iOS applications, however, do not often use XML documents to collect and store data. Instead, the iOS platform relies on another storage structure similar to XML documents called Property Lists.

4.3.3 Property Lists

Property lists or plists are XML or binary formatted files (Iqbal et al. 2012) used by Apple Inc. to manage configuration settings and store needed information (Epifani and Stirparo 2016). Both pre-installed and third-party applications use plists to store relevant data in a structured, transportable and easily accessible manner (Iqbal et al. 2012). XML formatted plists are simple text files and can easily be viewed using standard text or plist editors, which are available for both Mac and Windows computers (Zdziarski 2008, Iqbal et al. 2012, Epifani and Stirparo 2016).

Table 4.1: Plist files containing valuable information

Filename	File Location
Last Apple App Store Search	<code>/var/mobile/Library/Preferences/ com.apple.AppStore.plist</code>
SIM Card Information	<code>/var/wireless/Library/Preferences/ com.apple.commcenter.plist</code>
Account Information	<code>/var/mobile/Library/Preferences/ com.apple.accounts.plist</code>
Wireless Networks Accessed	<code>/var/references/SystemConfiguration/ com.apple.wifi.plist</code>
Mailboxes	<code>/var/mobile/Library/Mail/ MailboxCollections.plist</code>

Plists organise the data into named values and lists using several objects types (Iqbal et al. 2012). The file starts with standard header information and contains a single root object, wrapped using the `<plist>` document type tag. The data components are generally stored using either dictionaries (`<dict>`) or arrays (`<array>`). Dictionaries use key-value pairs, where each data member is encoded by placing a dictionary key in the `<key>` tag and immediately following the key with the required value, which is placed in the appropriate tag depending on the value's type. Arrays only contain a list of values passed as arguments. All of these types produce meaningful structured data and allows the Cocoa and Core Foundation plist API to quickly convert these hierarchically structured types to and from standard XML (Iqbal et al. 2012).

The data partition of iOS smartphones contains various plists that hold valuable information (Iqbal et al. 2012). These plists can store information such as accounts, preferences, system configurations and network information, as shown in Table 4.1. The popularity of plist files among iOS applications and the easy to use structure of these files make them ideal storage structures for smartphone data.

Table 4.2: SQLite WAL Header Format (*SQLite 2017a*)

Offset	Size	Description
0	4	Magic Number (0x377f0682 or 0x377f0683)
4	4	File Format Version (currently 3007000)
8	4	Database page size
12	4	Checkpoint sequence number
16	4	Salt-1: random integer incremented with each checkpoint
20	4	Salt-2: a different random number for each checkpoint
24	4	Checksum-1: first part of the checksum on first 24 bytes of header
28	4	Checksum-2: second part of the checksum on first 24 bytes of header

4.3.4 SQLite Databases

SQLite is a favoured database system for mobile devices and traditional operating systems (Hoog 2011). The consistent use of SQLite databases is due to the public availability and efficient design of the database structure. The entire SQLite code base is open-source and contains significant functionality in a single cross-platform file of a few hundred kilobytes (Hoog 2011). The compact and high-quality design of SQLite databases makes them the ideal solution for storing persistent data on smartphones.

SQLite is best described as an efficient software library that implements a lightweight Structured Query Language (SQL) database engine (Freiling et al. 2011, Jeon et al. 2012, Cheema et al. 2014). The compact and granular design of SQLite does not require a separate server, allowing for quick processing of stored data by reading and writing directly to a disk file (*SQLite 2017b*). The main database file (.db, .db3 or .sqlitedb) contains a complete SQL structure that includes tables, indices, triggers and views (*SQLite 2017b*). The SQL structure divides the main database file into one or more pages, and each page shares the same size (*SQLite 2017a*). The first page of the main database file is a 100-byte database header page. The database header page holds the file structural information and schema tables, which describes the tables, indices, triggers and views contained in the main database file (Jeon et al. 2012). The remaining pages follow the B-tree structure, where each page contains a B-tree index and B-tree table that hold the

Table 4.3: SQLite databases found on Android and iOS platforms

Android	
<i>Application</i>	<i>File Location</i>
Address Book/Contacts	/data/data/com.android.providers.contacts/ databases/contacts2.db
Browser History	/data/data/com.android.browser/ databases/browser.db
Calendar	/data/data/com.android.providers.calendar/ databases/calendar.db
Call History	/data/data/com.sec.android.provider. logsprovider/databases/logs.db
Cookies	/data/data/com.android.browser/ databases/webview.db
Messages	/data/data/com.android.providers.telephony/ databases/mmssms.db
WhatsApp	/data/data/com.whatsapp/databases/ msgstore.db
iOS	
<i>Application</i>	<i>File Location</i>
Address Book/Contacts	/var/mobile/Library/AddressBook/ AddressBook.sqlitedb
Bookmarks	/var/mobile/Library/Safari/Bookmarks.db
Calendar	/var/mobile/Library/Calendar/ Calendar.sqlitedb
Messages	/var/mobile/Library/SMS/sms.db
Notifications	/var/mobile/Library/Calendar/ Notifications.db
WhatsApp	/net.whatsapp.WhatsApp/Documents/ ChatStorage.sqlite

actual data (Patodi 2012).

During transactions, SQLite stores additional information in a secondary file called either a rollback journal or WAL file (*SQLite* 2017a). The purpose of this secondary file is to ensure the integrity of the data in the event of transaction failure. The rollback journal was the default method of SQLite to implement an atomic commit and rollback (Caithness 2012). With the release of SQLite version 3.7.0, the new WAL approach was adopted, which allowed for improved speed and concurrent execution (Caithness 2012). The WAL approach 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 (see Table 4.2) and zero or more WAL frames. When a checkpoint occurs, SQLite writes the updated or new pages in the WAL file to the main database file. Once completed, the WAL file remains untouched and can be reused rather than deleted (Caithness 2012). Traditionally, SQLite performs an automatic checkpoint when the WAL file reaches a size of 1000 pages (approximately 4MB in file size) (*SQLite* 2017c).

Table 4.3 captures popular applications of both the Android and iOS platforms that use SQLite databases to store persistent data. Currently, SQLite is the preferred database system for various Android and iOS applications (Bader and Baggili 2010, Vidas et al. 2011, Ayers et al. 2013, Kala and Thilagaraj 2013, Cheema et al. 2014, Choi and Lee 2016).

4.4 Value of Analysed Smartphone Data

Current smartphone technology equips devices with increasing storage capacity, permitting large quantities of smartphone data to reside in internal storage using various storage structures. As smartphone usage becomes more prevalent, interest regarding the data stored on the devices continues to grow. Recently, multiple research studies have focused on the collection and analysis of smartphone data.

Eagle and Pentland (2006) introduced a system for sensing complex social systems that use standard Bluetooth-enabled smartphones to measure information access. The developed system shows the value of the collected data to uncover structure and regu-

lar rules in the behaviour of individuals and organisations (Eagle and Pentland 2006). Mobivis is a visual analytic tool for exploring smartphone data by presenting social and spatial information as one heterogeneous network. The tool supports the temporal and semantic filtering through an interactive timeline and can represent both individual and group behaviour (Shen and Ma 2008). GroupUs, proposed by Do and Gatica-Perez (2011), is a probabilistic relational model for sensing group interaction. The model uses Bluetooth data collected by smartphones to analyse long-term dynamic social networks created by the physical proximity of people. The produced results allow for the detection of different interaction types and the discovery of different social contexts (Do and Gatica-Perez 2011). Min and Cho (2011) proposed SmartPhonebook, a tool that mines users' social network data to manage relationships by inferring social and personal contexts. SmartPhonebook uses icons and graphs to visualise social context, which allows users to understand their social situations (Min and Cho 2011). Weiss and Lockhart (2011) conducted a study that attempts to identify user traits, such as sex, height, and weight, by mining smartphone accelerometer data. Building predictive models using supervised learning methods allow for the identification of various traits from the collected accelerometer data (Weiss and Lockhart 2011). Altshuler et al. (2012) demonstrated the prediction of demographic information, such as ethnicity, age, and marital status of users, by analysing the personal features and behaviour properties of short message service (SMS) messages (Altshuler et al. 2012). Chittaranjan et al. (2013) investigated the relationship between behavioural characteristics derived from smartphone data and the self-reported Big-Five personality traits, which are extroversion, agreeableness, conscientiousness, emotional stability, and openness to experience. The behavioural characteristics included call logs, SMS logs, Bluetooth scans, calling profiles and application usage. The outcome of the study presents a detailed analysis of the relationship between smartphone usage and the Big-Five personality traits (Chittaranjan et al. 2013). Min et al. (2013) introduced a computational model that allows for the classification of contacts according to the following life facets: family, work, and social. The computational model uses call and text message logs retrieved from smartphones to extract features such as communication intensity, regularity, medium, and temporal tendency. Combining these features with machine learning techniques showed the classification of contacts

with 90% accuracy (Min et al. 2013). Do and Gatica-Perez (2014) also focused on studying location characterisation of people's everyday activities by using smartphones that continuously record data. The collected data allows for the studying of human mobility, including the identification of visiting patterns and the categorisation of places visited. The automatic labelling of locations only uses smartphone data, without relying on any geographical information (Do and Gatica-Perez 2014).

The described mining techniques focus mostly on the analysis of demographic information, personality traits, and behavioural characteristics. The reliance on and analysis of smartphone data to obtain such valuable information emphasises the need to establish the authenticity of the related data. Confirming the authenticity of smartphone data allows digital forensic professionals to draw accurate conclusions from the analysed smartphone data.

4.5 Measures to Ascertain the Authenticity of Smartphone Data

The forensic value of analysed smartphone data and the importance of such data during criminal or civil cases emphasise the need to confirm the authenticity of smartphone data. Authenticity refers to the preservation of smartphone data when first created and the ability to prove the integrity of the data over time (Losavio 2005, Casey 2011, Cohen 2012). Smartphone data is susceptible to change and can be altered intentionally or accidentally by end-users or installed applications, which impacts the authenticity of the data. Digital forensic professionals must, therefore, determine the authenticity of smartphone data, confirming the data refers to actual events before formulating any conclusions.

Physical device controls, such as a PIN/passcode, password, gesture (pattern between discrete nodes) or biometrics (physical characteristic of the end-user), restricts access to a smartphone (Abalenkovs et al. 2012) and only permit authorised end-users to access the smartphone data. These controls ensure the confidentiality and availability of the smartphone data. The integrity of smartphone data can be maintained using encryption, which is supported by Android and iOS mobile operating systems. Encryption

on Android is available since version 3.0 but remains an optional setting for end-users (Abalenkovs et al. 2012). However, iOS supports a dedicated Advanced Encryption Standard 256-bit cryptographic engine that encrypts smartphone data by default (D’Orazio 2013). Measures, such as device locks and encryption techniques, primarily assist with data protection but can also allow digital forensic professionals to determine the authenticity of smartphone data.

Various measures proposed by researchers can also assist digital forensic professionals to establish and confirm the authenticity of smartphone data. Research conducted by Pieterse et al. (2015) introduced a new solution, called the Authenticity Framework for Android Timestamps (AFAT), which allows examiners to establish the authenticity of timestamps found on Android smartphones. The framework determines the authenticity of timestamps found in SQLite databases by using two methods. The first method explores the Android file system (EXT4) for the presence of certain changes, which are indicators of the potential manipulation of the SQLite databases. The second method identifies inconsistencies in these SQLite databases. The presence of specific file system changes along with inconsistencies in the associated SQLite databases indicates the authenticity of the stored timestamps might be compromised (Pieterse et al. 2015). Verma et al. (2014) described a technique that can be used to identify the malicious tampering of dates and timestamps in Android smartphones. The proposed technique follows a reactive approach by gathering kernel-generated timestamps of events and storing these timestamps in a secure location outside the Android smartphone. In the case of a digital forensic investigation, the preserved timestamps are available to establish the authenticity of times and dates in question (Verma et al. 2014). Govindaraj et al. (2015) designed a solution, called iSecureRing, to secure iOS applications and preserve dates and timestamps for later use. The solution consists of two modules: the first module wraps the iOS application in an additional layer of protection while the second module preserves authentic dates and timestamps of events that relate to the application (Govindaraj et al. 2015).

Physical device controls, encryption and the presented measures can all contribute to the assessment of the authenticity of smartphone data. However, both the available controls and proposed measures include several limitations further highlighted in the

following section.

4.6 Limitations of Authenticity Measures

Traditional software applications include safeguards, such as audit logs or integrity checks (Thomson 2013), that can support the validity and reliability of data. Although such safeguards can assist digital forensic professionals to ascertain the authenticity of smartphone data, smartphones generally do not support sophisticated logs or similar safeguards. Meanwhile, existing commercial mobile forensic toolkits, namely Cellebrite Universal Forensic Device (UFED) and FTK Mobile Phone Examiner, provide limited support in establishing authenticity (Verma et al. 2014). This was confirmed by the detail review of mobile forensic toolkits in Section 2.5.3.

The measures presented in Section 4.5 provide digital forensic professionals with various options to evaluate smartphone data, especially with regards to the authenticity of the data. The presented measures are, however, limited in the applicability across various smartphone platforms. Each measure only supports a single platform, either Android or iOS, and this impacts the effectiveness of the measures to evaluate smartphone data collected from various smartphones. Certain measures also require additional software to be installed before seizing smartphones for investigation. Although this may be possible in an organisational environment where smartphones are provided to end-users, privately-owned smartphones generally will not include such software. Digital forensic professional must still be able to assess the authenticity of smartphone data without requiring the installation of additional software. Finally, all of the presented measures only focus on the evaluation of a subset of smartphone data namely timestamps and do not support the assessment of other forms of smartphone data.

The limitations of these presented measures, which are either platform-specific, evaluate only a specific subset of smartphone data or require additional software to be installed before the examination, emphasise the need for new and advanced measures to enable digital forensic professionals to determine the authenticity of smartphone data. These new measures to evaluate smartphone data must function independent of any specific smartphone platform and eliminate the need for the installation of additional software.

4.7 Summary

Smartphones are responsible for the creation and storage of large quantities of smartphone data. The available smartphone data, which resides directly on smartphones, can be categorised into one of the following sources: SIM card, device-specific, user-created, sensor or application-related. Storage structures, especially SQLite databases, collect and store these various sources of smartphone data, which can provide valuable information when analysed efficiently during examinations. To ensure the accuracy of the analysed smartphone data, the authenticity of such data is of great importance. Smartphones and the related application generally do not deploy safeguards that can confirm the authenticity of data. The limitations of existing controls and authenticity measures emphasise the need for new measures to determine the authenticity of smartphone data. Such measures must be platform-independent and incorporate a broader set of smartphone data. Since application-related data provide digital forensic professionals with the most significant subset of smartphone data, a promising solution is to consider the structure and behaviour of smartphone applications. Changes in the behaviour of smartphone applications can offer the first indication that the authenticity of the related data may be affected.

Chapter 5

Reference Architecture for Smartphone Applications

The previous chapter described various aspects of smartphone data. Although various sources of smartphone data exist, application-related data provides the most extensive collection of data on smartphones. Such large quantities of application-related smartphone data are a direct result of the frequent use of many smartphone applications by end-users to accomplish everyday activities. The sheer volume and value of application-related data emphasise the importance of such data during criminal or civil cases. Therefore, the remainder of this thesis focuses specifically on application-related data residing on smartphones.

With the focus strictly aimed at smartphone applications only, it is necessary to have a better understanding of the architecture and behaviour of such applications. This chapter introduces a new reference architecture for smartphone applications that model the components and expected behaviour of applications. The reference architecture is specifically designed to enable digital forensic professionals to quickly comprehend the modelled smartphone application and understand how the associated data originates, which is necessary to establish the authenticity of the smartphone data. In Section 5.1 existing reference architectures are presented, which highlights the need for a reference architecture to model smartphone applications. Section 5.2 describes the reference architecture derivation process that includes the design of conceptual architectures for

Android and iOS applications, as well as the identification of similarities between these architectures. The reference architecture for smartphone applications is introduced in Section 5.3, focusing on the components and modelling the behaviour of smartphone applications. Section 5.4 illustrates the value, purpose and necessity of the reference architecture by modelling an existing smartphone application. Section 5.5 concludes the chapter.

5.1 Existing Reference Architectures

Reference architecture for a domain is an architectural template for all the software systems in that domain. The architectural template defines the fundamental subsystems of the domain, as well as the relationships between these subsystems (Eixelsberger et al. 1998, Bergey et al. 1999). The presence of a reference architecture for a specific domain improves the understanding of the given system since the reference architecture serves as a template to analyse or re-engineer existing systems (Grosskurth and Godfrey 2005).

Reference architectures exist for mature domains, such as operating systems, compilers, web browsers and web servers. Their significant subsystems and related relationships have been studied thoroughly and are well understood (Hassan and Holt 2000). An operating system has the following systems: file system, memory manager, process scheduler, network interface and an inner process communication system (Tanenbaum 2009). Similarly, a software compiler consists of a scanner, parser, semantic analyser, and a code generator (Shaw and Garlan 1996). A reference architecture for web browsers, derived by Grosskurth and Godfrey (2005) using Mozilla and Konqueror open source web browsers, comprises of the following subsystems: user interface, browser engine, rendering engine, networking subsystem, JavaScript interpreter, XML parser, display back-end and data persistence (Grosskurth and Godfrey 2005). Hassan and Holt (2000) proposed a reference architecture for web servers using source code and available documentation of existing web servers. The web server reference architecture consists of seven subsystems, divided between two unique layers: server and support. The server layer captures the reception, request analyser, access control, resource handler and transaction log subsystems while the support layer contains the utility and operating system abstraction layer subsystems

(Hassan and Holt 2000).

Recently, the focus of research regarding the derivation of reference architectures shifted towards mobile-related systems. Diniz et al. (2016) proposed a reference architecture for mobile crowd-sensing platforms and described the architectural design in two service models: front-end and back-end. The front-end module captures the crowd-sensing systems, which can either be web or mobile, and the back-end module contains the RESTful application programming interface services (Diniz et al. 2016). A reference architecture for mobile devices that exploits cloudlets, which are virtual machine-based code offload elements within single-hop proximity to mobile devices, was proposed by Simanta et al. (2012). The two essential components of this architecture are the cloudlet host and mobile client. The cloudlet host is a physical server that hosts a discovery service, base virtual machine image, and a cloudlet server. The mobile client is a software component for hand-held or wearable mobile devices that includes a cloudlet client application and a collection of cloudlet-ready applications. Research conducted by Zhang et al. (2010) provided a reference architecture for elastic applications, which consists of multiple components called weblets (autonomous software entities) that can be launched on either a mobile device or in the cloud. This reference architecture consists of a user interface component, one or more weblets, and a manifest. Lewis et al. (2012) designed a reference architecture for group-context-aware mobile applications that enable the integration of contextual information from individuals and nearby team members to assist with the execution of a mission. The reference architecture is modelled according to the model-view-controller (MVC) pattern and consists of three layers: user interface, application and input/output (Lewis et al. 2012). Finally, Abadi (2011) presented a generic reference architecture for mobile applications designed to model all types of mobile applications. Construction of the generic reference architecture relied on four different mobile application scenarios. However, the author concluded that the generic reference architecture is not feasible to model all types of mobile applications (Abadi 2011).

With regards to smartphone applications, there currently exists no definitive reference architecture that can accurately capture the structure and behaviour of such applications across different platforms. The remainder of this chapter focuses on the creation of a reference architecture for smartphone applications, starting with the reference architecture

derivation process.

5.2 Reference Architecture Derivation Process

Proposing and designing a new reference architecture for smartphone applications requires the evaluation of available documentation, existing domain knowledge and architectural designs of applications created for specific smartphone platforms. The process follows a black-box analysis approach by examining functionality captured in architectural designs of applications without accessing internal structures or source code. Capturing and analysing all of the available information allow for the identification of common architectural components and support the modelling of the behaviour of these components. Therefore, a reference architecture derivation process is established to ensure the derived reference architecture for smartphone applications includes the necessary components and behaviour. The reference architecture derivation process consists of the following four steps:

1. Examine available domain knowledge, existing documentation and architectural designs for applications developed for each smartphone platform.
2. Derive a conceptual architecture for each smartphone platform's applications.
3. Identify architectural similarities between the conceptual architectures.
4. Derive a reference architecture using the identified architectural similarities.

The current focus of the reference architecture derivation process examines the leading and most prevalent smartphone platforms: Google Android and Apple iOS. Since it is not feasible to design a reference architecture that models all types of smartphone applications (Abadi 2011), this particular reference architecture only focuses on modelling applications that store application-related smartphone data directly on the smartphone.

Following the established process, it is now possible to derive conceptual architectures for the applications developed for Android and iOS smartphone platforms.

5.2.1 Conceptual Architecture of Android Applications

Applications developed for Android smartphones are written and compiled using the Java programming language and the Android software development kit (SDK) (Steele and To 2011, Meier 2012, Sokolova et al. 2014). The Android SDK contains a collection of tools, sample code, and documentation, which offer the necessary support to develop Android applications. A collection of developer-generated layout resource (XML) files constructs the visual design and graphical user interface of Android applications (Steele and To 2011). The layout resource files define the view elements (such as text fields, buttons or lists) for an application, which includes their positioning, executable actions and styling criteria (Goadrich and Rogers 2011, Steele and To 2011). Style resources contain a collection of properties that maintain a consistent look and feel for elements of a single view (Meier 2012). Styling properties typically include height, width, padding and colour scheme and developers apply these properties to a specific view. Also, developers select a theme for the application to allow for consistent styling across all views of the application. The layout resources thus decouple the user interface from the application's business logic (Meier 2012), providing simplicity when defining the visual structure of an Android application.

The core components of Android applications that capture the business logic are activities, services, broadcast receivers, intents, notifications and content providers (Meier 2012). Activities form the main component of an Android application and act as the entry point for end-users to interact with the application. An Android application can consist of many activities, but only a single activity is active at any given time (Sokolova et al. 2014). An activity allows the end-user to perform a specific task (edit a note or play a game), while simultaneously capturing and responding to the end-users' actions. Activities not only interact with end-users but also with other processes such as broadcast receivers and services. A broadcast receiver is an internal messaging system (Sokolova et al. 2014) that listens for relevant broadcast messages from the system and adequately responds to these messages (Steele and To 2011). A broadcast receiver will trigger on a specific event, for example, to indicate that the battery power of the smartphone is low. Services perform longer running background tasks (downloading files) without requiring any user interaction (Steele and To 2011). Besides the use of activities, broadcast re-

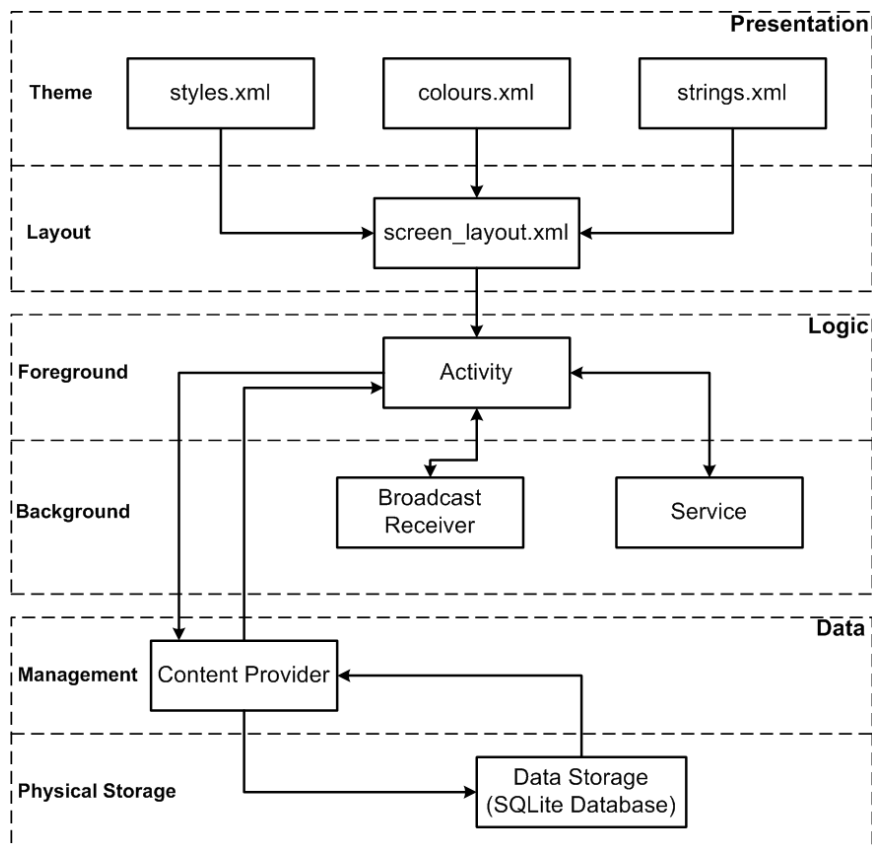


Figure 5.1: The conceptual architecture of Android applications

ceivers and services, applications developed for Android smartphones also rely on intents and notifications to allow the application to execute fluently. An Android intent is an inner application message-passing framework that is used to start or stop activities or services. Notifications alert end-users to application events without stealing focus or interrupting the current activity (Meier 2012). The final component used in Android applications is content providers. Content providers give Android applications access to locally stored data (Sokolova et al. 2014). Android applications have the following data storage options (*Android Developers* 2018a):

- Shared preferences: store private primitive data in key-value pairs.
- Internal storage: store private data on the device memory.
- External storage: store public data on the shared external storage.

- SQLite databases: store structured data in a private database.

One or more of the mentioned options can be used by developers to store application-related data.

There is no predefined architecture for the development of Android applications (Sokolova et al. 2014). It is, however, possible, based on the components identified above, to construct a conceptual architecture for Android applications. Visualised in Figure 5.1 is the conceptual architecture for Android applications, which consists of three layers: (i) presentation, (ii) logic and (iii) data.

The presentation layer captures the themes and layout resources of the application, which feeds into an activity defined in the logic layer. The activity also initiates or interacts with defined broadcast receivers and existing services. Interaction with locally stored persistent data occurs via the content provider, which will retrieve or store the relevant data. This conceptual architecture provides a template to capture and model the core components of an Android application.

5.2.2 Conceptual Architecture of iOS applications

Xcode is the preferred integrated development environment for iPhone applications. Such applications are generally developed using Swift or Objective-C, which is a superset of ANSI-C that borrows object orientation syntax from SmallTalk (Goadrich and Rogers 2011), as well as the iOS SDK (Joorabchi and Mesbah 2012). The iOS SDK imposes object orientation and the MVC architectural design pattern for iOS application development (Mark and LaMarche 2009, Iulia-Maria and Ciocarlie 2011). The MVC pattern assigns objects in an iOS application to one of three roles: (i) model, (ii) view, or (iii) controller (Sadun 2012). Since developers of iOS applications prefer to follow the MVC architectural design pattern, it is possible to base the conceptual architecture of iOS applications on this pattern. Figure 5.2 presents the conceptual architecture, which captures the core components, as well as the communication between these components for iOS applications.

The model object encapsulates the persistent data specific to an iOS application and defines the logic that manipulates and processes the data (Sadun 2012). There are four common approaches for data persistence in iOS applications (Jacobs 2015)

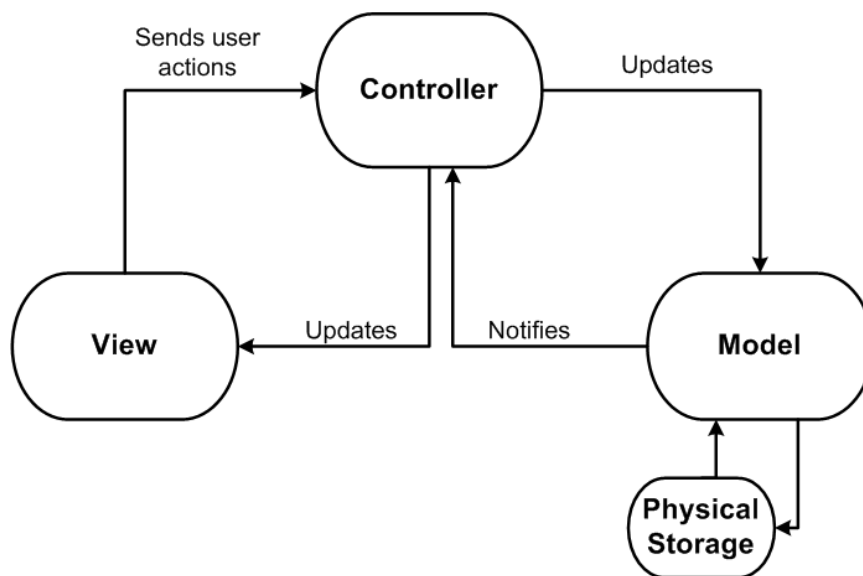


Figure 5.2: The conceptual architecture of iOS applications

- User defaults: stores user preferences.
- Property lists: hierarchically structured combinations of basic object types similar to standard XML.
- SQLite databases: stores structured data in a lightweight embedded relational database.
- Core data: provides a relational, object-oriented model serialised into XML, binary, or an SQLite store.

The view object is visible to the end-user and responds to received actions, displays data from the application's model object and allows for the editing of that data (Sadun 2012). The view object contains two principal classes: `UIView` and `UIViewController`. The `UIView` is the most abstract view class, and the `UIViewController` acts as a view handler, supporting the rotation, resizing, creation, and destruction of views. Developers populate view objects with standard user interface elements, provided by the `UIKit`, and usually include labels, buttons, tables or text fields (Iulia-Maria and Ciocarlie 2011, Joorabchi and Mesbah 2012).

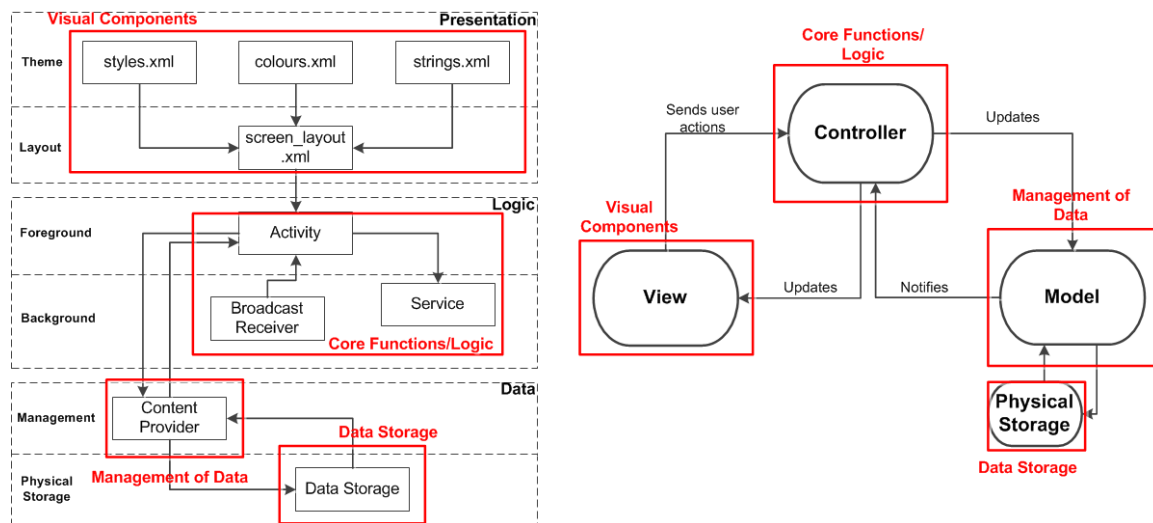


Figure 5.3: Visualisation of architectural similarities between Android and iOS applications

The controller object captures the application logic and acts as an intermediary between one or more of the application's view and model objects. The controller object implements behaviour using three key technologies: delegation, target action, and notification (Iulia-Maria and Ciocarlie 2011). These technologies perform the set-up and coordination of tasks for an iOS application (Sadun 2012).

This conceptual architecture for iOS applications, which follows the MVC design pattern, effectively captures and model the components of an iOS application.

5.2.3 Architectural Similarities

Conceptualising the architectural designs of both Android and iOS applications highlights the use of different design patterns by developers. It becomes possible, on close examination of the conceptual architectures, to identify similar architectural components for both Android and iOS applications. Figure 5.3 and Table 5.1 capture these similarities.

Examining the conceptual architectures shows that both Android and iOS applications make use of specific visual components to construct the application's graphical user interface. The core functions and logic of Android applications are developed using activities, broadcast receivers or services, while iOS applications contain the same

Table 5.1: Architectural similarities between Android and iOS applications

	Android	iOS
<i>Visual Components</i>	Themes, styles and layouts created using XML files.	View objects containing visual elements created using the UIKit
<i>Core Functions and Logic</i>	Core functions and logic contained in activities, broadcast receivers or services	Core functions and logic contained in the controller objects
<i>Management of Data</i>	Management of stored data occurs via content providers	Management of stored data occurs using model objects
<i>Data Storage</i>	Shared preferences, internal storage, external storage and SQLite databases	User defaults, property lists, SQLite databases and core data

functionality in controller objects. Content providers perform the management of data for Android applications and model objects for iOS applications. Both Android and iOS applications have multiple options to store persistent data, with SQLite databases being the preferred choice for both smartphone platforms (see Section 4.3.4).

The similarities found between the conceptual architectures of both Android and iOS applications allow for the identification of four unique architectural components. These architectural components are: (i) user interface, (ii) application logic, (iii) data management and (iv) data storage. These components allow for the construction of a single reference architecture to model both Android and iOS applications.

5.3 Reference Architecture for Smartphone Applications

The design of the conceptual architectures for both Android and iOS applications provide the necessary insight to identify common architectural components of applications devel-

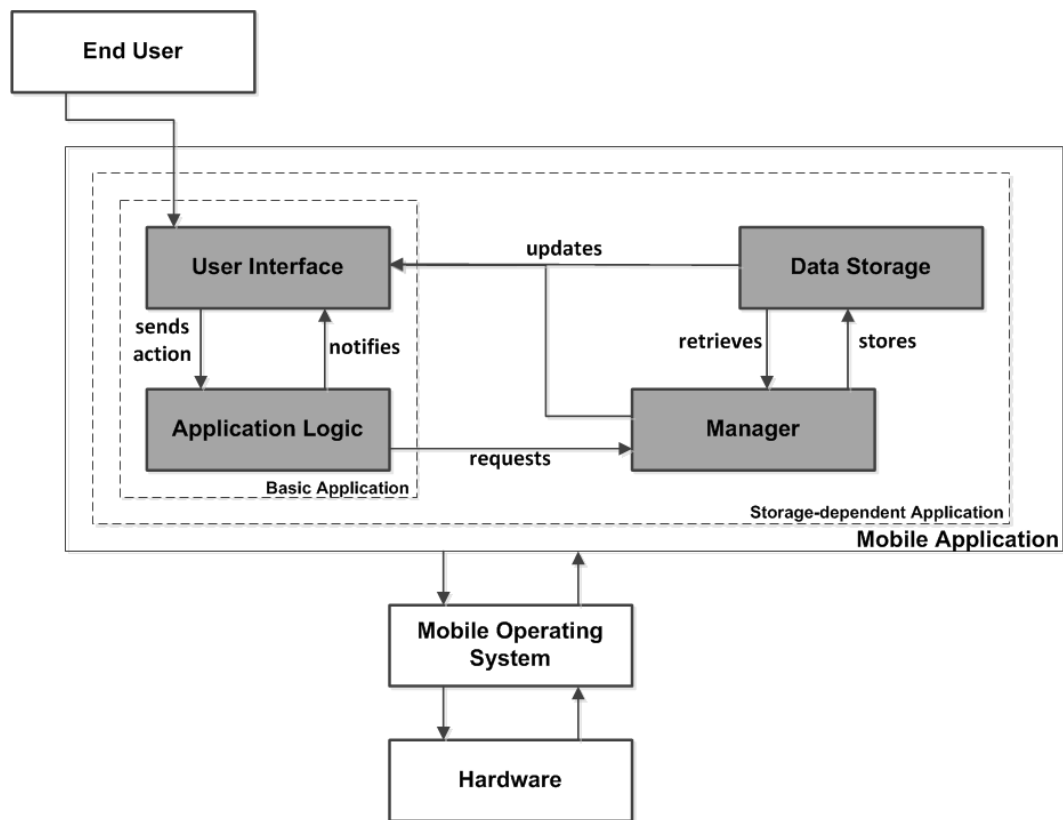


Figure 5.4: Reference architecture for smartphone applications

oped for both smartphone platforms. The four similar architectural components identified form the core of the Reference Architecture for Smartphone aPPLICATIONS (RASPP). RASPP is illustrated in Figure 5.4 and shows the architectural ordering of the components, as well as the interaction that occurs between these components.

The flexible ordering of the architectural components of RASPP supports the modelling of smartphone applications at different levels of complexity. Smartphone applications that do not rely on persistent storage of data, such as a Calculator or Compass application, can be modelled using only the user interface and application logic components. Such smartphone applications are called basic applications and do not require storage of persistent data. Smartphone applications relying on persistent data stored directly on the smartphone, called storage-dependent applications, are modelled using all four core components of RASPP. Storage-dependent applications include, but are

not limited to, messaging applications (WhatsApp, Facebook Messenger), web browsers (Chrome, Safari), and media applications (Camera).

These architectural components and their internal behaviour are further explored in the following sections.

5.3.1 Components of Reference Architecture

The grey blocks in Figure 5.4 represent the core architectural components of RASPP. The first of these components, called the user interface, captures the visual components and graphical design of a smartphone application. The user interface presents the point of entry where interactions between end-users and the smartphone application can occur. Such interactions allow for the effective operation of the smartphone application by permitting the end-user to execute a limited selection of events. Typical events include opening the application, writing a message, taking a picture or making a phone call. Completion of an event involves a specific action. A received action can also include data, such as the text of a message, a photograph or phone number of an outgoing phone call. Each specific smartphone application has thus predictable events, which leads to expected results.

The second component captures the core functions and workflow logic of a smartphone application and is, therefore, called the application logic component. The application logic component contains the necessary functions responsible for processing or executing the action accompanying an event received from the user interface component. During processing, data included along with an action is validated and prepared accordingly to assist with the execution of the action. Execution of an action that includes data can have one of the following outcomes:

- The action completely consumes the data.
- Parts of the data is kept in an original form and produced as part of the result of the action.
- The action generates additional data when executed.

- All or parts of the data received or produced during the execution of an action are stored.

Once the execution of the action completes, and the required data is either stored in or retrieved from data storage, the application returns to the user interface component and updates it accordingly.

The third component, called the manager component, is responsible for providing access to data storage (files or SQLite databases) to allow for storage or retrieval of data. The manager component receives the data from the application logic component. Furthermore, the manager component is also responsible for ensuring the validity of data, as well as transforming the data into a suitable format for either storage or presentation. The manager component, therefore, ensures that the applicable data is suitable for either visual presentation in the user interface component or storage.

The final component of RASPP is the data storage component. The data storage component is responsible for storing the persistent data and also making the stored data accessible for retrieval. The functions captured by the data storage component allows for the creation, update, insertion, or removal of records in SQLite databases, as well as the reading and writing of files.

Currently, the design of RASPP allows for the high-level modelling of smartphone applications. RASPP identifies only the core components of the modelled smartphone application and reveals the minimal interaction between the architectural components. Evaluation of smartphone data requires the modelling of smartphone applications at a finer level of granularity, which necessitates further exploration of the behaviour of smartphone applications.

5.3.2 Behavioural Models of Smartphone Applications

The interactions that occur within and between RASPP's architectural components represents the internal behaviour of smartphone applications. Modelling the interaction of the core architectural components provides additional insight into the expected behaviour of smartphone applications. The subsequent sections capture the expected behaviour of smartphone applications using a high-level data flow diagram and a finite state machine.

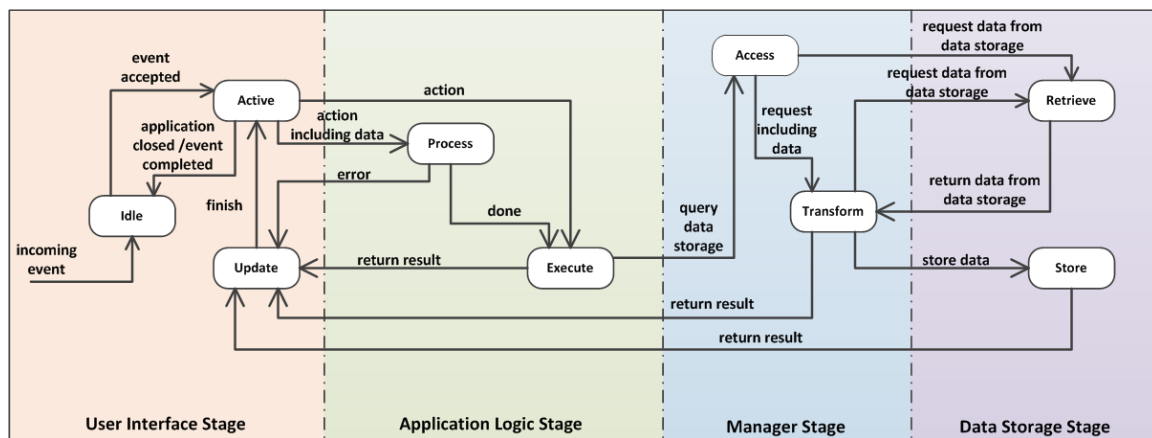


Figure 5.5: Modelling the behaviour of smartphone applications using a data flow diagram

Internal Behaviour of Smartphone Applications

The high-level data flow diagram, shown in Figure 5.5, provides the first illustration of the internal behaviour of smartphone applications. The data flow diagram offers a descriptive overview of the behaviour of smartphone applications in four different stages. Each stage, which corresponds directly to the core architectural components of RASPP, is further described and elaborated below.

The user interface stage comprises of three distinct states: (i) Idle, (ii) Active, and (iii) Update. Successful installation of a smartphone application places the application in the Idle state. A smartphone application in the Idle state remains closed and inactive while awaiting events from end-users or other applications. The Idle state of a smartphone application is interrupted when the end-user opens the application, or an external event occurs (for example an incoming phone call or system-wide notification). The external event or an end-user opening the smartphone application allows the application to transition from the Idle state to the Active state. A smartphone application in the Active state receives the event and generates the appropriate action to accomplish or complete the event. Next, the smartphone application transitions from the Active state to the Process state upon creating an action that includes data. An action involving no data will cause the smartphone application to transition directly to the Execute state. A completed action allows the smartphone application to transition to the Update state. The Update state is responsible for updating the user interface according to the

completed action or, in case of failure, display an error message. Updates occur to the displayed data or elements in the user interface. Successful completion of the action allows the application to transition to the Active state. Successful completion of the event or the closure of the application by the end-user will cause the application to return to the Idle state.

The application logic stage consists of two individual states: (i) Process and (ii) Execute. A smartphone application transitions to the Process state upon receiving an action that includes data. The Process state is responsible for separating the action from the data and processes the data accordingly. The processing of the data can include the proper validation of the data or the application/removal of additional security measures such as encoding or decoding. The processing of the data can follow multiple iterations and only when done successfully will cause the smartphone application to transition to the Execute state. The processing of incorrect or erroneous data will generate an error and cause the smartphone application to instead transition to the Update state to display an appropriate message. The Execute state is primarily responsible for completing the received action. If the action involves no data, the Execute state will merely complete the action and transition to the Update state. An action involving data can cause the Execute state to have several outcomes. Firstly, the action can completely consume the data once executed and directly transition to the Update state. Secondly, the data or parts thereof must be retained in data storage. The smartphone application, therefore, transitions to the Access state to gain access to the data storage. Thirdly an action, which either includes or excludes any data, can require data currently retained in data storage. This data must be retrieved and, therefore, the smartphone application once again transitions to the Access state. Once the data is retrieved, the smartphone application returns to the Update state to complete the action.

The manager stage consists of two states called (i) Access and (ii) Transform. The Access state is responsible for accepting queries to retrieve or store persistent data. A query involving no additional data causes the Access state to transition to the Retrieve state. A query that includes data will allow the smartphone application to transition from the Access state to the Transform state. The Transform state is responsible for accepting requests, which include specific data, to either retrieve or store persistent data.

Once received, the Transform state transforms the data into an acceptable form. The data included in the request is either directly obtained from the provided action or derived when the action was successfully executed. Depending on the received request the Transform state transitions to either the Store or Retrieve states. The smartphone application transitions to the Store state to store the transformed data. The Transform state uses the transformed data to initiate the request to retrieve persistent data before transitioning to the Retrieve state. Finally, to complete an action requiring the transformed data retrieved from data storage, the smartphone application transitions to the Update state.

The data storage state comprises of two states: (i) Retrieve and (ii) Store. The Store state accepts transformed data from the Transform state and stores the data either in a database or file. Once stored, the smartphone application transitions to the Update state. The Retrieve state receives requests from either the Access state (no data involved in the request) or the Transform state (data involved in the request) and fetches the required data currently retained in data storage. Once completed, the smartphone application transitions to the Transform state to transform the data into an acceptable form. Finally, upon receiving the correctly formatted data causes the smartphone application to transition to the Update state.

Finite State Machine

The high-level data flow diagram provides a descriptive overview of the internal behaviour of smartphone applications. A more abstract and extensible model of smartphone application behaviour is presented using a finite state machine. A finite state machine describes an idealised machine that illustrates the essential idea of a sequential circuit. Each piece of input provided to a finite state machine leads to a change in the current state of the machine, which in turn affects how subsequent input will be processed (Epp 2004). The finite state machine that captures the internal behaviour of smartphone applications is shown in Figure 5.6.

The finite state machine organises the states using the same colour schemes used in the data flow diagram to relate to the core architectural components of RASPP directly. Maintaining the presence of the core architectural components, as well as the same

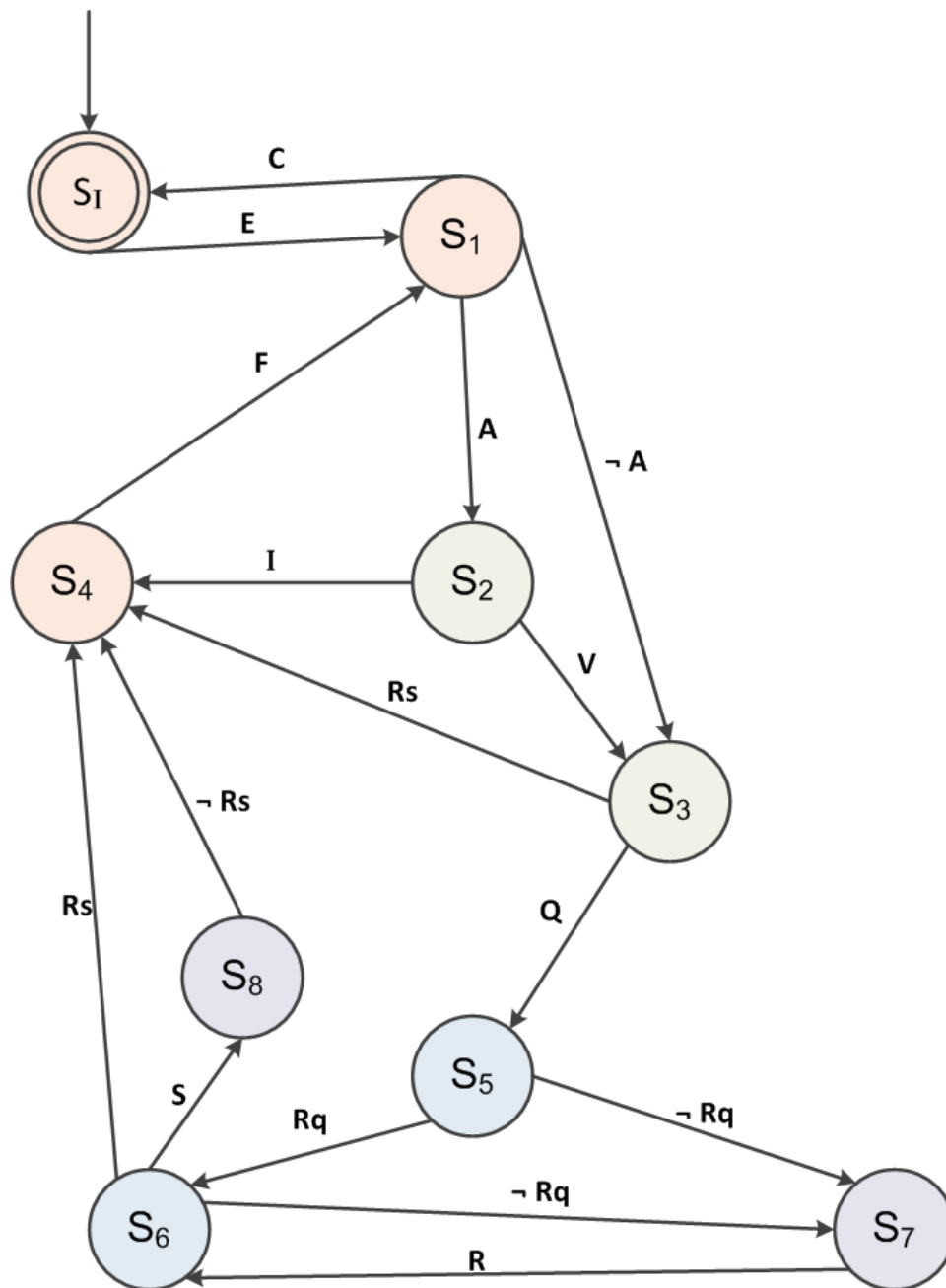


Figure 5.6: Modelling the behaviour of smartphone applications using a finite state machine

colour schemes, allows for better understanding and interpretation of the smartphone application behaviour. The states of the finite state machine, as depicted in Figure 5.6, are further explained below:

- S_1 indicates the smartphone application is active and received an event (E). The event received creates an action that either includes (A) or excludes ($\neg A$) data. Completing the event or closing (C) the smartphone application causes a return to the Idle state.
- S_2 is responsible for processing the data included in an action. The processed data can either be invalid (I) or valid (V).
- S_3 deals with the execution of the action. An executed action either immediately returns a result (Rs) or requires the storage or retrieval of persistent data (Q).
- S_4 is responsible for updating the user interface of the smartphone application according to the received result. Once finished (F), the smartphone application is again ready to receive action.
- S_5 allows the smartphone application to access the data storage, which generates a request that either includes (Rq) or excludes ($\neg Rq$) data.
- S_6 is responsible for transforming received data into a suitable working format. The transformed data is returned as a result (Rs), requested to be stored (S), or form part of a request to retrieve existing data (Rq) in data storage.
- S_7 retrieves existing data from data storage and returns the retrieved data (R).
- S_8 stores data in data storage and returns a result that excludes any additional data ($\neg Rs$).
- S_I is both the initial and accepting state and indicates the smartphone application is idling in the background. A received event (E) interrupts the idle state.

Besides the described states (Q), the finite state machine also consist of an input alphabet (Σ), identification of the initial (q_o) and accepting state (F), and a specification for a next-state function (δ) that defines which state is produced by each input in each state (Epp 2004). The full functionality of the finite state machine for smartphone application behaviour is captured using the following mathematical notation:

$$Q = \{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_I\}$$

$$\Sigma = \{C, E, A, \neg A, F, I, V, Rs, \neg Rs, Q, Rq, \neg Rq, R, S\}$$

$$\delta = \{\delta(S_1, A, S_2), \delta(S_1, \neg A, S_3), \delta(S_2, I, S_4), \delta(S_2, V, S_3), \delta(S_3, Rs, S_4), \delta(S_3, Q, S_5), \\ \delta(S_4, F, S_I), \delta(S_5, Rq, S_6), \delta(S_5, \neg Rq, S_7), \delta(S_6, Rq, S_7), \delta(S_6, S, S_8), \delta(S_6, Rs, S_4), \\ \delta(S_7, R, S_6), \delta(S_8, \neg Rs, S_4), \delta(S_1, C, S_I), \delta(S_I, E, S_1)\}$$

$$q_o = S_I$$

$$F = S_I$$

It is possible to infer a state transition table using the finite state machine drawn in Figure 5.6 and the provided mathematical model. The state transition table, captured in Table 5.2, shows the transition between states based on the received input. Also, the state transition table also reveals the deterministic nature of the finite state machine since each input causes the state machine to transition to a specific next state.

The finite state machine is responsible for modelling the inner workings and behaviour of smartphone applications at a finer level of detail. Using both the high-level data flow diagram and the finite state machine now allows digital forensic professionals to easily comprehend the expected or normal behaviour of the modelled smartphone application.

Table 5.2: State transition table

Input \ State	A	$\neg A$	C	V	I	Rs	Q	F	Rq	$\neg Rq$	S	Rs	Rd	$\neg Rs$	E
S_1	S_2	S_3	S_I	-	-	-	-	-	-	-	-	-	-	-	-
S_2	-	-	-	S_3	S_4	-	-	-	-	-	-	-	-	-	-
S_3	-	-	-	-	-	S_4	S_5	-	-	-	-	-	-	-	-
S_4	-	-	-	-	-	-	-	S_1	-	-	-	-	-	-	-
S_5	-	-	-	-	-	-	-	-	S_6	S_7	-	-	-	-	-
S_6	-	-	-	-	-	-	-	-	S_7	-	S_8	S_4	-	-	-
S_7	-	-	-	-	-	-	-	-	-	-	-	-	S_6	-	-
S_8	-	-	-	-	-	-	-	-	-	-	-	-	-	S_4	-
S_I	-	-	-	-	-	-	-	-	-	-	-	-	-	-	S_1

5.3.3 Smartphone Application Characteristics

The proposed reference architecture and behaviour models provide an abstraction of smartphone applications. The available behaviour models also support the evaluation of various smartphone applications at different levels of complexity. Also, the behaviour models enable digital forensic professionals to more easily comprehend the application's behaviour and understand how the related data originated. From this abstraction, it is possible to identify the following general characteristics regarding smartphone applications:

- Only the smartphone application can access and update the persistent data.
- Persistent data of a smartphone application is only accessible via an executed event.
- Data displayed by the user interface directly corresponds to the persistent data.
- Changes to persistent data by a smartphone application can only occur after an event is received.
- Only the end-user (human operator), the current smartphone application (internal event) or another smartphone application (external event) can provide the event.
- A smartphone application only accepts a fixed set of events.
- An event in the fixed set leads to an expected result.
- Each event generates a specific action.

These characteristics capture the behaviour of smartphone applications. Changes to one or more of these characteristics indicate changes to the behaviour of smartphone applications. Such changes offer the first indication that the authenticity of the related data may be affected.

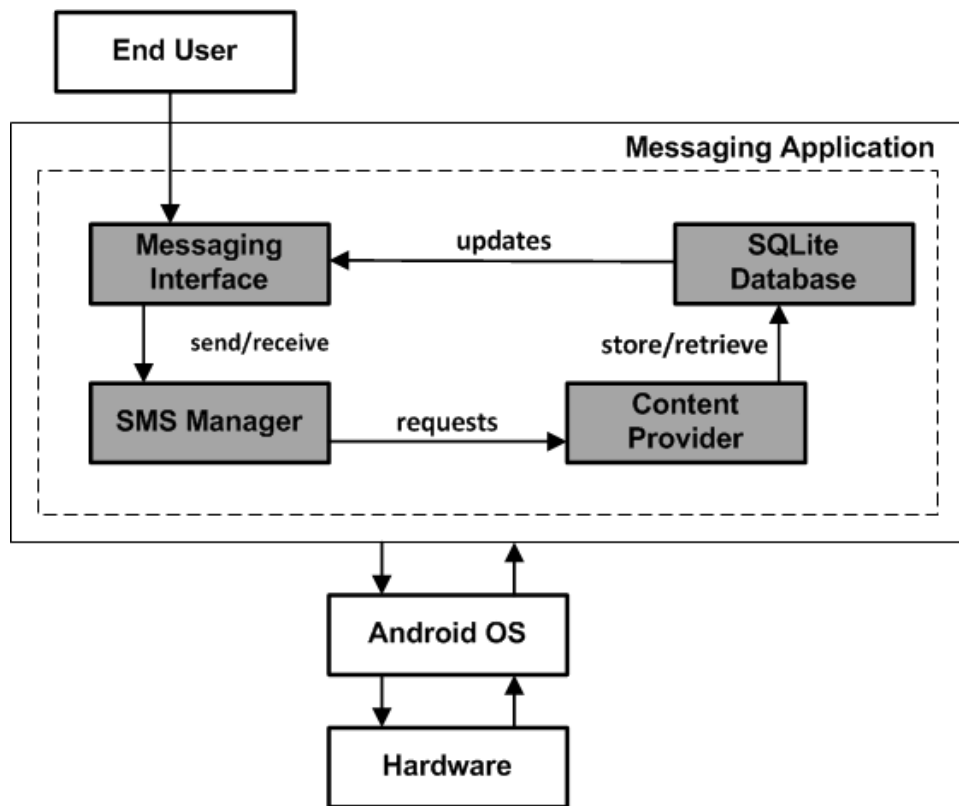


Figure 5.7: Reference architecture for Android's default messaging application

5.4 Modelling an Application using Reference Architecture

The proposed reference architecture, along with the behaviour models allow digital forensic professionals to model various applications. Modelling Android's default messaging application using the proposed reference architecture and behaviour models further illustrates the purpose and value of RASPP. It is possible to affirm the behaviour of Android's default messaging application by closely monitoring the application during execution on a smartphone.

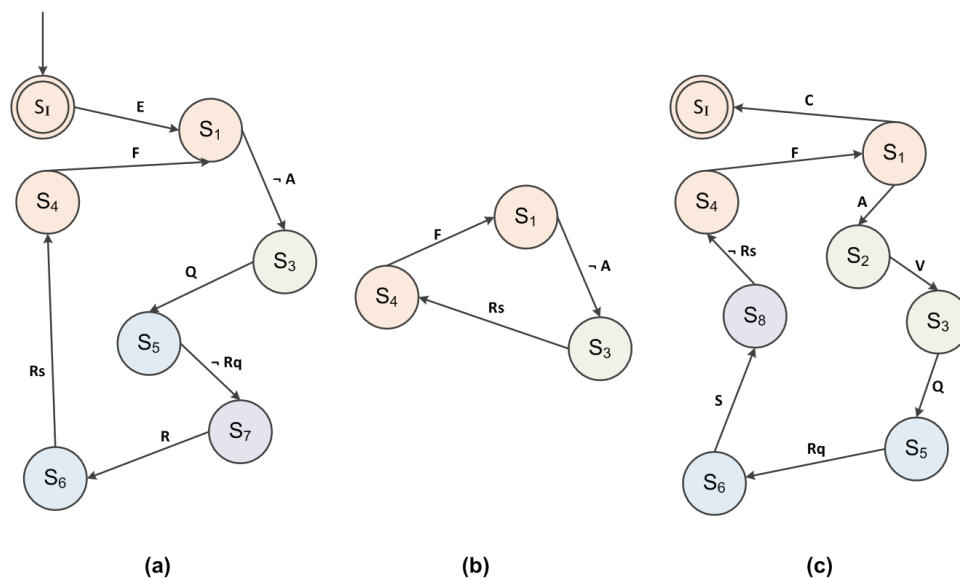


Figure 5.8: Modelling the behaviour of Android’s default messaging application

5.4.1 Modelling Expected Application Behaviour

Android smartphones ship with messaging functionality and end-users can use the default messaging application that comes pre-installed to send and receive text or multimedia messages. Using RASPP, it is possible to model and identify the messaging application’s core architectural components. Figure 5.7 presents the reference architecture of the messaging application. The messaging interface allows end-users to view, delete, create and receive text or multimedia messages. The SmsManager, which contains the workflow logic of the messaging application, is an existing Android class and manages the messaging operations (*Android Developers 2018b*). The SmsManager class uses public methods to fulfil the requested action, such as sending or receiving a message. The content provider handles access to the persistent data and assists with the necessary transformation of the data into a suitable format for storage or viewing. The suitable format includes the creation and formatting of timestamps and the extraction of additional information such as the service centre number. The persistent data related to the default messaging application is retained in an SQLite database (`mmsms.db` and `mmsms.db-wal` files).

The expected behaviour of Android’s default messaging application is best illustrated

by sending a text message. This expected behaviour is captured using the finite state machine behaviour model. Figure 5.8 illustrates the sending of a text message in three distinct stages: opening the application, selecting the option to write a new text message, as well as sending the text message. Firstly, the finite state machine (Figure 5.8 (a)) shows the messaging application receives an event (open application) and proceeds to the Active (S_1) state. An internal action, which includes no data, causes a transition to the Execute (S_3) state to retrieve all the stored messages. The application proceeds to the Access (S_5) state to obtain admission to the data storage. Subsequently, the application proceeds to the Retrieve (S_7) state to obtain the messages before transitioning to the Transform (S_6) state to correctly format the messages for display. The application transitions to the Update (S_4) state to refresh the user interface and display the messages, before returning to the Active (S_1) state.

Secondly, the finite state machine shows (Figure 5.8 (b)) the messaging application receives an event from the end-user to write a new text message. This event creates an action that includes no additional data, which allows the application to proceed to the Execute (S_3) state. The Execute (S_3) state completes the action, which doesn't require access to data storage, and transitions to the Update (S_4) state. The updated user interface allows the end-user to type in the recipient(s) and the message. Once completed, the application returns to the Active (S_1) state.

Finally, the finite state machine illustrates (Figure 5.8 (c)) the event of sending a text message after providing the necessary data. The event generates an action that includes data (recipient(s) phone number and text message) by selecting the send button. The action causes the application to transition from the Active (S_1) state to the Process (S_2) state, which ensures the validity of the message. After that, the messaging application proceeds to the Execute (S_3) state to send the outgoing text message. The application must record this text message in data storage and, therefore, the application transitions to the Access (S_5) state. Once admission to data storage is obtained, the messaging application proceeds to the Transform (S_6) state to correctly format the message and then to the Store (S_8) state to record the text message. The messaging application proceeds to the Update (S_4) state to refresh the user interface and display the newly sent text message. The messaging application then returns to the Active (S_1) state

before transitioning to the Idle (S_I) state once the end-user closes the application.

The illustration of the three distinct finite state machine behavioural model captures the expected behaviour of Android's default messaging application. It is, however, necessary to confirm that this is indeed the expected behaviour by monitoring the actual execution of the default messaging application on an Android smartphone.

5.4.2 Confirming Application Behaviour

The expected behaviour captured by the finite state machine behaviour model is confirmed by monitoring the execution of Android's default messaging application on a smartphone. The test smartphone, used to capture the behaviour of the application, is a Samsung Galaxy S5 Mini running Android version 6.0.1 (Marshmallow). The critical files to monitor on the Android file system are the SQLite database (`mmsms.db` and `mmsms.db-wal`) files of the messaging application and the log files (XML format) collected in the `/data/system/recent_tasks` folder.

During the experiment, the end-user sends a new text message from the test smartphone. Analysis of the newest log file created in the `recent_tasks` folder, which is presented in Figure 5.9, shows the messaging application (`com.android.mms`) was first active (application opened) on December 11, 2017, at 17:46:00.144 GMT+02:00. Figure 5.10 shows the captured message in the SQLite database, viewed using the SQLite Expert Personal application. Reviewing the SQLite database (`mmsms.db` and `mmsms.db-wal` files) confirms the database captured the outgoing text message on December 11, 2017, at 17:46:31.518 GMT+02:00 (also see Figure 5.11). Closer inspection of the log file shown in Figure 5.9 confirms that the messaging application was last active (application closed) on December 11, 2017, at 17:47:02.761 GMT+02:00.

From the descriptions above and the captured timestamps collected from the test smartphone, it is possible to confirm the behaviour of Android's default messaging application. The capture timestamps show the messaging application transitions from the Idle (S_I) to the Active (S_1) state when opened by the end-user. Subsequently, the messaging application sends the text message and captures the message in the SQLite database. Storing the text message causes the messaging application to transition between the Access (S_5), Transform (S_6) and Store (S_8) states. Once completed, the

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<task task_id="26" real_activity="com.android.mms/.ui.ConversationComposer" affinity="
"android.task.mms" root_has_reset="true" auto_remove_recents="false" asked_compat_mode="false"
user_id="0" effective_uid="10051" task_type="0" first_active_time="1513007160144" last_active_time
="1513007222761" last_time_moved="-1513007222753" never_relinquish_identity="true"
task_description_color="ffe89c05" task_description_text_color="ff000000" task_affiliation_color="
-1532923" task_affiliation="26" prev_affiliation="-1" next_affiliation="-1" calling_uid="10010"
calling_package="com.sec.android.app.launcher" resizable="false" privileged="true"
is_private_mode="false">
<intent action="android.intent.action.MAIN" component="com.android.mms/.ui.ConversationComposer"
flags="10200000">
<categories category="android.intent.category.LAUNCHER" />
</intent>
</task>
```

Figure 5.9: Snapshot of the newest log file in the /data/system/recent_tasks folder

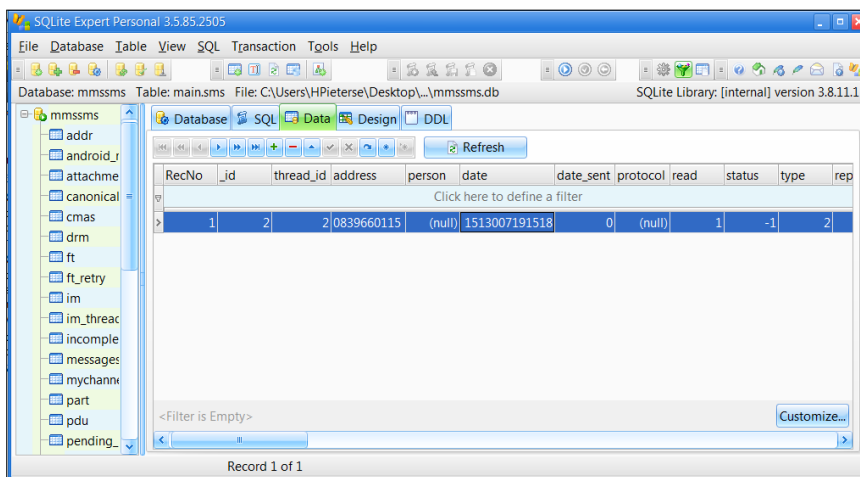


Figure 5.10: Snapshot of the SQLite database of the default messaging application

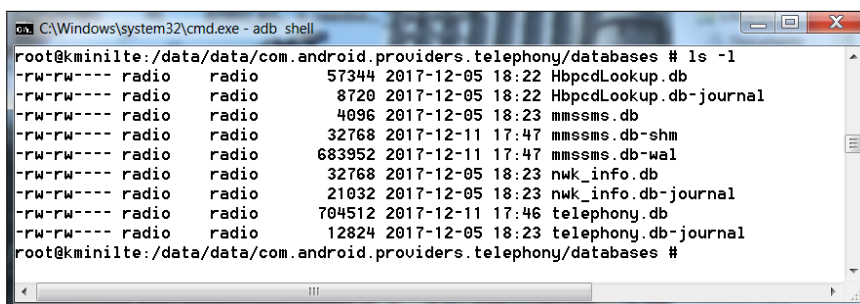


Figure 5.11: SQLite database timestamps after sending the text message

messaging application is closed, allowing for the transition from the Active (S_1) state to the Idle (S_I) state.

Although the captured timestamps do not reflect all the various transitions presented in the finite state machine behaviour model, the flow of the timestamps do correspond with the model. Modelling Android default messaging application using the behaviour models of the reference architecture offers valuable insights into the origin of the smartphone data and the expected behaviour of the application.

5.5 Summary

As with all forms of smartphone data, application-generated data is also susceptible to change and unknown alterations. Changed or manipulated smartphone data can cause misleading results when analysed. It becomes essential to establish a link between the smartphone data and the related smartphone application to ensure the authenticity of the data. Understanding the internal structure and behaviour of a smartphone application permits the identification of data that originated as a result of the expected behaviour of applications. The presented reference architecture allows digital forensic professionals to better and more accurately capture the core architectural components and expected behaviour of modelled smartphone applications. Modelling Android's default messaging application showed the reference architecture serves as a valuable template to gain a good understanding of the expected behaviour of the modelled application. Also, the modelling of smartphone applications provides the first step to determine the authenticity of the related data. Evaluation of smartphone application behaviour alone is, however, insufficient to establish the authenticity of smartphone data. Additional requirements are required to evaluate and confirm the authenticity of smartphone data.

Chapter 6

The Authenticity of Smartphone Data

Smartphone data, especially application-generated data, becomes essential when smartphones form part of a criminal or civil case. However, smartphone data is susceptible to change and can be manipulated, fabricated or removed intentionally or unintentionally by end-users or installed applications. It is, therefore, imperative for digital forensic professionals to establish the authenticity of smartphone data before formulating any conclusions. Establishing the authenticity of smartphone data requires digital forensic professionals to have a better understanding of the applications responsible for creating the data. Digital forensic professionals achieve such an understanding by modelling the applications using the reference architecture introduced in Chapter 5. From the modelled smartphone application, digital forensic professionals can infer inconsistencies with regards to the behaviour of the application, which impact the authenticity of the related data. Smartphone applications are, however, not the only role player that can influence the authenticity of smartphone data.

This chapter presents the requirements to measure authentic smartphone data. These requirements are captured in a model to assist digital forensic professionals with the evaluation of smartphone data. Section 6.1 describes the term “authenticity” and Section 6.2 presents a formal description of authentic smartphone data. The purpose of Section 6.3 is to describe the requirements for authentic smartphone data while Section

6.4 introduces the smartphone data evaluation model, which is constructed using the identified requirements. Section 6.5 evaluates the benefits and limitations of the model while Section 6.6 concludes the chapter.

6.1 Authenticity

Data available on a smartphone provides digital forensic professionals with valuable insights about the interactions that took place involving the smartphone. However, the data is vulnerable to change, and doubts of authenticity stem from numerous concerns such as undetectable fabrication, intentional manipulation or accidental alteration (Losavio 2005, Hannon 2014). To exclude unreliable data, digital forensic professionals must have access to the necessary techniques to establish the authenticity of smartphone data before arriving at conclusions.

Authenticity, in common law environments, means that something is what it claims to be (Losavio 2005) and forms part of the five fundamental requirements that must be considered when assessing the admissibility of digital evidence (Duranti and Endicott-Popovsky 2010). Proof of authenticity is required because before digital data can be admitted into evidence, it is important to show the digital data is what it claims to be (Losavio 2005, Casey 2011, Hannon 2014). Authenticity describes a digital record that has not been tampered with or corrupted, either intentionally or accidentally. An authentic digital record is, therefore, a record that (1) preserves the same identity it had when first created and (2) can be presumed or proven to have maintained its integrity over time (Duranti 2009, Cohen 2012). Identifying and preserving the authenticity of digital records are responsibilities that shift from individual to individual managing the records. Such responsibilities involve the lawful custody of the digital records, establishing the trustworthiness of the system responsible for creating the records and various preservers handling the records, who must guarantee the authenticity over the entire lifecycle of the records (Cohen 2012).

The vulnerable nature of digital data makes authenticity a vital issue (Losavio 2005). Digital forensic professionals must be aware of the authenticity of digital data in order to make informed decisions regarding the use of the digital data. The following section

highlights processes and techniques that can assist with the identification of authentic data, especially authentic smartphone data.

6.2 Authentic Smartphone Data

The standard and intended operation of smartphones by end-users create a sophisticated, interconnected environment that involves several components. These components are continuously active, interacting and communicating at varying degrees. Such interaction between the components within this interconnected environment leads to the creation of digital data. Part of the digital data is persistent smartphone data, created by storage-dependent smartphone applications and stored directly on smartphones. The components operating in this environment that are directly responsible for the creation of such smartphone data are end-users' use and operation of smartphones and the installed applications, execution of those applications, and the mobile network operators.

The following two examples best illustrate the creation of smartphone data. The first example illustrates bidirectional communication between two smartphones. An end-user sends a text message using a messaging application installed on a smartphone. This text message is delivered to the recipient via the mobile network infrastructure, who then views the received message using a similar messaging application installed on a smartphone. The second example demonstrates unidirectional communication between an end-user and the smartphone. The end-user uses a pre-installed application to access the camera and capture a photograph that is stored on the smartphone. From the examples briefly described above, it is possible to extract the following common elements involved in the creation and management of smartphone data:

- End-user's interaction with the and operation of the smartphone (open and close applications).
- End-user's usage of the installed smartphone applications to perform actions (create text messages or take photographs).
- Operation of the smartphone application to complete received actions (send a text message or store a photograph).

- The role of the mobile network operator as a delivery platform (deliver text message).

These common elements lead to the establishment of four core components: (i) end-user behaviour, (ii) smartphone operational state, (iii) smartphone application behaviour and (iv) external environment. Authentic smartphone data originates as a direct result of the expected operation and regular execution of these four core components.

Authenticity, as described in Section 6.1, refers to digital data that has not been tampered with and remained unchanged over time. For smartphone data to be authentic, it is necessary that the four core components responsible for the creation and management of the data operate as expected and remains unaffected. These four components, thus, form critical pillars in maintaining the authenticity of smartphone data. Any component that is affected and operates unexpectedly directly impacts the authenticity of the smartphone data. It is, therefore, necessary for digital forensic professionals to be able to confirm the reliability of these components for the smartphone data to be deemed to be authentic.

6.3 Requirements for Authentic Smartphone Data

Identification of authentic smartphone data necessitates the confirmation of the standard use and operation of the core components recognised in this interconnected environment. Performing such confirmation is possible by forming a collection of requirements to measure each component. These requirements capture the expected operational behaviour of each component. Digital forensic professionals can use these requirements to assess the components and from the assessment, determine whether the evaluated smartphone data is indeed authentic.

6.3.1 End-user Behaviour

The first core component encapsulates the end-users and their use of smartphones. These requirements focus on the expected operation of both the smartphones and the installed applications. These requirements assess the usage of smartphone applications, the op-

eration of the smartphone with regards to rebooting and eliminating the presence of anti-forensic applications.

Consistent Application Usage

The design of storage-dependent smartphone applications only permits end-users (human operator, existing smartphone application or another smartphone application) to initiate an event that retrieves or stores persistent smartphone data. The end-user, using the smartphone application, must execute a collection of actions to complete the event that will store or retrieve the persistent smartphone data. It is, therefore, necessary to confirm the usage of the smartphone application when changes to the persistent data occurred. Intentional alterations made to the persistent smartphone data will not involve the direct usage of the smartphone application. The requirement of consistent application usage verifies the end-user used the smartphone application to access or affect changes to the persistent smartphone data. The verification is possible by determining an appropriate time frame that stipulates when changes to the persistent smartphone data should reflect after application usage. Such changes that fall within the pre-determined and agreed upon time frame increases the authenticity of the related smartphone data.

Rarely Rebooted Smartphone

The social media-driven era of the 21st century compels people always to be online, connected and available (Hanna et al. 2011). Such needs cause end-users always to keep their smartphones powered and switched on. Smartphones are, therefore, not regularly turned off or rebooted. A regularly rebooted smartphone may offer an indication of potential changes made to a smartphone application's data since a system reboot ensures intentional changes made to the smartphone data reflects in the user interface of the smartphone application. Such a system reboot will generally occur after making the intentional changes to the smartphone data. The rarely rebooted smartphone requirement evaluates timestamps associated with a system reboot. Such timestamps that follow closely after the modification of smartphone data indicates the authenticity of that data may be affected.

Anti-forensic Applications Eliminated

Anti-forensic applications for smartphones allow end-users to destroy, hide, manipulate or prevent the creation of digital evidence (Distefano et al. 2010, Sporea et al. 2012). Smartphone applications, such as File Shredder (Android) or iShredder (iOS), can destroy smartphone data or the data can be hidden using StegDroid or MobiStego (both Android) applications. Eliminating the presence of anti-forensic applications currently or previously installed on the smartphone adds further support to the authenticity of the smartphone data. It is, therefore, necessary to assess all installed applications for anti-forensic functionality, which includes the ability to hide, destroy, fabricate or manipulate smartphone data. The presence of anti-forensic applications installed on a smartphone may indicate the end-user used the applications to delete or alter smartphone data. It is, however, not a direct indication of intentional tampering of smartphone data of a specific application. Meeting the requirement of eliminating the presence of installed anti-forensic applications increases the authenticity of the related smartphone data.

6.3.2 Smartphone Operational State

The second core component evaluates the operational state of smartphones. The operational state of a smartphone refers to the current working state of the smartphone, which reflects how the end-user used the device. The focus of the requirements for this component is on the current state of the smartphone (whether the smartphone is rooted or jailbroken) and the presence of known critical files.

Standard Smartphone State

This thesis defines a standard smartphone as a smartphone that is not currently and has not previously been rooted (Android) or jailbroken (iOS). The standard smartphone state requirement evaluates the current state of the smartphone to determine if the smartphone is or has been rooted/jailbroken. Rooting/jailbreaking a smartphone provides access to the file system partitions usually inaccessible to end-users. The access permits the retrieval of both smartphone applications and the corresponding data stored by the applications (Lessard and Kessler 2010, Miller 2011). Although not a direct indication

of the intentional tampering of smartphone data, a rooted/jailbroken smartphone lacks the additional protection measures required for smartphone data to remain authentic.

Critical Files Present

Individuals with malicious intent that manipulate or fabricate smartphone data may attempt to impede the subsequent examination of the smartphone by removing traces created due to the intentional changes made to the data. User activity reports, system reboot logs, or in files associated with a specific smartphone application usually collect such traces. The removal of these traces can potentially hide ill-intent or non-standard activities that took place on the smartphone. For this requirement, it is necessary to confirm the presence of critical files on the smartphone. Critical files include any file that the digital forensic professional must evaluate to establish the authenticity of the smartphone data. The absence of any critical file reflects intentional changes made by the end-user and also indicates that the authenticity of the smartphone data may be affected.

6.3.3 Smartphone Application Behaviour

The third component assesses the behaviour of installed smartphone applications. The requirements for this component are, therefore, directed at evaluating the expected behaviour of smartphones applications that operate under normal conditions. These requirements evaluate the persistent data and the storage structures responsible for storing the data of a specific smartphone application.

Corresponding Data (Internally)

Storage-dependent smartphone applications include actions to retrieve or store persistent data. Such data is made visible or accessible to the end-user via the user interface of the smartphone application. For smartphone data to be authentic, the persistent data stored in databases or files must correspond to the data viewed via the user interface. This requirement confirms that the internally stored data corresponds by viewing and comparing the persistent and user interface displayed data. Evaluation of this require-

Table 6.1: SQL queries to identify inconsistent records

Query No.	Query
Q.1	CREATE TABLE temp (new-id INTEGER PRIMARY KEY AUTOINCREMENT, original-id INTEGER, timestamps INTEGER);
Q.2	INSERT INTO temp (original-id, timestamp) SELECT id, date FROM table;
Q.3	SELECT T1.original-id, T1.timestamps, (T1.timestamps - T2.timestamps) AS difference FROM temp T1, temp T2 WHERE T2.new-id = T1.new-id + 1 AND difference > 0;

ment is necessary since intentional or unintentional changes made to the persistent data may not always immediately reflect in the user interface because of cached data.

Internal Database Consistency

Storage-dependent smartphone applications have various options to store persistent data, one of the most popular options being SQLite databases (see Section 4.3.4). For smartphone data stored in an SQLite database to be authentic, the ordering of the database records in a table must be consistent. A consistent record in an SQLite database is a record that is listed correctly when ordered according to the following fields: auto-incremented primary key and a field containing a date or timestamp. Confirming the consistency of such records is necessary since it is possible to alter the data stored in SQLite databases intentionally or unintentionally. The requirement of internal database consistency evaluates the records of an SQLite database and identifies inconsistent records. Such inconsistent records are identifiable by executing a collection of SQL queries, listed in Table 6.1.

To preserve the integrity of the data stored in the original table, the first SQL query (Q.1) creates a temporary table using the CREATE TABLE statement. The temporary table contains a primary key, which is an integer value that auto-increments, and all the fields that are necessary to identify the consistent SQLite database records. The second SQL query (Q.2) populates the temporary table with the original SQLite database records using a combination of the INSERT INTO and SELECT statements. The SELECT statement selects all the records from the table currently under evaluation while the INSERT INTO statement inserts these selected records into the temporary table. It is necessary to compare the values in the timestamp field of subsequent records to confirm the consistency of the records collected in the temporary table. Since all the values in the timestamp field are expected to follow one another (each new record is appended at the end of the table), the difference between two subsequent values in the timestamp field must be smaller than or equal to zero. A positive difference is an indication of a timestamp that is inconsistent. The third and final SQL query (Q.3) performs the comparison of timestamps of subsequent records.

Applying the listed SQL queries to the SQLite database responsible for storing the persistent smartphone data allow for the identification of inconsistent records. Confirming the consistency of internal database records increases the authenticity of the related smartphone data.

File System Consistency

For modifications to occur in smartphone data, the application assigns specific owners and necessary permissions to the files holding the data. Assignment of ownership (individual and group owners) and file permissions (read/write/execute) occurs on first creation of the files. Ownership belongs to the smartphone application responsible for creating the files and is recognisable via a unique user identifier (UID). These files also include specific read/write permissions to retrieve or store data. Intentional changes made to the data collected in these files will require a change of the existing file permissions, as well as subsequent changes to the UID of the individual or group owners. The purpose of the file system consistency requirement is to identify changes to the ownership and file permissions of files relating to a particular smartphone application, which

reflects changes made to persistent data. Identification of such changes is significant since the changes impact the authenticity of the smartphone data.

Database File Consistency

SQLite version 3.7.0 preserves original records in the main database file and appends changes to a separate WAL file (`.db-wal`). The file size of the main database file of an SQLite database is expected to be smaller than the size of the related WAL file. This remains true for the first 1000 frames accumulated since the SQLite database appends new data to the WAL file. The file size of the main database file only increases once a checkpoint occurs, causing all the frames in the WAL file to be transferred to the main database file. A checkpoint, however, only occurs when the WAL file reach the limit of 1000 frames (approximately 4MB in size) (*SQLite 2017c*). The file size of the main database file thus remains relatively small for a period, depending on the use of the smartphone application. An end-user with the intent to make changes to persistent smartphone data will cause an automatic checkpoint to occur after opening the main database file to perform the changes. To prevent the newly made changes in the main database file from being overwritten by existing frames in the WAL file, it is necessary to delete the WAL file and reboot the smartphone. A successful smartphone reboot automatically generates a new WAL file. This new WAL file contains limited structural information and thus has a file size smaller than the file size of the main database file. The requirement of database file consistency confirms the consistent sizes of the main database file and the related WAL file, supporting the authenticity of the smartphone data.

6.3.4 External Environment

The final core component evaluates the environment external to the end-user and the associated smartphone. This external environment includes all other smartphones involved in bidirectional communication, as well as the mobile network operator(s). These requirements focus on the correspondence of persistent data between different smartphones, as well as data collected by the mobile network operator(s).

Corresponding Data (Externally)

Specific smartphone applications support bidirectional communication, which involves two-way communication between two smartphones such as sending text messages or making phone calls. For smartphone data to be authentic, the persistent data stored on both smartphones must match. This requirement confirms the persistent smartphone data stored on two or more smartphones corresponds by viewing the stored data. Evaluation of this requirement depends on the availability of the other smartphone(s).

Mobile Network Operator Consistency

Mobile network operators collect detail records regarding the communication (call history/text messages) that occurred via the network. These records prove to be valuable data to confirm bidirectional communication that transpired between two smartphones and also highlights data previously deleted from the smartphone(s). For smartphone data to be authentic, the available persistent data stored on both smartphones must correspond to the records of the mobile network operator(s). This requirement confirms the matching of the persistent smartphone data with the mobile network operator(s) records, should these records be available for evaluation.

6.4 Smartphone Data Evaluation Model

The collection of requirements identified for each core component equips digital forensic professionals with the necessary instruments to evaluate smartphone data. There is, however, no structure or order to these requirements, which can impact the practical use of the requirements when examining a smartphone. Therefore, the requirements are collected into the smartphone data evaluation model to assist digital forensic professionals. The smartphone data evaluation model provides a step-by-step guide for evaluating and reviewing smartphone data. The model consists of three phases: (i) pre-evaluation phase, (ii) smartphone evaluation phase and (iii) documentation phase. Successful completion of all three phases permits digital forensic professionals to classify the authenticity of the evaluated smartphone data.

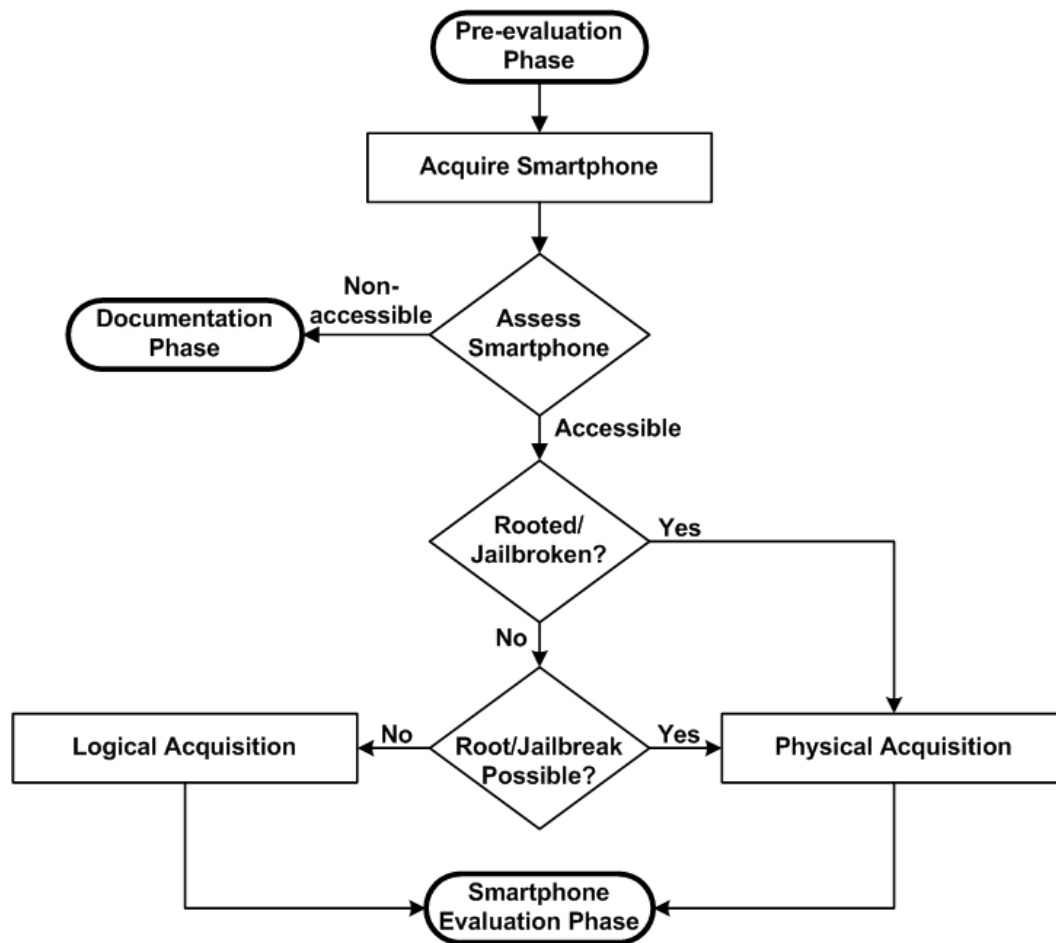


Figure 6.1: Pre-evaluation phase

6.4.1 Pre-evaluation Phase

The first phase of the smartphone data evaluation model requires digital forensic professionals to perform a pre-evaluation of the smartphone. Figure 6.1 presents the steps of the pre-evaluation phase. This phase first acquires and conducts an initial assessment of the smartphone to determine if the content on the smartphone is accessible and not blocked by a screen lock. A locked smartphone can impede the examination and without the required pin, password/passcode or pattern the digital forensic professional may not be able to proceed with the evaluation of the smartphone data. A non-accessible smartphone ultimately concludes the evaluation of the smartphone data and transitions to the

documentation phase. An accessible smartphone allows the digital forensic professional to establish the current state of the smartphone, which is necessary since the state of the smartphone will influence the course of the evaluation.

The easiest and least intrusive manner to determine the current state of a smartphone (rooted/jailbroken) is to view the availability of over-the-air (OTA) updates. A previously rooted/jailbroken smartphone prevents OTA updates to avoid the removal of the root/jailbreak state. Unavailability of OTA updates shows that a smartphone has been rooted/jailbroken but does not indicate the smartphone is currently rooted/jailbroken. A digital forensic professional can verify the current state of a smartphone via a terminal by confirming the availability of superuser/root access. Superuser/root access allows the digital forensic professional to proceed with the physical acquisition of the available smartphone data. Should a smartphone not be currently rooted/jailbroken, a digital forensic professional must investigate the feasibility of rooting/jailbreaking the smartphone using recognised and acceptable methods that will cause minimal alterations to the existing data. Failing to root/jailbreak the smartphone will cause the evaluation to proceed with the logical acquisition of the smartphone data. The digital forensic professional can now continue with the evaluation of the smartphone data, using either logical or physical acquisition to acquire the data.

The outcome of the pre-evaluation phase highlights the accessibility and current state of the examined smartphone, which directs the further evaluation of the smartphone data.

6.4.2 Smartphone Evaluation Phase

The assessment of the smartphone data continues into the smartphone evaluation phase after the pre-evaluation phase concluded and found the smartphone to be accessible for further evaluation. The evaluation of smartphone data depends, however, on the acquisition technique used to retrieve the data from the smartphone. The most popular and widely used acquisition techniques to acquire smartphone data are logical or physical acquisition techniques. Logical acquisition retrieves a bit-by-bit copy of files and directories currently stored on a smartphone (see Section 2.5.2). Generally, this acquisition technique is unable to retrieve all of the data stored on a smartphone and is also further

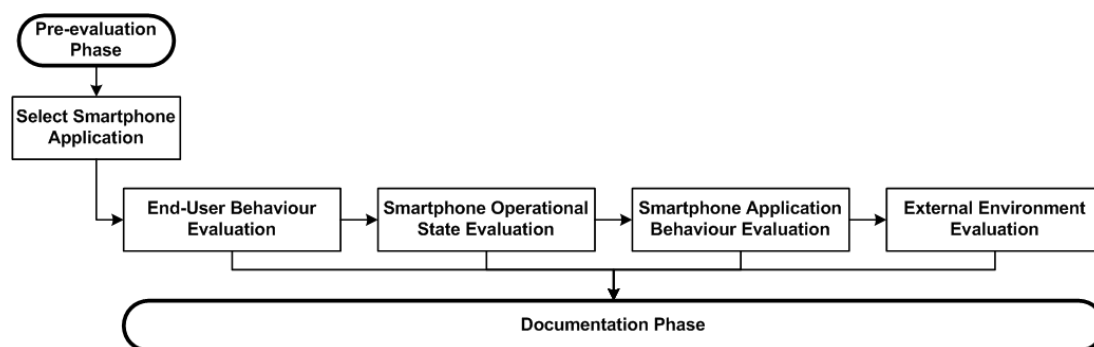


Figure 6.2: Smartphone evaluation phase

influenced by the underlying smartphone operating system version and the current device model. Logical acquisition, therefore, limits the number of requirements available for evaluation during this phase. Physical acquisition, however, allows for the creation of a bit-by-bit raw disk image of the entire physical store of a smartphone (see Section 2.5.2). Selection of physical acquisition during the pre-evaluation phase will allow the digital forensic professional to obtain a copy of all the data stored on the smartphone. Thus, digital forensic professionals can evaluate all of the available requirements.

Figure 6.2 illustrates the individual components of the smartphone evaluation phase. The structure of the smartphone evaluation phase follows the core components identified in Section 6.2. The first step of the smartphone evaluation phase stipulates the digital forensic professional must select a single smartphone application to evaluate. The identified smartphone application must reside on the smartphone. Once selected, the digital forensic professional must interpret and evaluate the requirements of each core component sequentially against the collected smartphone data. Interpretation of these requirements involves three distinct actions: (i) provide input, (ii) make an informed decision or (iii) assess the data. The first action (represented using white rectangular blocks) supports the assessment of smartphone data by providing required input, which can include information such as a list of installed applications or the availability of specific files. The second action (represented using diamond-shaped blocks) requires the digital forensic professional to make an informed decision based on the comparison of specific data sources. The third action (represented using grey rectangular blocks) assesses the smartphone data using the provided input, which produces a ternary result

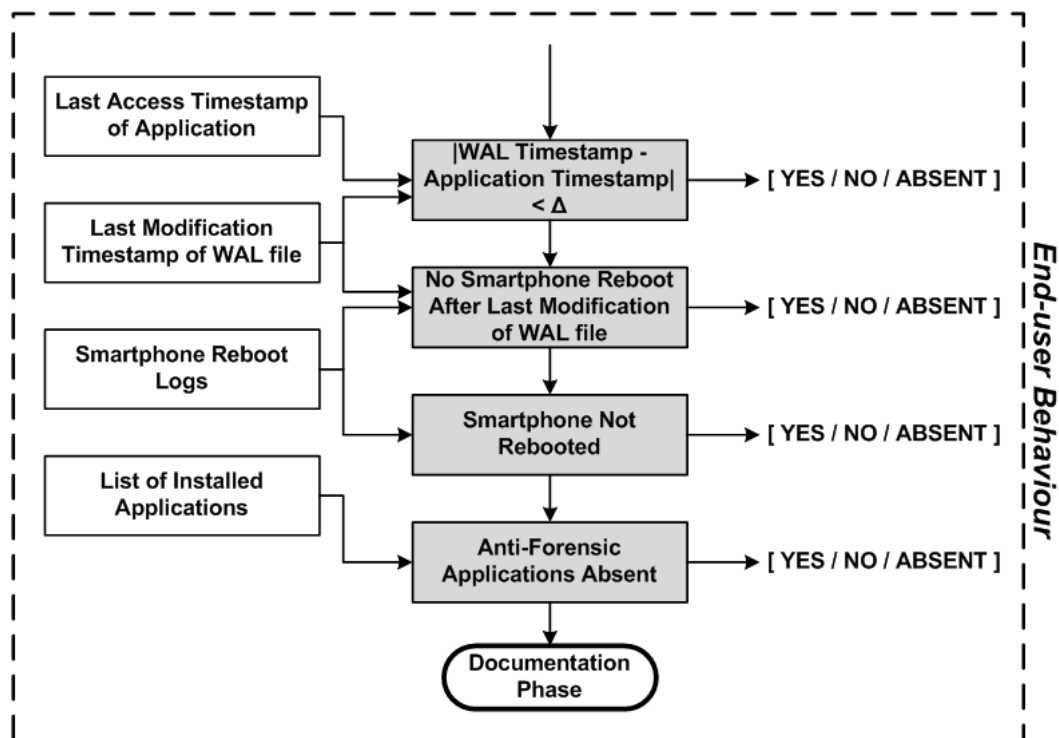


Figure 6.3: End-user behaviour evaluation

[yes/no/absent]. These assessment points evaluate the established requirements, and each requirement has one or more assessment points. The collection of ternary results produced by each evaluated assessment point will allow digital forensic professionals to establish the authenticity of the smartphone data. The remainder of this section focuses on the detailed evaluation of each core component.

Figure 6.3 presents the assessment of the requirements collected in the end-user behaviour core component. The first assessment point confirms the usage of the smartphone application to affect changes to data storage by comparing the last access timestamp of the application against the last modification timestamp of the WAL file. The difference between these two timestamps must be smaller than the pre-determined time frame (Δ). The second assessment point affirms a reboot of the smartphone did not follow after the modification of the WAL file. The third assessment point confirms the smartphone was not rebooted during the period of interest while the final assessment point ensures the absence of anti-forensic applications (any application that supports data hiding,

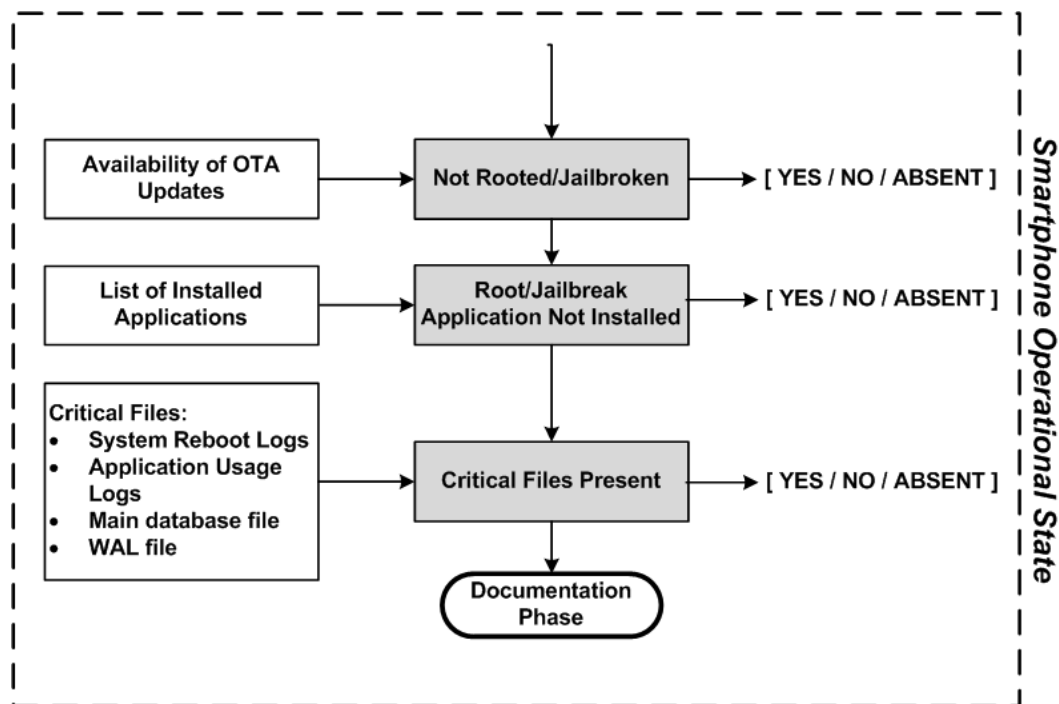


Figure 6.4: Smartphone operational state evaluation

destruction, fabrication or prevents the creation of data).

Figure 6.4 illustrates the assessment of the smartphone operational state requirements. The first assessment point ensures the smartphone is not or has not previously been rooted/jailbroken by reviewing the availability of OTA updates. Available OTA update confirms the standard state of the smartphone. The second assessment point further evaluates the operational state of the smartphone by confirming the absence of any root/jailbreak applications. Such applications are identifiable by included functions that support or manage superuser access, for example, SuperSu (Android) or Cydia (iOS). Finally, the last assessment point affirms all known critical files are present on the smartphone. The critical files that must be present for the evaluation of the assessment point to be true are the log files capturing application usage, system reboot logs, main database file and the associated WAL file.

Figure 6.5 shows the assessment of the requirements representing smartphone application behaviour. The first assessment point confirms the persistent data corresponds to the data viewed via the user interface of the application. Next, the digital forensic

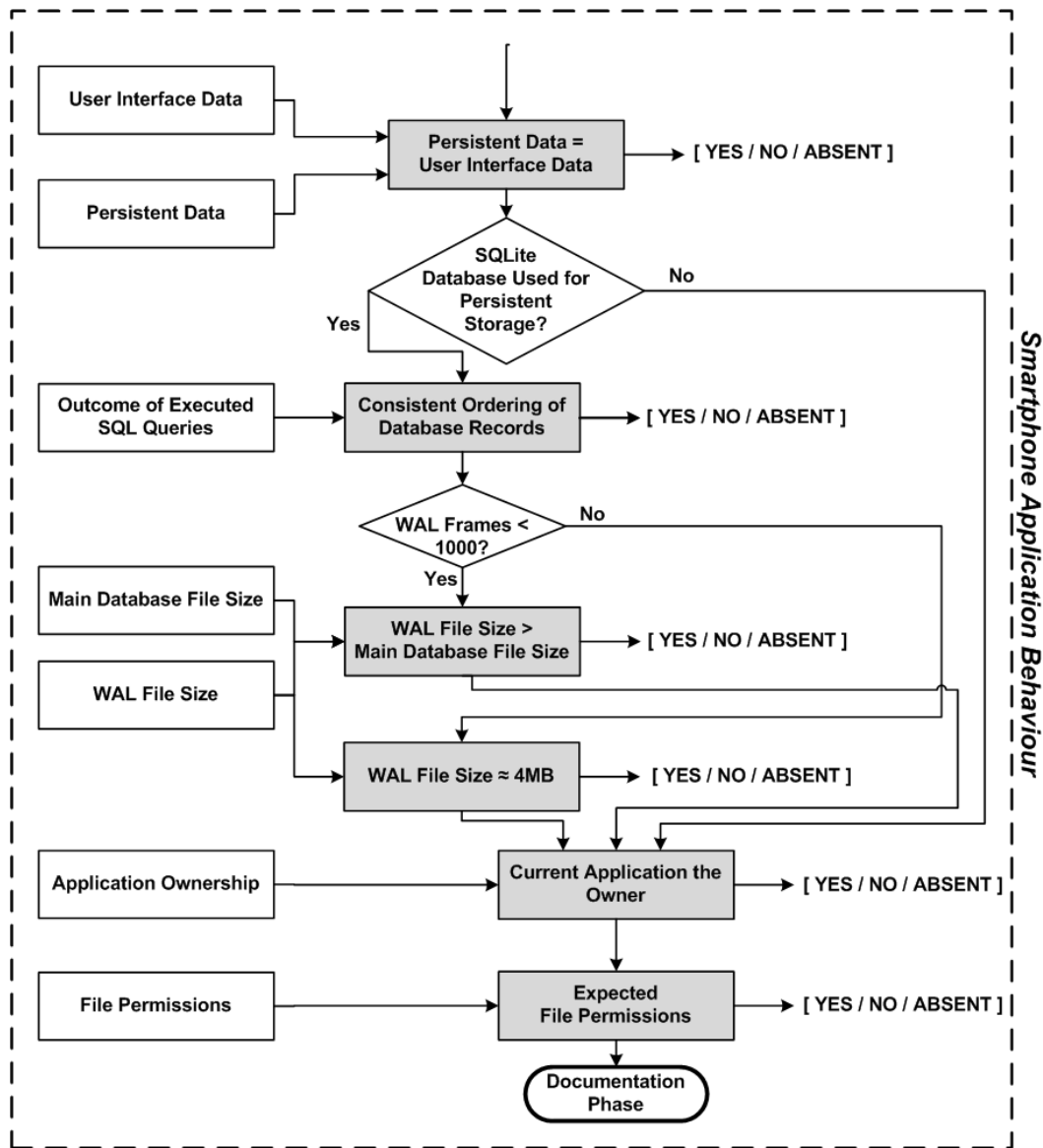


Figure 6.5: Smartphone application behaviour evaluation

professional must make an informed decision regarding the storage structure used for persistent storage. In the absence of SQLite databases used for persistent storage, the evaluation directly proceeds to the final two assessment points, which focus on ownership and file permissions. Usage of an SQLite database for persistent storage necessitates the assessment of the stored records and database file sizes. The second assessment point,

therefore, confirms the consistent ordering of the database records by executing the queries listed in Section 6.3.3 and reviewing the result. The following two assessment points review and compare the size of the WAL file against the main database file. Firstly, the digital forensic professional must make an informed decision regarding the number of frames captured in the WAL file. Calculation of the number of frames relies on the size of each frame (F_s) listed in the WAL header (see Section 4.3.4), WAL file size (W_s), WAL header size (W_h) and frame header size (F_h). Equation 6.1 formally captures the calculation of the total WAL frames (W_F).

$$W_F = (W_s - W_h)/(F_s + F_h) \quad (6.1)$$

Based on the number of frames available in the WAL file, the digital forensic professional can evaluate one of the following two assessment points. A WAL file with fewer than 1000 frames has yet to perform a checkpoint and, therefore, is expected to have a file size larger than the size of the main database file. Based on multiple viewings, a WAL file containing 1000 or more frames is expected to have a file size of approximately 4MB. Adequate evaluation of the assessment point requires the examined WAL file size to fall within one standard deviation of the expected size of 4MB. Following the assessment of the SQLite database, it is necessary to evaluate the final two assessment points. The second last assessment point confirms the application is the owner of the files storing the persistent smartphone data and the final assessment point ensures the file permissions are unchanged (only the individual and group owners are assigned read/write permissions).

Figure 6.6 presents the assessment of the final requirements collected in the external environment core component. Before proceeding with the evaluation, the digital forensic professional must make an informed decision regarding the form of communication supported by the application. Bi-directional communication will cause comparable data to reside on the corresponding smartphone(s). The availability of the corresponding smartphone(s) directly depends on the ongoing case and whether the smartphone(s) were acquired. Should the smartphone(s) be available for evaluation, the first assessment point confirms the persistent data corresponds between the smartphones. Such comparison involves collections of smartphone data that remains fixed during deliver

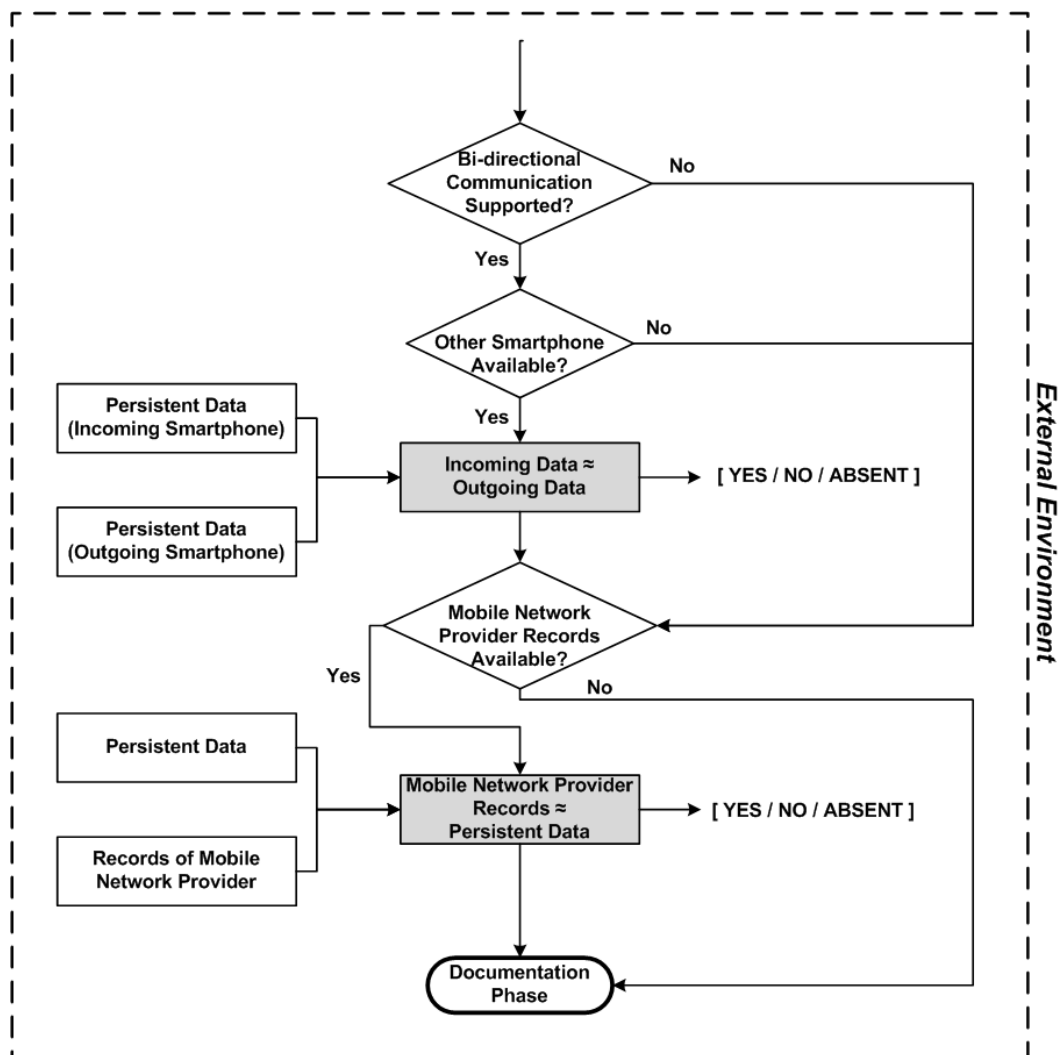


Figure 6.6: External environment evaluation

between smartphones. Fixed smartphone data include textual data, contact numbers, photographs or documents. Dynamic sources of smartphone data, such as timestamps, are excluded from the comparison since various factors (delivery failures or time settings) can influence consistency. Unavailability of the smartphone(s) or the reliance on the unidirectional communication of the application will cause the evaluation to proceed to the final assessment point. Before performing the assessment, the digital forensic professional must first make an informed decision regarding the availability of the records collected by the mobile network provider. Should these records be available for evaluation, it is

necessary to confirm that the collected records correspond to the persistent smartphone data. Similar to the comparison of persistent data between smartphones, the comparison of mobile network provider records also only focuses on fixed data sources, such as contact numbers. Inability to review the mobile network provider records will conclude the evaluation of the external environment requirements.

Should the evaluation require the assessment of data from more than one smartphone application, the digital forensic professional must repeat this phase for each smartphone application. Once the digital forensic professional completed the interpretation of the requirements and collected the necessary results, the evaluation proceeds to the documentation phase to conclude the smartphone assessment.

6.4.3 Documentation Phase

The final phase of the smartphone data evaluation model is the documentation phase, which collects and aggregates all the results produced by the reviewed assessment points during the smartphone evaluation phase. The collected results captured while evaluating the smartphone data permit digital forensic professionals to make informed and well-weighted decisions regarding the evaluated data. A more extensive collection of positive ([yes]) results indicates with higher certainty that the evaluated smartphone data are authentic. The increasing presence of negative ([no]) results indicate a lower certainty of the authenticity of the smartphone data and shows the data to be unlikely authentic. Further evaluation of the collected results is possible by submitting the results to a classification model (see Chapter 7), which will provide the digital forensic professional with a more explicit indication of the authenticity of the smartphone data. The outcome of the classification model is an authenticity classification associated with a completeness score that reflects the overall authenticity of the evaluated smartphone data. This phase, thus, concludes the evaluation of the smartphone and the related smartphone data.

6.5 Model Qualities and Characteristics

The purpose of the smartphone data evaluation model is to allow digital forensic professionals to assess the authenticity of smartphone data easily. The model consists of

three sequential phases that determine the accessibility of the smartphone, evaluates the smartphone data and documents the final result. Most important, is the smartphone evaluation phase. Therefore, the evaluation phase collects and captures the diverse collection of requirements into a simplistic and easy to follow structure. The structure consists of four individual elements representing the core components responsible for the creation of authentic smartphone data. Depending on the acquired data, digital forensic professionals can either evaluate all or a subset of the requirements per smartphone application. The flexible structure of the smartphone data evaluation model easily supports different collections of smartphone data. The simplified flow structure provides a step-by-step guide for evaluating the smartphone data of an application. This structured approach quickens the assessment of smartphone data and saves digital forensic professionals valuable time during examinations. The outcome of the smartphone data evaluation model is a collection of results that describe the authenticity of the data.

The result produced by the model allows digital forensic professionals to make informed decisions regarding the inclusion or exclusion of smartphone data. Digital forensic professionals can eliminate unreliable data from being submitted as potential digital evidence and only use reliable data to formulate and draw accurate conclusions. The produced results cannot, however, allow the digital forensic professional to identify fabricated or manipulated smartphone data. The final decision regarding the inclusion or exclusion of the smartphone data based on the data's evaluated authenticity remains with the digital forensic professional.

Key characteristics of the smartphone data evaluation model include simplicity, comprehensiveness and transparency. The easy to understand and to the point model only evaluates a single application at a time. The model only assesses the authenticity of smartphone data and does not attempt to evaluate other concepts such as accuracy or reliability. By only assessing a single concept of an individual application ensures the simplicity of the model. The comprehensive model supports data collected from either Android or iOS smartphones and can assist with the evaluation of both original or manipulated smartphone data. The deployed acquisition technique (physical or logical) also does not influence the functional evaluation capability of the model. The transparency of the model will guide digital forensic professionals to appropriate results. The model

thus provides digital forensic professionals with the necessary techniques and guidance for establishing the authenticity of smartphone data.

6.6 Summary

The purpose of this chapter was to define smartphone data formally and from the definition identify a collection of requirements that can evaluate the authenticity of smartphone data. The requirements permit digital forensic professionals to review the four components of authentic smartphone data, namely (i) end-user behaviour, (ii) smartphone operational state, (iii) smartphone application behaviour and (iv) external environment. Confirming the consistency and reliability of these components establish a level of certainty that the smartphone data is authentic. To further assist digital forensic professionals with the evaluation of smartphone data, these requirements form part of the smartphone data evaluation model. Using the model allows digital forensic professionals to evaluate the authenticity of smartphone data and permits the elimination of unreliable data. To further support the evaluation of smartphone data, the presented requirements and smartphone data evaluation model forms the foundation of a classification model. The smartphone data classification model, which is the focus of the following chapter, provides digital forensic professionals with a calculated outcome regarding the authenticity of smartphone data.

Chapter 7

Classification of Evaluated Smartphone Data

The focus and intended purpose of the previous chapter was to formally describe authentic smartphone data and establish a collection of requirements that support the evaluation of the authenticity of such data. These requirements were used to construct the smartphone data evaluation model, which provides digital forensic professionals with a structured approach for evaluating the authenticity of smartphone data. Currently, the smartphone data evaluation model only stipulates how smartphone data must be evaluated and not what the outcome of the evaluation is. To assist digital forensic professionals with the proper utilisation of the smartphone data as a form of digital evidence, a classification model that categorises the authenticity of the evaluated smartphone data is required.

This chapter introduces a new classification model for smartphone data, constructed using the stipulated requirements and smartphone data evaluation model. The classification model classifies the evaluated smartphone data using an ordered pair of values. The first value represents the grade of the authenticity ((i) High, (ii) Moderate, (iii) Low or (iv) Unsatisfactory), calculated using a mathematical equation built from the requirements identified in Chapter 6. The second value describes the completeness (high or low) of the evaluation, which allows digital forensic professionals to present the authenticity of the evaluated smartphone data with certain confidence. Section 7.1 describes the

various models available to classify digital evidence. These models provide the necessary guidance to establish the classification model for smartphone data, which is presented in Section 7.2. Section 7.3 introduces the classification model prototype, called the Smartphone Application Data Authenticity Classifier (SADAC). Section 7.4 concludes the chapter.

7.1 Classification of Digital Evidence

With the increasing maturity of digital forensics as a discipline, digital forensic professionals identified a need to assign a level of certainty to their formulated conclusions. Currently, digital forensics lacks formal mathematics or statistics to associate a level of certainty to evaluated digital evidence (Casey 2011). The lack of a formal model is mostly due to the complexity and multiplicity of evolving digital systems. Generally, digital forensic professionals turn to an informal model to assign certainty, which classifies digital evidence according to various degrees of likelihood: (i) almost definitely, (ii) most probably, (iii) probably, (iv) very possible and (v) possibly (Casey 2011). Digital forensic professionals are, however, influenced by their current knowledge and experience, which may lead to different uses of the informal classes. Such inconsistencies can confuse and emphasise the need for a more formal and consistent methodology to classify the certainty of digital evidence. Casey (2011), therefore, proposed a certainty scale that allows digital forensic professionals to assign a level of certainty to a given piece of digital evidence. The certainty scale, visible in Table 7.1, offers various levels that digital forensic professionals can use to assess the probative value of the digital evidence. Also, the certainty scale introduces a form of consistency that allows digital forensic professionals to easily comprehend, understand and share the certainty assigned to pieces of digital evidence among one another.

The proposed certainty scale of Casey (2011) also permits digital forensic professionals to assign a level of certainty to smartphone data. The primary purpose of the certainty scale is not, however, to classify authenticity. Therefore, the following section investigates the development of a classification model for smartphone data.

Table 7.1: Proposed Scale for Classifying Levels of Certainty (Casey 2011)

Certainty Level	Description	Commensurate Qualification
C0	Evidence contradicts known facts.	Erroneous/Incorrect
C1	Evidence is highly questionable.	Highly Uncertain
C2	Only one source of evidence is not protected against tampering.	Somewhat Uncertain
C3	The source(s) of evidence is more difficult to tamper with but there is not enough evidence to support a firm conclusion, or there are unexplained inconsistencies in the available evidence.	Possible
C4	(a) Evidence is protected against tampering or (b) evidence is not protected against tampering but multiple, independent sources of evidence agree.	Probable
C5	Agreement of evidence from multiple, independent sources that are protected against tampering. However, small uncertainties exist.	Almost Certain
C6	The evidence is tamperproof or has a high statistical confidence.	Certain

7.2 Classification Model for Smartphone Data

Similar to the consistency required when assigning levels of certainty to conclusions drawn from the digital evidence, such consistency is also necessary when classifying the authenticity of smartphone data. Currently, the certainty scale of Casey (2011) only focuses on the classification of certainty, which emphasises the need for a methodological approach to classify authenticity. Collectively, the requirements and smartphone data evaluation model presented in Chapter 6, along with the existing classification mod-

els, provide the necessary foundation to establish a classification model for smartphone data. This classification model focuses exclusively on classifying the authenticity of application-generated smartphone data residing on the smartphone. The output of the classification model is an authenticity classification, which is an ordered pair of values describing the grade of authenticity and the completeness of the evaluation. The following sections describe the calculation of these values, as well as the final representation of the authenticity classification.

7.2.1 Categorisation of the Requirements

It is necessary to formulate a detailed mathematical equation that produces dependable results to consistently classify the authenticity of evaluated smartphone data. The basis of this mathematical equation is the requirements and smartphone data evaluation model presented in Chapter 6. In total, eleven requirements were identified and categorised according to four core components: (i) end-user behaviour, (ii) smartphone operational state, (iii) smartphone application behaviour and (iv) external environment. These components are used to measure the authenticity of smartphone data. Evaluation of each requirement occurs using one or more assessment points, reflected in the smartphone evaluation phase of the smartphone data evaluation model (see Section 6.4.2). Currently, the interpretation of smartphone data occurs using fifteen assessment points across the four core components. The outcome of each assessment point is a ternary result [yes/no/absent]. A positive result [yes] confirms the requirement is met while a negative result [no] indicates the evaluated data contradicts the requirement. Should the data be unavailable or insufficient to evaluate the assessment point, an absent [absent] result is produced.

The impact of each result produced by the assessment points is, however, not equal since each assessment point evaluates different aspects of the authenticity of smartphone data. The categorisation of the assessment points into appropriate classes with a distinct focus will provide a more accurate evaluation of the authenticity. Stemming from the definition of authenticity (see Section 6.1), which describes data that preserves the same identify it had when first created and can be proven to have maintained its integrity over time, are two distinct classes. The first class (A) contains assessment points that

confirm no opportunity existed for the smartphone data to change. The second class (*B*) collects assessment points that evaluate the consistency of the various components responsible for creating the smartphone data, as well as the data itself. The purpose of class *B* assessment points is to evaluate the smartphone, the smartphone application and the application's associated data. Therefore, it is necessary to further categorise class *B* assessment points into the following subclasses:

- **Subclass 1:** assessment points only evaluate the application's data.
- **Subclass 2:** assessment points evaluate the behaviour of the application and the file structure used to store the data.
- **Subclass 3:** assessment points evaluate the state of the smartphone.

Tables 7.2 to 7.5 categorise the available assessment points according to the classes established and the core component to which the assessment point belongs. The categorisation of the assessment points according to classes A and B allows for a weighted calculation of the authenticity score.

Table 7.2: Categorisation of assessment points (End-user Behaviour)

Requirement Addressed	Assessment Point	Class
Consistent Application Usage	$ \text{WAL Timestamp} - \text{Application Timestamp} < \Delta$	B.2
Consistent Application Usage	No smartphone reboot after last modification of WAL file	B.2
Rarely Rebooted Smartphone	Smartphone not rebooted	B.3
Anti-Forensics Applications Eliminated	Anti-forensic applications absent	A

Table 7.3: Categorisation of assessment points (Smartphone Operational State)

Requirement Addressed	Assessment Point	Class
Standard Smartphone State	Not rooted/jailbroken	A
Standard Smartphone State	Root/Jailbreak application not installed	B.3
Critical Files Present	Critical files present	B.3

Table 7.4: Categorisation of assessment points (Smartphone Application Behaviour)

Requirement Addressed	Assessment Point	Class
Corresponding Data (Internally)	Persistent data = User interface data	B.1
Internal Database Consistency	The consistent ordering of database records	B.1
File System Consistency	The current application is the owner	B.2
File System Consistency	File permissions unchanged	B.2
Database File Consistency	WAL file size > Main database file size	B.2
Database File Consistency	WAL file size approximately 4MB	B.2

7.2.2 Authenticity Score Calculation

Calculation of the authenticity score follows a weighted approach since the outcome of each assessment point impacts the authenticity of the smartphone data differently.

Table 7.5: Categorisation of assessment points (External Environment)

Requirement Addressed	Assessment Point	Class
Corresponding Data (Externally)	Incoming data \approx Outgoing data	B.1
Mobile Network Operator Consistency	Mobile network provider records \approx persistent data	B.1

The weighted calculation of the authenticity score relies, therefore, on the assignment of appropriate weights to each class. The weight assigned to each class reflects the impact the evaluated assessment point will have on the final authenticity score.

Section 7.2.1 identified two distinct classes for assessment point categorisation. Since class *A* contains approximately 15% of the available assessment points (see Tables 7.2 to 7.5), a weight of 0.15 is assigned to the class. The weight assigned to class *B* is the remainder, which is 0.85 and represents a larger collection of assessment points. However, the assigned weight of class *B* must be further subdivided in order to assign an appropriate weight for each individual subclass. The focus of subclass 1 assessment points is strictly aimed at the evaluation of the data of the smartphone application and the produced results will, therefore, have a significant influence the outcome of the authenticity score. The importance of the results produced by these assessment points necessitates a larger weight to be assigned to subclass 1. The mean of the weight originally attributed to class *B* is assigned as the weight of subclass 1. Assessment points belonging to subclass 2 focus on evaluating the behaviour of the smartphone application but excludes the application's data. The results produced by evaluating the behaviour of the smartphone application has a lesser influence on the calculated authenticity than the evaluation of the data. Assigned as the weight of subclass 2 is the mean of two-thirds of the original weight attributed to class *B*. Finally, assessment points of subclass 3 focus only on the smartphone. Since these assessment points do not directly address the smartphone application or related data, the produced results will have a minimal impact on the authenticity score. Therefore, subclass 3 receives a weight that is the mean of

Table 7.6: Weight assignments for the classes

Class A	Class B		
	Subclass 1	Subclass 2	Subclass 3
0.15	0.425	0.28	0.14

one-third of the original class B weight. Table 7.6 presents the numerical value of the weight for each class.

Calculation of the score for class A occurs using equation 7.1. The assessment points evaluated per class produce a collection of positive or negative results. However, the acquisition technique followed to acquire the data (see Section 2.5.2) can impact the ability to assess all available assessment points. Therefore, the collection of positive (pos_c) results are divided by the number of assessment points (n_c) evaluated per class (c), which is then weighed using the assigned weight (w_c) as shown in Table 7.6. The score for class B is calculated using equation 7.2 and is the sum of the individual scores of the subclasses. Equation 7.3 calculates the final authenticity score (A_s), which is the sum of the scores calculated for classes A and B .

$$S_A = w_c \frac{pos_c}{n_c} \quad (7.1)$$

$$S_B = \sum_{c=1}^3 w_c \frac{pos_c}{n_c} \quad (7.2)$$

$$A_s = \sum_{c=A}^B S_c \quad (7.3)$$

Currently, the presented authenticity score is merely a percentage and lacks context or description. It is necessary to further describe the authenticity score by assigning an appropriate grade.

7.2.3 Authentic Grading Scale for Smartphone Data

The authenticity score, produced by the mathematical equations introduced during the previous section, presents the authenticity of the evaluated smartphone data as a per-

centage. The percentage, alone, is inadequate and requires further description and categorisation to better reflect the classified authenticity of the smartphone data. The categorisation of the authenticity score requires additional interpretation of the evaluated assessment points and all possible outcomes. Both the number of assessment points evaluated and the possible outcomes factor significantly into the categorisation of the authenticity score. Therefore, it is first necessary to confirm the assessment points evaluated and calculate all possible outcomes relating to the evaluation of these assessment points. The result is a collection of outcomes that follows a bell shaped curve or normal distribution. The normal distribution presents two clusters of potential outcomes. The first cluster (below the mean of the normal distribution) present the outcomes of evaluated assessment points that mostly produced negative results. It is possible to further group these outcomes as follows:

- Outcomes of evaluated assessment points is negative (authenticity unsatisfactory).
- Outcomes of assessment points produced more negative results that outweighed the positive results (authenticity low).

The second cluster of outcomes (above the mean of the normal distribution) represent the opposite where the outcomes of the evaluated assessment points mostly produced positive results. It is also possible to further group these outcomes as follows:

- Outcomes of assessment points produced more positive results that outweighed the negative results (authenticity moderate).
- Outcomes of evaluated assessment points is positive (authenticity high).

The classification model, therefore, consists of four distinct grades of authenticity, presented in Table 7.7. These grades of authenticity can be compared with values in the certainty scale (see Table 7.1) as follows: High — C5, Moderate — C4, Low — C2 and Unsatisfactory — C1. The comparison confirms the classification model complements and extends the completeness of the evaluation provided by the certainty scale.

Assigning a grade to the final authenticity score requires the division of the normal distribution representing all possible outcomes into quartiles. The lower quartile distinguishes between the unsatisfactory and low authentic grading, the middle quartile

Table 7.7: Authentic grading scale for smartphone data

Grade	Description
Unsatisfactory	Evaluated smartphone data fails to meet most of the requirements.
Low	Evaluated smartphone data meets some of the requirements.
Moderate	Evaluated smartphone data meets most of the requirements, especially requirements captured in subclasses 2 and 3.
High	Evaluated smartphone data meets most of the requirements, especially requirements captured in subclasses 1 and 2.

separates the low and moderate authentic grading and the upper quartile distinguishes the high authentic grading from the moderate authentic grading.

The quartiles make it possible to construct the authentic grading scale that provides context and better describes the classified authenticity. The quartile values provide the boundaries between the distinct grades of authenticity. The authenticity score is then plotted on the scale to determine the authentic grading of the evaluated smartphone data. This constant and formal measurement of smartphone data ensures that digital forensic professionals can conclusively establish the authenticity of smartphone data and also easily comprehend the grade of authenticity among one another.

7.2.4 Completeness

The calculation of the authenticity score and the construction of the authentic grading scale directly depends on the collection of assessment points evaluated. Influencing the availability of these assessment points is the acquisition technique used to acquire the smartphone data. The completeness score (C_S), established using equation 7.4, allows digital forensic professionals to present the authenticity score of the evaluated

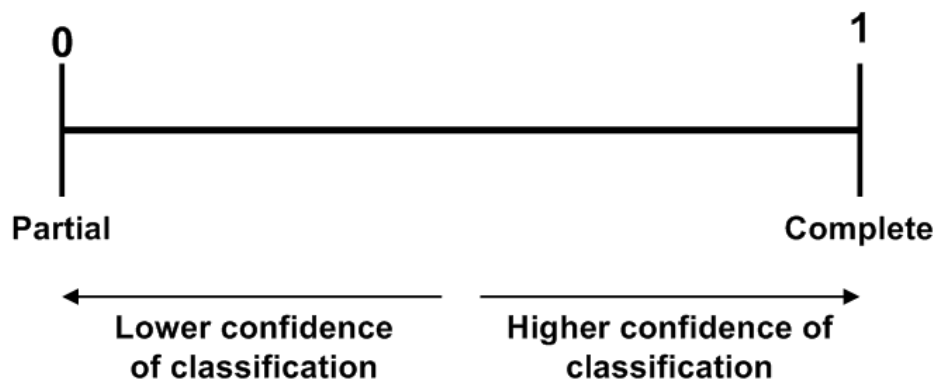


Figure 7.1: Completeness scale

smartphone data with certain confidence. Therefore, the calculated completeness score compliments the authenticity grading.

$$C_s = \sum_{i=1}^4 \left(\frac{a_i}{t_i} \right) (0.25) \quad (7.4)$$

For each core component, the evaluated assessment points (a_i) are counted and divided by the total assessment points (t_i) available for that core component. The final completeness score is a weighted score calculated using 25% weight measurement per core component. The weighted score ensures equal importance among the core components. The final completeness score is plotted on the scale shown in Figure 7.1. A more substantial collection of assessment points evaluated presents a more thorough and complete classification of the authenticity of the smartphone data. The availability of fewer assessment points highlights a more partial evaluation of the smartphone data, lowering the confidence associated with the classification of the authenticity of the smartphone data.

7.2.5 Authenticity Classification

The calculated authenticity (A_S) and completeness (C_S) scores are the key results produced by the described classification model. These scores form an ordered pair of values that represents the authenticity classification (A_C) of the evaluated smartphone data (see equation 7.5).

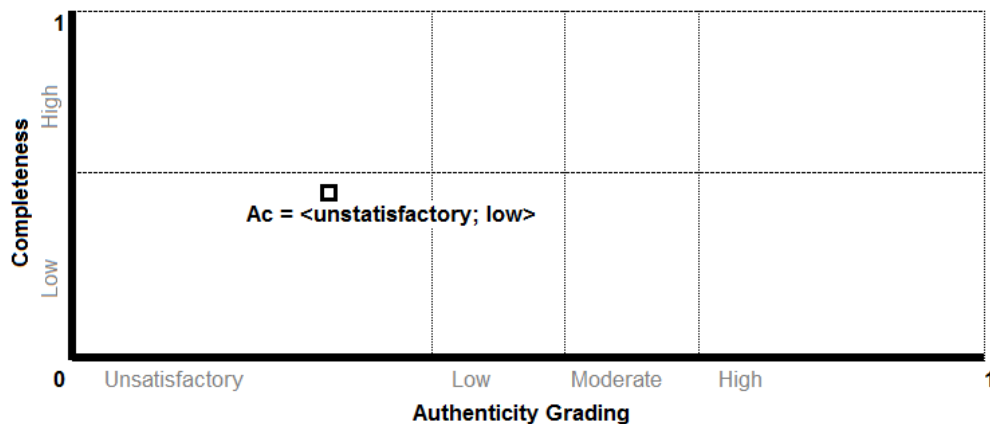


Figure 7.2: Authenticity Classification Graph

$$A_C = \langle A_S; C_S \rangle \quad (7.5)$$

Visual representation of the final authenticity classification occurs using the authenticity classification graph shown in Figure 7.2. The authenticity classification graph follows a two-dimensional structure to depict both the grade of authenticity and the completeness of the evaluation. The x-axis represents the authentic grading scale, and the vertical lines divide the available space into four quartiles to portray the four grades of authenticity. The y-axis represents the completeness scale, and the single horizontal line distinguishes between higher and lower confidence of classification. Finally, the drawn square illustrates and confirms the authenticity classification of the evaluated smartphone data.

The presented authenticity classification of the smartphone data provides digital forensic professionals with necessary insight into the authenticity of the data. Although digital forensic professionals can manually complete the provided mathematical equations, human error can severely impact the classification of the authenticity. The following section focuses exclusively on the development of the Smartphone Application Data Authenticity Classifier, which automates the authenticity classification of smartphone data and eliminates potential human error.

7.3 Smartphone Application Data Authenticity Classifier (SADAC)

SADAC is a proof of concept digital forensic tool that automates the mathematical equations provided by the classification model. The primary objective of SADAC is to quicken, simplify and ensure the accuracy of the calculated authenticity classification. SADAC also supports the visualisation of the calculated authenticity classification according to the two-dimensional graph shown in Figure 7.2. The remainder of this section focuses on the functional requirements, architectural design, technical platform specification and interface layout of SADAC.

7.3.1 Functional Requirements

The purpose of the SADAC is to accurately calculate and present the authenticity classification of the evaluated smartphone data. Therefore, the SADAC tool must support the evaluation of all assessment points and collect the outcome of each assessment point's ternary result [yes/no/absent]. Collectively, these results are used to calculate the authenticity and completeness scores using equations 7.1 and 7.2 respectively. The final authenticity classification of the evaluated smartphone data is then established using these values and represented as an ordered pair of values using equation 7.3. Also, the SADAC tool must utilise the authenticity classification graph to depict the final authenticity classification.

The SADAC tool must provide digital forensic professionals with the option to clear current results to allow for the re-use of the application. Such an option must return the SADAC tool to the original state and allow digital forensic professionals to re-evaluate the smartphone data of the selected smartphone application.

7.3.2 Architectural Design

The architectural design of the SADAC tool follows the input-process-output (IPO) design pattern. Figure 7.3 illustrates the architectural design of the SADAC tool according to the IPO pattern. The SADAC tool receives from a digital forensic professional as input

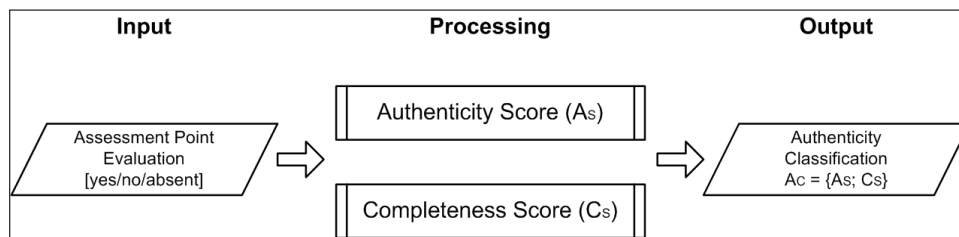


Figure 7.3: SADAC architectural design

the outcome of all the evaluated assessments points. The input is a collection of positive results, as well as the number of assessment points evaluated due to the availability of the smartphone data. The process component uses the provided input to calculate the authenticity and completeness scores. As output, the SADAC tool produces the authenticity classification of the evaluated smartphone data and visualise the classification using the authenticity classification graph. Following the IPO design pattern for the SADAC tool implementation ensures accurate results are produced quickly and allows for effective operation by all digital forensic professionals.

7.3.3 Technical Platform Specification

The SADAC tool was developed using Java, a general-purpose programming language, and the Eclipse (version 4.7.2) development environment. Java 1.8 was selected for the primary reason of supporting platform independence. The final version of the SADAC tool was tested on Dell Latitude E5570 machine running Windows 7 (64-bit) operating system and supporting an Intel Core i7 2.70 GHz processor with 16 GB DDR3 RAM.

7.3.4 Interface Layout

Figure 7.4 captures the structural ordering and layout of the key components provided by the SADAC user interface. The main window of the interface includes three menu items namely (i) “About”, (ii) “Help” and (iii) “Exit”. The “About” menu offers a brief description of the SADAC tool, along with the name of the lead developer and current version. The “Help” menu offers guidance to digital forensic professionals regarding the use and operation of the SADAC tool. The final menu item, “Exit”, closes the tool.

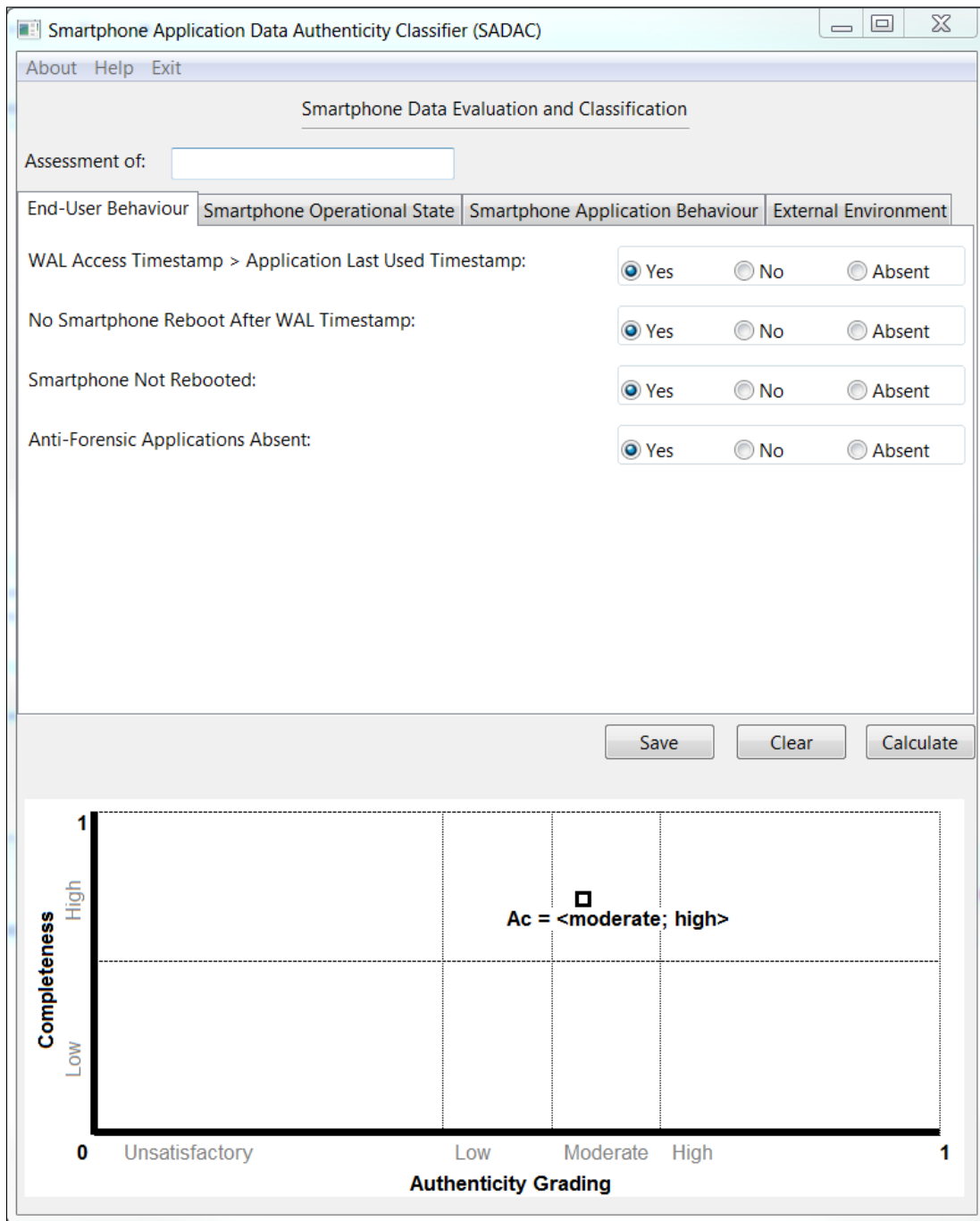


Figure 7.4: The SADAC tool interface layout

Immediately following the menu is the option to provide the name of the smartphone application selected for evaluation, which allows digital forensic professionals to assign a unique name to each assessment.

The central viewing area of the SADAC tool consists of functional tabs, three interactive buttons and a canvas to draw the authenticity classification graph. Each tab represents a core component of authentic smartphone data (see Section 6.2) and captures collectively the assessment points belonging to each component. Representation of the ternary result [yes/no/absent] for each assessment point is achieved using radio buttons, which enforces the selection of a single option only. Required decisions, based on comparing specific data sources, that also involves the availability of assessment points are also provided using a [yes/no] radio button option. The radio buttons allow for more interactive evaluation of the smartphone data.

The “Calculate” button collects the selection of all assessment points and computes the authenticity classification using the provided equations. The final authenticity classification is visually illustrated using the authenticity classification graph in the canvas panel below the buttons. The “Clear” button provides the option to clear the current selection of assessment points and reset the SADAC tool to the original state, ready to accept new input. The “Save” button creates a Portable Network Graphic (PNG) image of the authenticity graph, which digital forensic professional can use for comparison or presentation purposes.

The simple and minimalistic design of the user interface for the SADAC tool allows for easy comprehension and effective operation of the available functions. It is only necessary for digital forensic professionals to provide the SADAC tool with the required input as the remainder of the process is entirely automated. It is, however, important to note that the SADAC tool is not developed to replace existing mobile forensic tools, but instead compliment these tools by overcoming inherent limitations.

7.4 Summary

The identification of key requirements that authentic smartphone data must adhere to, along with the construction of the smartphone data evaluation model, provided the

foundation to establish the classification model. This classification model serves to classify evaluated smartphone data of the selected smartphone application. As output, the model produces the authenticity classification that is an ordered pair of values. These values describe both the authenticity of the smartphone data using an appropriate grade and the completeness of the classification. The SADAC tool was developed to organise the evaluation of the smartphone data and automates the authenticity classification of the evaluated data. The assignment of the authenticity classification effectively concludes the evaluation of the smartphone data and permits digital forensic professionals to make informed decisions regarding the use of the smartphone data as digital evidence. The following chapter focuses on the validating the models by evaluating manipulated smartphone data and confirming the practical use of the models to classify smartphone data.

Chapter 8

Smartphone Data Evaluation

Accurate and thorough evaluation of authentic smartphone data relies on the newly designed smartphone data evaluation and classification models. The smartphone data evaluation model captures the assessment structure required to evaluate the data of a single smartphone application while the classification model measures the authenticity of the evaluated smartphone data. The previous chapter introduced the SADAC tool to assist digital forensic professionals with the evaluation of smartphone data and eliminate potential calculation errors. The purpose of the SADAC tool is to consolidate and simplify the evaluation, as well as the classification processes. The purpose of Chapter 8 is to assess both the effectiveness and efficiency of the SADAC tool, which encapsulates the smartphone data evaluation and classification models, to evaluate smartphone data. Before assessing the developed SADAC tool and related models, it is necessary first to determine how and what manipulative changes apply to smartphone data. The steps followed to manipulate smartphone data leads to the construction of a generic process to generalise the manipulation of data across both the Android and iOS platforms. The manipulation of smartphone data is primarily an attack on the authenticity of the data and best describe using an attack tree. The attack tree presents various attack scenarios that can be used to manipulate smartphone data. The chapter concludes by applying a collection of these attack scenarios at a theoretical level and the effects evaluated using the SADAC tool.

Section 8.1 describes the motivations for and the techniques to manipulate smart-

phone data. The purpose of Section 8.2 is to investigate the manipulation of smartphone data on both the Android and iOS platforms, which leads to the formulation of a generic process for smartphone data manipulation. The generic process supports the construction of an attack tree model for smartphone data, which is discussed in Section 8.3. In Section 8.4 various attack scenarios derived from the attack tree model are applied at a theoretical level to manipulate smartphone data, which is then evaluated using the SADAC tool. Section 8.5 concludes the chapter.

8.1 Manipulation of Smartphone Data

The extensive and diverse use of smartphones by end-users to accomplish daily activities provide these devices with rich collections of smartphone data. Generally, the available smartphone data describes events, such as the sending of a text message or browsing a web page, which occurred on the smartphone. Such smartphone data, for example, contacts, text messages, call lists, browsing history or e-mails, becomes valuable since the data provides a clear snapshot of end-user events, as well as the chronological ordering of the events. The exact events recorded by a smartphone depend, however, on several internal and external factors, such as smartphone settings, operation by the end-user and execution of installed applications. Regardless of the potential influence of such factors, the present and available smartphone data can still offer insight and guidance to digital forensic professionals.

The value of smartphone data as a form of digital evidence has, however, led to heightening awareness among end-users regarding the presence of data on smartphones. The presence of specific data sources can be a cause for concern (Tsavli et al. 2015), which drives end-users to apply manipulative techniques to the data to eliminate or remove any potential value. The motivation for manipulating smartphone data is two-fold. Firstly, benign end-users can deploy specific techniques to manipulate smartphone data deemed private or sensitive and minimise the presence 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 are commonly referred to as anti-forensics (see Section 3.3.3) and allow end-

users to manipulate smartphone data.

Manipulation of smartphone data occurs by modifying, fabricating or deleting the data. Modification of the smartphone data refers to tampering or altering existing smartphone data. Modification applies changes to the existing data that reflects misleading information. Fabrication describes the creation of new but false smartphone data. End-users insert fabricated or counterfeited data to represent actual data but also provide misleading information. Finally, deletion of smartphone data removes the presence of the data and impacts the availability of digital evidence. Applying one or more of these manipulation techniques influence the authenticity of the smartphone data that ultimately misleads digital forensic professionals during examinations.

8.2 Generic Process for Smartphone Data Manipulation

The manipulation of smartphone data, as described in the previous section, is performed for different reasons by applying various techniques. Application of these techniques requires the execution of several steps to effectively manipulate the smartphone data. Collectively, these steps can form a generic process for smartphone data manipulation on both the Android and iOS platforms. Establishing the generic process relies on performed exploratory experiments involving both the Android and iOS default messaging application. The purpose of these experiments is to obtain access to the persistent data of the default messaging applications and attempt the manipulation (modify, fabricate or delete) of the data. While performing the exploratory experiments, the steps followed to access and manipulate the smartphone data are carefully documented. From the observations, similarities are identified and collected into a generic process. This generic process stipulates the steps to follow to manipulate smartphone data on either the Android or iOS platform.

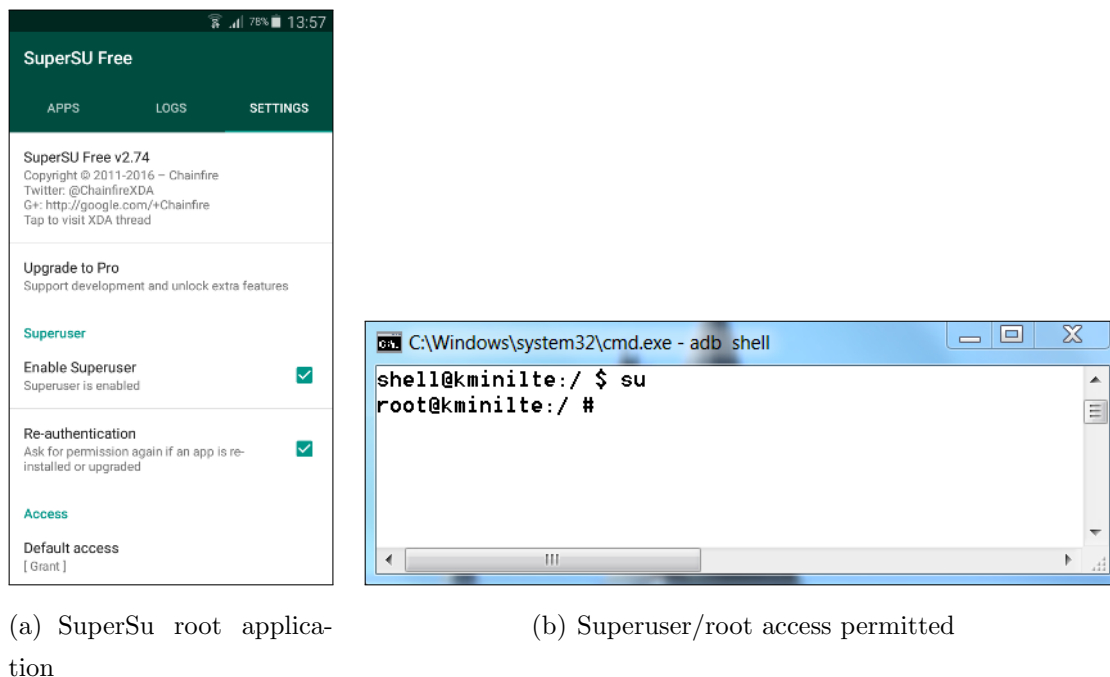
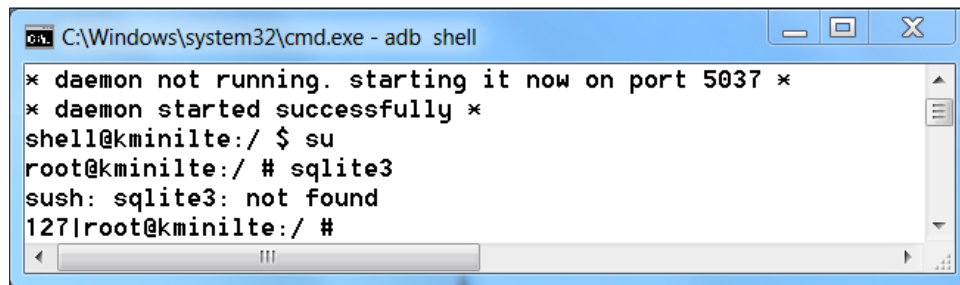


Figure 8.1: Confirmation of root access on the Android smartphone

8.2.1 Manipulation of Android Smartphone Data

The first exploratory experiment focuses on the Android platform, and the test smartphone is a Samsung Galaxy S5 Mini, running Android version 6.0.1 (Marshmallow). Manipulation of the smartphone data associated with Android’s default messaging application relies on access to the storage structure responsible for storing the data. The Android platform stores all application-related smartphone data in the `/data` folder. Only a rooted smartphone provides access to the `/data` folder. Obtaining root access on the Samsung Galaxy S5 Mini requires the execution of the CF Auto Root and Odin tools. Figure 8.1 confirms the availability of root access and the automatic installation of a root application. Also, access to the root directory (`/`) also requires the enabling of the USB debugging functionality (Lessard and Kessler 2010), 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 checkbox next to “USB debugging” will enable this feature.

Following the enabling of the “USB debugging” feature, it is necessary to establish



```
C:\Windows\system32\cmd.exe - adb shell
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
shell@kminilte:/ $ su
root@kminilte:/ # sqlite3
sush: sqlite3: not found
127|root@kminilte:/ #
```

Figure 8.2: Unavailability of the `sqlite3` command-line program

next a communication channel between a shell running on a computer and the Android smartphone. ADB is a versatile command-line utility that communicates with a connected Android smartphone (*Android Developers 2018c*). The communication channel is established using `adb shell`, followed immediately by the `su` command. Execution of the `su` command creates access to the root directory along with the necessary permissions.

Android's default messaging application uses an SQLite database for storing persistent data. Currently, the SQLite database (`mmssms.db` and `mmssms.db-wal`) is located in the `/data/data/com.android.providers.telephony/databases/` 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. The removal of the SQLite database files and the subsequent smartphone reboot delete 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. Direct manipulation of the data in the SQLite database files is not feasible due to the complex structure of the files and the possibility of the WAL file overwriting the applied changes. Two approaches exist to access the SQLite database files: direct or off-device.

The first approach involves the direct manipulation of the smartphone data by opening the SQLite database on the Android smartphone. Direct manipulation requires the use of an appropriate tool, such as the `sqlite3` command-line program that allows for the manual entering and execution of SQL statements (*SQLite 2017d*). 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`). Generally, Android smartphones do not include a pre-installed `sqlite3` command-line program, as confirmed in Figure 8.2. The absence of or failure to utilise the `sqlite3` command-line program effectively requires the transferral of the SQLite database (both the `.db` and `.db-wal` files) to the connected computer that supports an SQLite editor.

The second approach uses the established connection to transfer the SQLite database files to and from the connected computer. It is necessary to first transfer the files to the `/sdcard` folder before downloading the files to the connected computer. The `/sdcard` folder is available on all Android smartphones, regardless of make or model, and allows end-users to store additional files and data. Using an SQLite editor allows for viewing and editing of SQLite database records. Opening the `.db` file causes an automatic checkpoint to occur and ensures the transferral and visibility of all the records in the WAL file (`.db-wal`). 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, it is necessary to close the SQLite database to ensure the SQLite database correctly captures the changes.

The `.db` file is the only remaining file, which now contains the manipulated smartphone data. It is necessary to return the `.db` file to the Android smartphone by first transferring the file to the `/sdcard` folder, replacing the previous file. Before returning the changed `.db` file to the `/data/data/com.android.providers.telephony/databases/` folder, the original SQLite database files (`.db` and `.db-wal`) must be removed using the `rm` command. Removal of these files ensures the existing SQLite database does not overwrite the manipulated smartphone data with previously stored records. Thereafter, the `.db` file, which currently resides in the `/sdcard` folder, can be returned to the `/data/data/com.android.providers.telephony/databases/` folder using the `mv` command. The only required file is the `.db-wal` file, which cannot be manually replaced but is generated following a smartphone reboot. It is necessary to first change the current permissions of the `.db` file to allow for the generation of a blank `.db-wal` file upon a smartphone reboot. Changing the permissions of the `.db` file occurs using the `chmod 666` or `chmod a=rw` command. The final step performs a

```

root@kminilte:/data/data/com.android.providers.telephony/databases # ls -l
-rw-rw---- radio radio 57344 2017-12-05 18:22 HbpcdLookup.db
-rw-rw---- radio radio 8720 2017-12-05 18:22 HbpcdLookup.db-journal
-rw-rw---- radio radio 4096 2017-12-05 18:23 mmsms.db
-rw-rw---- radio radio 32768 2017-12-11 17:58 mmsms.db-shm
-rw-rw---- radio radio 683952 2017-12-11 17:47 mmsms.db-wal
-rw-rw---- radio radio 32768 2017-12-05 18:23 nwk_info.db
-rw-rw---- radio radio 21032 2017-12-05 18:23 nwk_info.db-journal
-rw-rw---- radio radio 704512 2017-12-14 20:27 telephony.db
-rw-rw---- radio radio 12824 2017-12-05 18:23 telephony.db-journal
root@kminilte:/data/data/com.android.providers.telephony/databases #

```

(a) SQLite database before manipulation

```

root@kminilte:/data/data/com.android.providers.telephony/databases # ls -l
-rw-rw---- radio radio 57344 2017-12-05 18:22 HbpcdLookup.db
-rw-rw---- radio radio 8720 2017-12-05 18:22 HbpcdLookup.db-journal
-rw-rw-rw- root root 303104 2017-12-14 20:58 mmsms.db
-rw-rw-rw- radio radio 32768 2017-12-14 21:07 mmsms.db-shm
-rw-rw-rw- radio radio 24752 2017-12-14 21:07 mmsms.db-wal
-rw-rw---- radio radio 32768 2017-12-05 18:23 nwk_info.db
-rw-rw---- radio radio 21032 2017-12-05 18:23 nwk_info.db-journal
-rw-rw---- radio radio 704512 2017-12-14 21:07 telephony.db
-rw-rw---- radio radio 12824 2017-12-05 18:23 telephony.db-journal
root@kminilte:/data/data/com.android.providers.telephony/databases #

```

(b) SQLite database after manipulation

Figure 8.3: Changes to the file structure of the SQLite database files (Android)

smartphone reboot to ensure the creation of the `.db-wal` file and the visibility of manipulated data on the Android smartphone. Figure 8.3 captures the changes occurring to the SQLite database file structure as a direct result of the applied manipulation.

The reboot of the Android smartphone concludes the exploratory experiment that focuses on the manipulation of Android smartphone data. The following section attempts the manipulation of data residing on the iOS platform.

8.2.2 Manipulation of iOS Smartphone Data

The second exploratory experiment focuses on the iOS platform, and the test smartphone is an Apple iPhone 7, running iOS version 10.0.1. Manipulation of the smartphone data that forms part of the default messaging application of the iPhone 7 necessitates access to the storage structure responsible for storing the data. The iOS platform stores all application-related smartphone data in the `/private/var/mobile/Library` folder. Only jailbroken smartphones provide access to this folder. Performing a jailbreak of

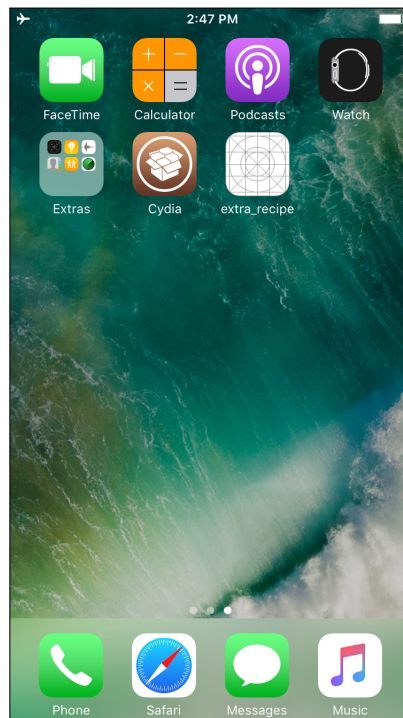


Figure 8.4: Confirmed installation of the Cydia application on the iPhone 7

the iPhone 7 is possible 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 automatic installation of the Cydia application (see Figure 8.4), a package manager for jailbroken iPhones, confirms the jailbreak status.

iPhone's default messaging application also uses an SQLite database for storing persistent data. Currently, the location of the SQLite database (`sms.db` and `sms.db-wal`) is 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. As with Android smartphones, direct manipulation of the data captured in the SQLite database files is not feasible due to the complex structure of the files and the possibility of over-

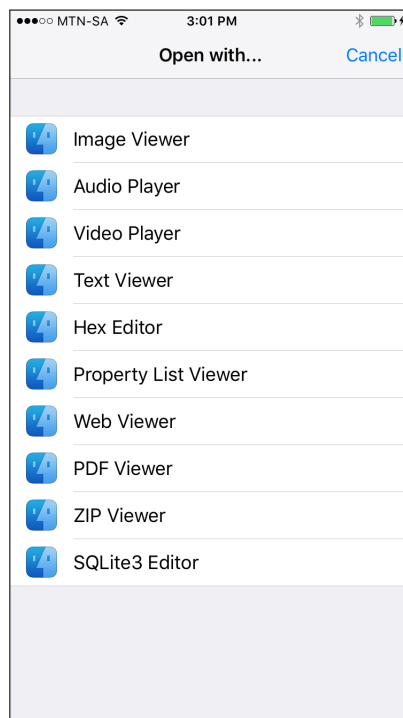


Figure 8.5: Confirmed presence of the `sqlite3` program on the iPhone 7

writing the manipulated data with existing data. It is necessary to acquire access to the SQLite database records, which is possible via one of the following two approaches: (1) direct or (2) off-device.

The first approach involves the direct 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. Generally, iOS comes pre-installed with the `sqlite3` program, which provides 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. The `sqlite3` program is accessible either using a terminal or the iFile application, both of which offers direct access to the SQLite database and allows for the execution of SQL statements (see Figure 8.5). Should the `sqlite3` program fail to apply the changes to the smartphone data effectively, it will be necessary to transfer the SQLite database to a computer that supports an SQLite editor.

The second approach performs off-device manipulation and requires the transferral

of the SQLite database (both the `.db` and `.db-wal` files) to a connected computer. Establishing a connection between the iPhone 7 and the computer is possible using the iFunbox and puTTY applications. Using the established connection and standard iOS credentials, namely root (username) and alpine (password), provide direct access to the file system of the iPhone 7. After that, transferral of the SQLite database to and from the iPhone 7 becomes possible using the established connection. The transfer occurs by first moving the `.db` and `.db-wal` files to the `/var/mobile/Media` folder before downloading the files to 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. Usage of the SQLite editor again makes it possible to open the `.db` file. Opening the `.db` file causes an automatic checkpoint to occur and ensures the transferral and visibility of the records available in the WAL file (`.db-wal`) in the editor. It is now possible to manipulate the smartphone data using the available SQL statements (`INSERT`, `UPDATE` or `DELETE`). It is necessary to close the SQLite database again to complete the manipulation of the smartphone data and ensure the SQLite database correctly captured the changes.

Similar to the first experiment, the `.db` file is the only remaining file, which now contains the manipulated smartphone data. It is necessary to return the `.db` file to the iPhone 7 by copying the `.db` file to the `/private/var/mobile/Media` folder, replacing the previous file. Before returning the file to the `/private/var/mobile/Library/SMS` application folder, it is necessary to first remove the existing SQLite database (`.db` and `.db-wal` files) using the `rm` command. Removal of these files, especially the `.db-wal` file, ensure the SQLite database does not overwrite the manipulated smartphone data using previously stored records. Only then is it possible to transfer the `.db` file currently residing in the `/private/var/mobile/Media` folder to the `/private/var/mobile/Library/SMS` folder using the `mv` command.

The only required file is the `.db-wal` file, which cannot be manually replaced but is generated following a smartphone reboot.

Ensuring the creation of the `.db-wal` file necessitates a change to the permissions of the `.db` file. Changing the permissions allow for the generation of a blank `.db-wal` file upon a smartphone reboot. It is possible to change the permissions of the main database

```

MobileAsset      empty      logs      root      wireless
MobileDevice     folders    mobile    run
MobileSoftwareUpdate  installd  msgs      spool
Heloises-iPhone:/var root# cd mobile/Library/SMS
Heloises-iPhone:/var/mobile/Library/SMS root# ls -l
total 244
drwxr-xr-x  2 mobile mobile    68 Dec  7 08:43 Drafts
drwxr-xr-x  2 mobile mobile    68 Dec  7 08:37 EmergencyAlerts
-rw-r--r--  1 mobile mobile   261 Dec  7 08:48 com.apple.messages.geometrycache_
v1.plist
-rw-r--r--  1 mobile mobile   4096 Dec  7 08:36 sms.db
-rw-r--r--  1 mobile mobile  32768 Dec  7 08:36 sms.db-shm
-rw-r--r--  1 mobile mobile 206032 Dec  7 08:36 sms.db-wal
Heloises-iPhone:/var/mobile/Library/SMS root# █

```

(a) SQLite database before manipulation

```

login as: root
root@127.0.0.1's password:
Heloises-iPhone:~ root# cd /var/mobile/Library/SMS
Heloises-iPhone:/var/mobile/Library/SMS root# ls -l
total 276
drwxr-xr-x  3 mobile mobile   102 Dec  7 11:24 Drafts
drwxr-xr-x  2 mobile mobile    68 Dec  7 08:37 EmergencyAlerts
-rw-r--r--  1 mobile mobile   856 Dec  7 11:25 com.apple.messages.geometrycache_
v1.plist
-rw-rw-rw-  1 root   mobile 180224 Dec  7 11:22 sms.db
-rw-r--r--  1 mobile mobile  32768 Dec  7 11:30 sms.db-shm
-rw-rw-rw-  1 mobile mobile  61832 Dec  7 11:30 sms.db-wal
Heloises-iPhone:/var/mobile/Library/SMS root# █

```

(b) SQLite database after manipulation

Figure 8.6: Changes to the file structure of the SQLite database files (iOS)

file as required using the `chmod 666` or `chmod a=rw` command. The final step is a smartphone reboot to ensure the creation of the `.db-wal` file and the visibility of the manipulation on the iPhone 7. Figure 8.6 captures the changes to the file structure of the SQLite database after performing the manipulation.

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

	Stage	Requirements
1	Set Smartphone State	<ul style="list-style-type: none"> Establish physical access to smartphone (Root/Jailbreak)
2	Locate Storage Structure	<ul style="list-style-type: none"> Select smartphone application Access application folder Identify structure storing the smartphone data
3	Access Smartphone Data	<ul style="list-style-type: none"> Identify most appropriate approach to access the data Direct or Off-device
3.1	Direct	<ul style="list-style-type: none"> Open the storage structure directly on the smartphone Manipulate the smartphone data (modify, fabricate or delete) Close the storage structure
3.2	Off-device	<ul style="list-style-type: none"> Establish a connection between smartphone and computer Transfer storage structure to connected computer Open the storage structure Manipulate the smartphone data (modify, fabricate or delete) Close the storage structure Return the storage structure to the connected smartphone Assign read/write permissions to the storage structure
4	Reboot	<ul style="list-style-type: none"> Manually reboot/restart the smartphone

Figure 8.7: Generic process for smartphone data manipulation

8.2.3 Formulation of the Generic Process

The conducted exploratory experiments confirm that it is indeed possible to manipulate smartphone data on both Android and iOS platforms. From the experiments, it is possible to identify various similarities between the steps followed to manipulate smartphone data. These similarities are conceptualised into a generic process that generalises the manipulation of smartphone data. Figure 8.7 illustrates the generic process, which consists of four distinct stages. Each stage describes the progression of the generic process to manipulate the smartphone data along with the requirements that must be met to complete the stage successfully.

The first stage of the generic process ensures the selected smartphone is in the correct state to allow for interaction with the smartphone data. This interaction necessitates physical access to the smartphone by confirming the smartphone is either rooted (An-

droid) or jailbroken (iOS). An accessible smartphone also supports the establishment of a connection between the smartphone and a computer, which is required for the off-device manipulation approach.

The second stage of the generic process focuses explicitly on the selection of the application and locating the storage structure, such as an SQLite database, storing the persistent smartphone data. The selected smartphone application must use a storage structure that resides on the smartphone.

The third stage attempts to access the smartphone data using the most appropriate approach: direct or off-device. 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 storage structure. Unavailability of such a program or utility compels the use of the off-device approach. The off-device approach requires the transferral of the storage structure to the connected computer. Using the most appropriate program or utility makes it possible to access and manipulate the contents of the storage structure. Once completed, it is necessary to close the storage structure and return the storage structure to the smartphone to overwrite the original smartphone data. Finally, the assignment of the necessary read/write permissions ensures the smartphone application can interact with the manipulated smartphone data.

Finally, the fourth stage requests a reboot of the smartphone to ensure the manipulated smartphone data is captured and reflects in the user interface of the application.

This proposed generic process for smartphone data manipulation captures the steps to follow to modify, fabricate or delete all or specific smartphone data. Using this generic process permit smartphone end-users to attack the authenticity of smartphone data and mislead digital forensic professionals during the examination of the data. The following section investigates the construction of an attack tree to encapsulate the various attack scenarios available to manipulate smartphone data.

8.3 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 an attack on the authenticity of smartphone

data and further described using an attack tree. An attack tree provides a formal and methodical way to describe various attacks against a system (Schneier 1999). The attacks are represented using a conceptual tree structure with the primary goal of these attacks listed as the root node. The nodes following the root describe 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.

The remainder of this section focuses on the construction of an attack tree that describes the step to accomplish smartphone data manipulation. From this attack tree, two distinct attack scenarios are identified that can be followed to manipulate smartphone data.

8.3.1 Attack Tree Construction

For the construction of this attack tree, “manipulation of smartphone data” is set as the goal and denoted by G . The intermediate causes of the goal are deletion, modification or fabrication, which are respectively marked with I_1 , I_2 , I_3 . Following the intermediate goals are the sub-goals that describe the required steps to accomplish each intermediate goal.

There are two options for deletion (I_1): removal of the storage structure, which deletes all of the data (S_1) or removing specific data such as individual records (S_2). Removal of the storage structure requires access to the smartphone (S_5), which is achieved by either rooting (Android) or jailbreaking (iOS) the smartphone. Once access is acquired, it is necessary to pinpoint the location of the storage structure (S_6), remove the required files and reboot the smartphone (S_7). Removal of the files ensures the deletion of all the smartphone data related to a specific smartphone application. Removal of specific data also requires access to the smartphone and locating the storage structure holding the data. Since this attack focuses on the manipulation of specific data, it is necessary to access the storage structure (S_8). Opening the storage structure occurs either directly on the smartphone (S_9) or off-device on a connected computer (S_{10}). Opening and viewing the data collected by the storage structure on the smartphone emphasise the need for 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 storage structure must

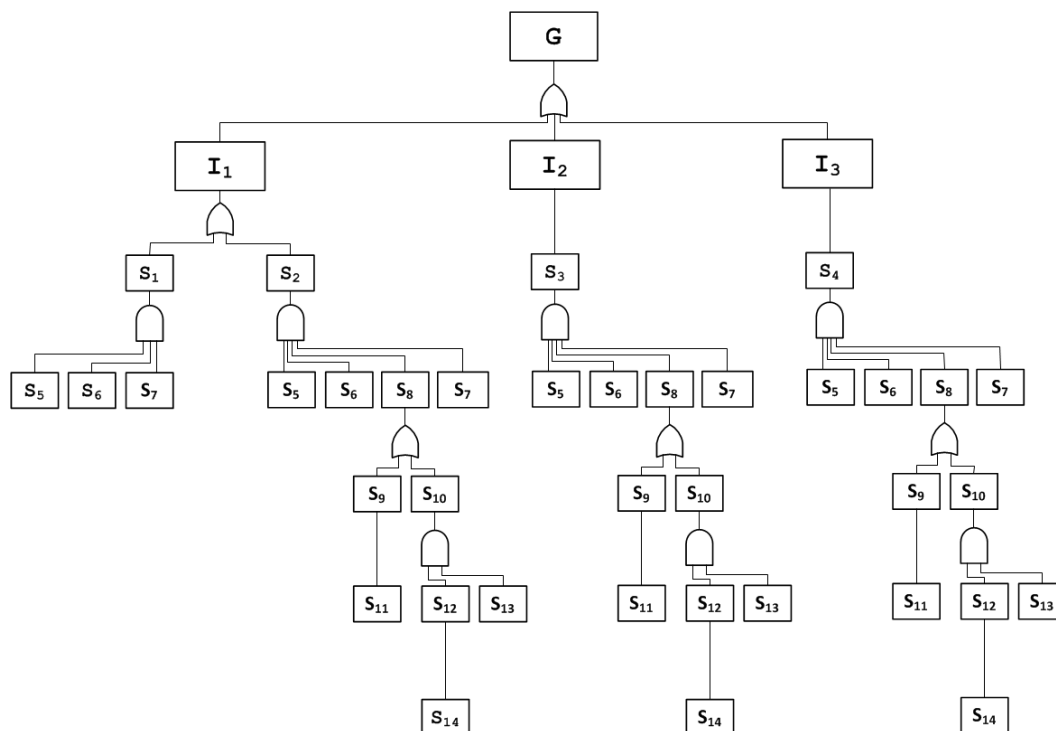


Figure 8.8: Attack tree for smartphone data manipulation

occur off-device on a computer connected to the smartphone. Off-device manipulation requires the transferral of the storage structure (S_{12}), which relies on an established connection between the smartphone and the connected computer (S_{14}). After completing the manipulation, the changed storage structure must be returned to the smartphone via the established connection (S_{13}). The return of the storage structure is followed by a smartphone reboot (S_7) to ensure the manipulation reflects on the smartphone.

The remaining manipulation techniques, modification (I_2) and fabrication (I_3), follows similar attack paths. To either change existing (S_3) or insert fabricated data (S_4), it is necessary to access the storage structure. Therefore, these manipulation techniques follow a path identical to the removal of specific data. Again a smartphone reboot (S_7) follows to ensure the changes take effect.

According to the descriptions above, it is possible to construct the attack tree shown in Figure 8.8. The attack tree forms the basis for deriving attack scenarios to manipulate smartphone data.

8.3.2 Attack Scenarios

The constructed attack tree (see Figure 8.8) offers the necessary insight to derive attack scenarios that lead to the manipulation of smartphone data. Each path in the attack tree illustrates a different avenue to accomplish the manipulation of the smartphone data, which includes modification, fabrication or deletion. Although the attack tree encapsulates various attack scenarios, the focus of this section will only be on two distinct attack scenarios.

The outcome of the first attack scenario is the manipulation of smartphone data by fabricating and inserting new data. For this attack, the Android operating system is identified as the target platform and requires rooting. The application selected to hold the fabricated data is the WhatsApp application. The fabricated smartphone data is, therefore, a new message created as part of an existing conversation. Due to the unavailability of the `sqlite3` command-line program on the Android platform, the fabrication of the smartphone data occurs off-device. For this attack scenario to be successful, the fabricated message must show in the WhatsApp application.

Using the provided attack tree, the described attack scenario is denoted as follows:

$$AS_1 = \{I_3, S_4, S_5, S_6, S_8, S_{10}, S_{12}, S_{14}, S_{13}, S_7\}$$

The focus of the second attack scenario is the modification of existing smartphone data by deleting specific information forming part of an individual record. For this attack, the iOS operating system is identified as the target platform and involves a previously jailbroken iPhone. The Safari web browser application contains the smartphone data that requires manipulation. The manipulation of this data occurs directly on the smartphone since the iOS platform includes the `sqlite3` command-line program. It is, therefore, not necessary to transfer the smartphone data and conduct off-device manipulation. For this attack scenario to be successful, the manipulated smartphone data must not show in the user interface of the Safari application. Again using the provided attack tree, the described attack scenario is denoted as follows:

$$AS_2 = \{I_2, S_3, S_5, S_6, S_8, S_9, S_{11}, S_7\}$$

The presented attack scenarios describe the manipulation of smartphone data using different techniques (fabrication and modification) and by following different approaches (direct and off-device) on both the Android and iOS platforms. Although the focus is only on two attack scenarios, these scenarios include most of the sub-goals required to achieve the set goal. These attack scenarios thus provide a comprehensive overview of achieving the attack goal, which is the manipulation of smartphone data.

8.4 Evaluation of Smartphone Data

The constructed attack tree along with the derived attack scenarios present suitable techniques to manipulate smartphone data. Such manipulation impacts and negatively influences the authenticity of the smartphone data. It becomes, therefore, essential to establish the authenticity of smartphone data and confirm the data refers to actual events. It is possible to evaluate the authenticity of smartphone data using the SADAC tool, which encapsulates the smartphone data evaluation and classification models. Applying the identified attack scenarios (see Section 8.3.2) at a theoretical level allows for the evaluation of the practicality of the models to evaluate the authenticity of smartphone data. The developed SADAC tool makes it possible to measure the effects of the applied attack scenarios on the authenticity of the data.

8.4.1 Attack Scenario 1

The first attack scenario attempts the creation and insertion of fabricated data on a non-rooted Android smartphone. The fabricated data is a new message that forms part of the WhatsApp application. Inserting the fabricated data necessitates access to the smartphone. As the Android smartphone is currently non-accessible, the appropriate steps must be taken to root the Android smartphone. Following the rooting of the Android smartphone, it is necessary to locate the storage structure of the Contacts application. The WhatsApp application stores the persistent data in an SQLite database located in the `/data/data/com.android.providers.contacts/` folder. Unavailability of the `sqlite3` command-line program entails the transferral of this SQLite database to a connected computer. After completing the transfer of the SQLite database files, the

Table 8.1: Traces created by Attack Scenario 1

Trace No.	Sub Goal	Trace Created
T_1	S_5	Automatic installation of a root application.
T_2	S_5	Unavailability of OTA updates.
T_3	S_8	Discrepancy between WAL file and application usage timestamps.
T_4	S_{12}	Change in ownership of storage structure files.
T_5	S_{12}	Change in permissions of storage structure files.
T_6	S_{14}	Presence of additional enabled setting (USB debugging).
T_7	S_{13}	An increase in the main database file size over the WAL file size.
T_8	S_{13}	The presence of a clean WAL file.
T_9	S_7	The creation of a new entry in the reboot log.

Table 8.2: Attack Scenario 1 evaluation (End-user Behaviour)

Assessment Point	Outcome	Reason
$ \text{WAL timestamp} - \text{App usage timestamp} < \Delta$	No	Presence of T_3
No smartphone reboot after WAL timestamp	No	Presence of T_9
Smartphone not rebooted	No	Presence of T_9
Anti-forensic applications absent	Yes	No trace indicates the presence of anti-forensic applications.

fabricated data, which is a new message, is inserted as a new record in the database. The final step of the attack scenario close and return the SQLite database to the Android smartphone. Immediately following the final step is the reboot of the Android smartphone to ensure the new message reflects in the WhatsApp application.

Completion of the attack scenario described above will have inherent side-effects that

Table 8.3: Attack Scenario 1 evaluation (Smartphone Operational State)

Assessment Point	Outcome	Reason
Not rooted or jailbroken	No	Presence of T_1 and T_2
Root/jailbreak application not installed	No	Presence of T_1
Critical files present	Yes	All critical files present.

Table 8.4: Attack Scenario 1 evaluation (Smartphone Application Behaviour)

Assessment Point	Outcome	Reason
Persistent data = user interface data	Yes	Persistent data resembles data viewed via the user interface.
Consistent order of database records	Yes	Database records ordered correctly.
WAL file size > main database file size	No	Presence of T_7 and T_8
Current application the owner	No	Presence of T_4
Expected file permissions	No	Presence of T_5

Table 8.5: Attack Scenario 1 evaluation (External Environment)

Assessment Point	Outcome	Reason
Incoming data \approx outgoing data	N/A	Assessment point not evaluated due to the unavailability of the other smartphone.
Mobile network provider records \approx persistent data	N/A	Assessment point not evaluated because of unavailable mobile network provider data.

leave various traces on the Android smartphone. Table 8.1 lists the traces specific to this attack scenario. Rooting the Android smartphone (S_5) causes the automatic installation of a root application (T_1) and prevents the availability of OTA updates (T_2). Gaining access to the persistent data stored in the SQLite database (S_8) without accessing the application causes a discrepancy between the last modification timestamp of the WAL file and the last used timestamp of the application (T_3). The discrepancy is due to the difference between the timestamps being greater than the acceptable time interval. The transferral of the SQLite database files (S_{12}) leads to changes in both the ownership (T_4) and permissions (T_5) of the SQLite database files. This transferral of the SQLite database files relies on the established connection between the Android smartphone and the computer (S_{14}), which requires the enabling of the “USB debugging” setting (T_6). Upon returning the SQLite database file (S_{13}) and rebooting the Android smartphone (S_7) to capture the changes, a new and clean WAL file is created (T_8) that is smaller in size than the main database file (T_7). Finally, the smartphone reboot (S_7) also cause the creation of a new entry in the reboot log (T_9) located in `/data/system/dropbox/` folder.

Identification of these traces is of great importance since the traces support the evaluation of the authenticity of the smartphone data. The developed SADAC tool provides the necessary platform to evaluate the smartphone data. The SADAC tool organises the evaluation of the smartphone data according to the four core components of authentic smartphone data, with each component containing a collection of assessment points.

The identified traces makes it is possible to evaluate all of the assessment points and classify the authenticity. Tables 8.2 - 8.5 capture the outcome of all evaluated assessment points and also provide supportive reasoning for each outcome.

Using the results collected in Tables 8.2 - 8.5 as input make it possible to establish the authenticity of the smartphone data using the SADAC tool. The outcome of the authenticity grading is expected to be either “low” or “unsatisfactory” due to the manipulation applied to the smartphone data. Further expected is a “high” completeness interval since the available assessment points of three core components were evaluated. Figure 8.9 presents the authenticity classification, calculated using the SADAC tool.

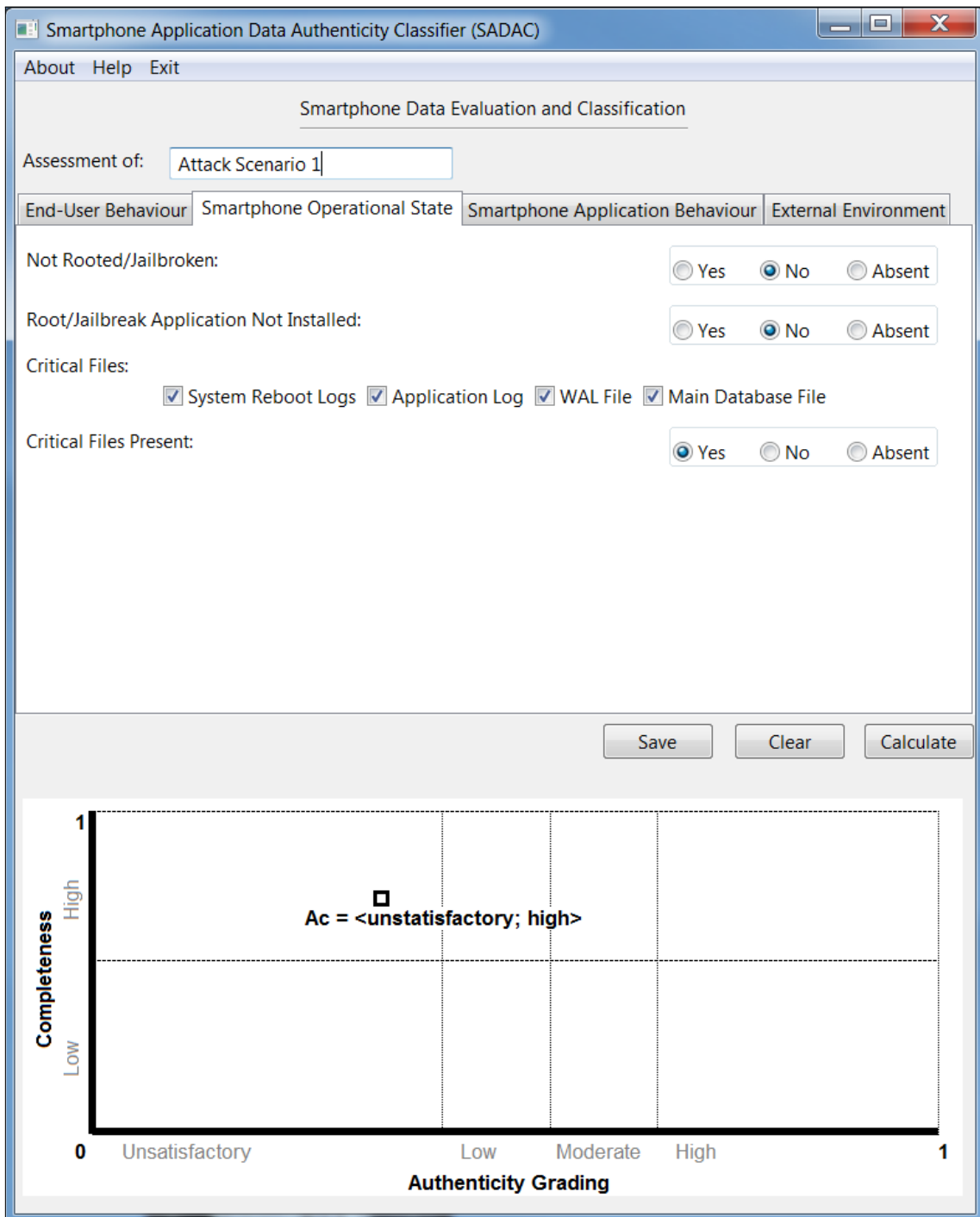


Figure 8.9: Authenticity classification of Attack Scenario 1 smartphone data

The calculated authenticity classification of the smartphone data shows the assignment of an “unsatisfactory” authenticity grading with a “high” completeness interval. This authenticity classification aligns with the predicted outcome, and since the evaluated smartphone data only met some of the required requirements (see Chapter 7), an “unsatisfactory” authenticity grading is acceptable. The assigned authenticity classification confirms the impact of the manipulation on both the authenticity and availability of the smartphone data. The final assignment of the authenticity classification concludes the evaluation of the Android smartphone data.

8.4.2 Attack Scenario 2

The second attack scenario attempts the modification of smartphone data by deleting specific information from an individual record. The manipulation occurs on a previously jailbroken iPhone 7 and involves a bookmark entry of the Safari application. Due to the current jailbreak status of the iPhone 7, it is possible to follow the direct approach and modify the record directly on the iPhone 7. The Safari application stores the persistent data in an SQLite database located in the `/var/mobile/Library/Safari` folder. The iFile application makes it possible to locate the required folder and access the persistent data. After modifying the record, it is necessary first to close the SQLite database to ensure the accurate capturing of the manipulation. Similar to the previ-

Table 8.6: Traces created by Attack Scenario 2

Trace No.	Sub Goal	Trace Created
T_1	S_5	Automatic installation of the Cydia application.
T_2	S_5	Unavailability of OTA updates.
T_3	S_8, S_9	Discrepancy between WAL file and application usage timestamps.
T_4	S_{11}	Usage of the <code>sqlite3</code> program.
T_5	S_8	Presence of a clean WAL file.
T_6	S_7	The creation of entry in the reboot log file.

Table 8.7: Attack Scenario 2 evaluation (End-user Behaviour)

Assessment Point	Outcome	Reason
$ \text{WAL timestamp} - \text{App usage timestamp} < \Delta$	No	Presence of T_3
No smartphone reboot after WAL timestamp	No	Presence of T_3 and T_6
Smartphone not rebooted	No	Presence of T_6
Anti-forensic applications absent	Yes	No trace indicates the presence of anti-forensic applications.

Table 8.8: Attack Scenario 2 evaluation (Smartphone Operational State)

Assessment Point	Outcome	Reason
Not rooted or jailbroken	No	Presence of T_1 and T_2
Root/jailbreak application not installed	No	Presence of T_1
Critical files present	Yes	All critical files present.

ous attack scenario, the final step is again the reboot of the iPhone 7 to complete the manipulation of the bookmark entry of the Safari application.

Similar to the previous attack scenario for Android smartphone data, this particular attack scenario will also have inherent side-effects in the form of traces left behind on the iPhone 7. Table 8.6 lists the traces specific to this attack scenario. Jailbreaking the iPhone 7 (S_5) causes the automatic installation of the Cydia application (T_1) and prevents the availability of OTA updates (T_2). Again gaining access to the persistent data stored in the SQLite database (S_8 and S_9) following the direct approach but without

Table 8.9: Attack Scenario 2 evaluation (Smartphone Application Behaviour)

Assessment Point	Outcome	Reason
Persistent data = user interface data	Yes	Persistent data resembles data viewed via the user interface.
Consistent order of database records	Yes	Database records ordered correctly.
WAL file size > main database file size	No	Presence of T_5
Current application the owner	Yes	No change to owner UID.
Expected file permissions	Yes	No change to read/write permissions.

Table 8.10: Attack Scenario 2 evaluation (External Environment)

Assessment Point	Outcome	Reason
Incoming data \approx outgoing data	N/A	Assessment point not evaluated because of unidirectional design of the application.
Mobile network provider records \approx persistent data	N/A	Assessment point not evaluated because of unidirectional design of the application.

accessing the application causes a discrepancy between the last modification timestamp of the WAL file and the last usage timestamp of the application (T_3). The direct approach relies on the use of the `sqlite3` program to acquire access to the persistent data (S_{11}), which causes a change to the last access timestamp associated with the program (T_4). This timestamp will also closely follow the last modification timestamp of the WAL file. Accessing the SQLite database to manipulate the record (S_8) will cause an immediate checkpoint to occur. Therefore, after closing the SQLite database, a clean and empty WAL is present on the iPhone 7. Finally, rebooting the iPhone 7 (S_7) causes the creation

of a new entry in the `/var/mobile/logs/lockdownd.log` reboot log (T_6).

Once again, the identification of these traces is of great importance since the traces supports the evaluation of the authenticity of the smartphone data. The previous evaluation of the first attack scenario showed the developed SADAC tool can classify the authenticity of the smartphone data. Therefore, the SADAC tool will again be deployed to evaluate and classify the smartphone data. The identified traces allows for the evaluation of the relevant assessment points. Tables 8.7 to 8.10 capture the outcome of all the evaluated assessment points and provide supportive reasoning for each outcome.

Again, using the results collected in Tables 8.7 - 8.10 as input make it possible to establish the authenticity of the smartphone data using the SADAC tool. The assigned authenticity grading for this attack scenario may be higher since the manipulation of the smartphone data occurred using the direct approach, which will cause fewer traces to remain on the smartphone for evaluation. The outcome of the authenticity grading for this attack scenario is expected to be either “low” or “moderate”. Further expected is also a “high” completeness interval since a more substantial subset of the available assessments were evaluated. Figure 8.10 presents the authenticity classification of the smartphone data, calculated using the SADAC tool. The calculated authenticity classification shows the assignment of a “low” authenticity grading with a “high” completeness interval. This authenticity classification aligns with the predicted outcome since the evaluated smartphone data met most of the requirements belonging to Classes 1 and 2 (see Chapter 7). However, the assigned authenticity classification still confirms that the smartphone data was indeed manipulated due to the changes made to a single record. The authenticity classification also further confirms that the manipulation does indeed influence the authenticity of the data. The final assignment of the authenticity classification concludes the evaluation of the iPhone 7 smartphone data.

8.5 Summary

The purpose of this chapter was to assess the usability of the established models to evaluate and classify smartphone data. The chapter followed a two-pronged approach to assess the established models. Firstly, it was necessary to determine how smartphone

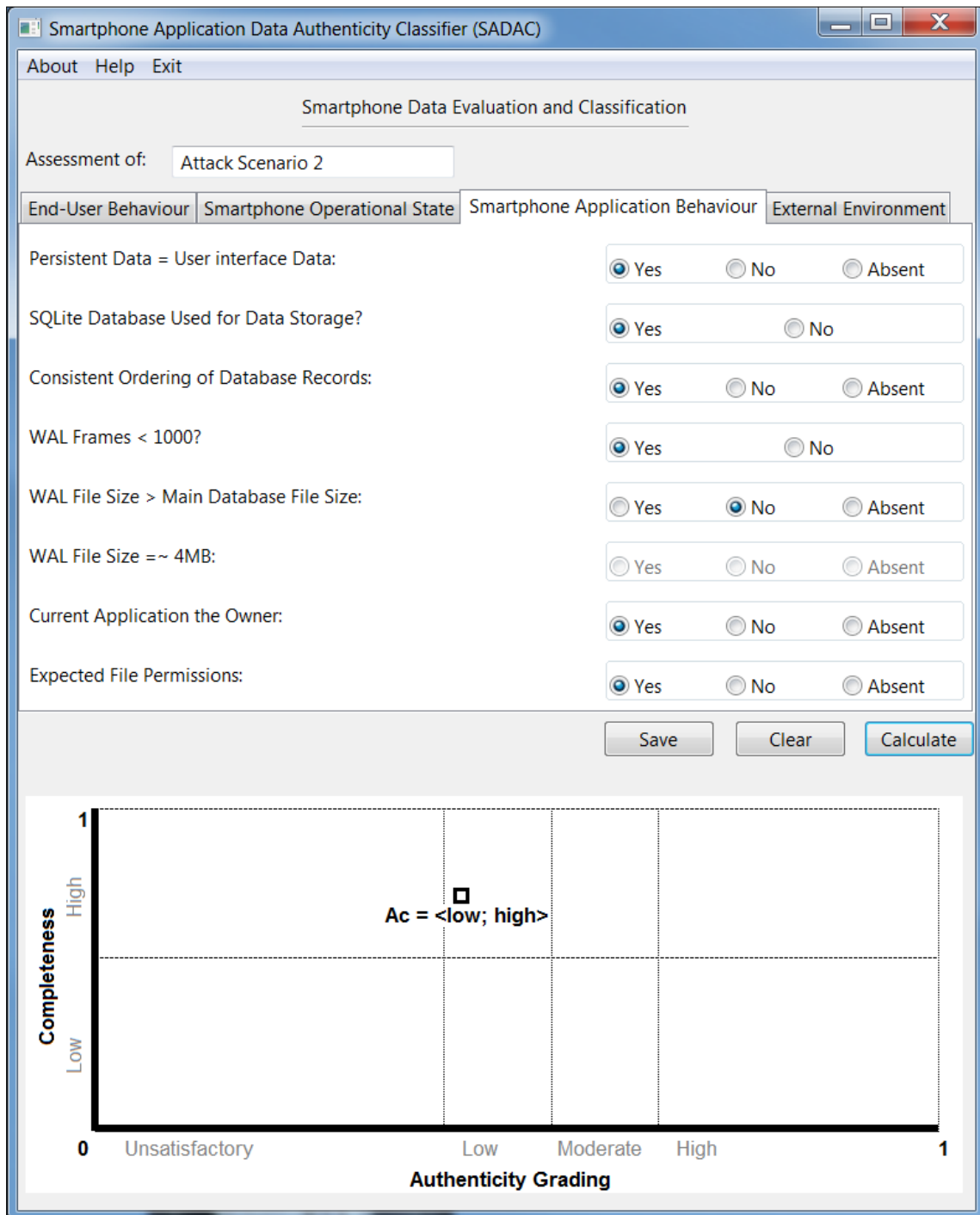


Figure 8.10: Authenticity classification of Attack Scenario 2 smartphone data

data can be manipulated on both the Android and iOS platforms, which led to the formulation of a generic process for smartphone data manipulation. Secondly, an attack tree was constructed using the generic process and, from the attack tree, various scenarios were derived to manipulate smartphone data. Application of these attack scenarios at a theoretical level permitted the evaluation of the effects using the developed SADAC tool. The collective results produced by theoretically evaluating the smartphone data associated with each attack scenario showed that classifying the authenticity of smartphone data is feasible. Derived from the results was the finding of smartphone data with an authenticity grading below “excellent” implies the manipulation of the data. Digital forensic professionals can, therefore, use the provided models and developed SADAC tool to measure the authenticity of smartphone data before submitting such data as digital evidence. The following chapter concludes the thesis.

Chapter 9

Conclusion

The focus of this research study was on the evaluation of smartphone data and establishing the authenticity of such data. To establish the authenticity of the smartphone data a better understanding of the expected behaviour of smartphone applications is required. Obtaining such an understanding is possible by modelling the smartphone applications using a reference architecture. The constructed reference architecture provided the necessary insight to identify requirements authentic smartphone data must adhere to, which led to the design of the smartphone data evaluation model. The results produced by the smartphone data evaluation model are used as input for the classification model to establish the authenticity of the smartphone data. Development of the SADAC proof of concept digital forensic tool streamlined and simplified the evaluation and classification of smartphone data.

This last chapter concludes the thesis by revisiting the problem statement and evaluating the extent to which the stated objectives have been accomplished in Section 9.1. Section 9.2 presents and briefly discusses the main contributions of this research study while Section 9.3 gives directions for potential future work.

9.1 Revisiting the Problem Statement and Research Questions

The purpose of this thesis was to examine the problem of *establishing the authenticity of smartphone data*. Section 1.2 describes the research questions needed to formulate a suitable solution to the described problem. Answers to the presented research questions, as well as an appropriate solution for the stated problem statement, is given in this section.

The first research question, **how does smartphone data originate?**, was addressed in Chapter 5 by constructing a new reference architecture for smartphone applications. This reference architecture consists of four architectural components (user interface, application logic, manager and data storage) that model the structure and behaviour of smartphones applications. Capturing the behaviour of smartphone applications are the following models: high-level data flow diagram and finite state machine. These behavioural models, along with the reference architecture, illustrate and confirm the origin of smartphone data. Understanding how smartphone data originates is a necessary step to be able to identify authentic smartphone data.

The following research question, **what is authentic smartphone data?**, was presented in Chapter 6 of the thesis. Formally defining authentic smartphone data necessitates a better understanding of both the term “authenticity” and the interconnected environment in which smartphones operate. The following four components: (i) end-user behaviour, (ii) smartphone operational state, (iii) smartphone application behaviour and (iv) external environment exist in this interconnected environment and support the creation and management of smartphone data. These components form critical pillars in maintaining the authenticity of smartphone data. Therefore, authentic smartphone data is defined as *data that originates as a direct result of the expected operation and normal execution of these four components*.

The following two research questions: **what are the requirements smartphone data must adhere to be deemed authentic?** and **how is smartphone data evaluated using the requirements?**, were also addressed in Chapter 6. For smartphone data to be authentic, it is necessary that the four core components mentioned above op-

erates and executes as expected. For each component, a collection of requirements were identified. These requirements evaluate the expected operational behaviour and confirm the reliability of each component, which is necessary to establish the authenticity of the smartphone data. Therefore, the requirements form the foundation for the construction of the new smartphone data evaluation model, which comprises of three phases: (i) pre-evaluation, (ii) smartphone evaluation and (iii) documentation. The smartphone evaluation phase assesses the smartphone data and, therefore, was structured according to the four core components. Incorporating all four components ensure a comprehensive evaluation of the smartphone data and allow digital forensic professionals to establish the authenticity of the smartphone data.

The final research question, **how can authenticity be classified based on the evaluation of the smartphone data?**, was presented in Chapter 7. The newly introduced smartphone data evaluation model only describes how smartphone data must be evaluated and not what the outcome of the evaluation is. Therefore, a new classification model for smartphone data was constructed using the identified requirements and smartphone data evaluation model. Classification of the evaluated smartphone data is presented using an ordered pair of values: grade of authenticity (high, moderate, low or unsatisfactory) and completeness of the evaluation (high or low). The presented classification accurately describes the authenticity of the evaluated smartphone data.

By resolving these questions throughout this thesis, the research provides the necessary foundation to address the problem statement presented at the start of this section. The performed research confirms that it is indeed possible to establish the authenticity of smartphone data. The authenticity of smartphone data is determined by using the well-established requirements collected in the smartphone data evaluation model to assign an appropriate classification using the constructed classification model.

9.2 Main Contributions

The research conducted throughout this study involved several key contributions which were required to address the problem of establishing the authenticity of smartphone data. The first contribution of the research is the modelling of smartphone applications

according to the newly constructed reference architecture. This is the first reference architecture to capture the architectural components and model the expected behaviour of both Android and iOS smartphone applications. The purpose of the reference architecture is to allow digital forensic professionals to more easily understand smartphone applications and the origin of the related data. The presence of the reference architecture allowed for key behavioural characteristics of smartphone applications to be identified, which are needed to establish the authenticity of smartphone data.

The second contribution made by this research is the newly constructed smartphone data evaluation model. This model uses the requirements identified for each of the four components to evaluate smartphone data. The purpose of this evaluation model is to provide digital forensic professionals with a comprehensive step-by-step guide to evaluate and review smartphone data. Using the model provides structure and order to evaluation of smartphone data and allow digital forensic professionals to repeat the evaluation and confirm the findings.

The third contribution of the research further extends the previous contribution of the smartphone data evaluation model by introducing a new classification model for smartphone data. The purpose of the classification model is to accurately classify the authenticity of smartphone data evaluated using the smartphone data evaluation model. The classification model relies on two distinct mathematical models to produce the authenticity classification. The first mathematical model uses the established requirements and the authentic grading scale to assign an appropriate authenticity grade to the evaluated smartphone data. The second mathematical model establishes the completeness of the evaluation by confirming the number of requirements assessed. Collectively, the authenticity classification allows digital forensic professionals to confirm the authenticity of the evaluated smartphone data with certain confidence. The outcome produced by the classification model provides digital forensic professionals with the necessary insight to either include or exclude the smartphone data as part of a criminal or civil case.

The fourth contribution of this thesis is the proof of concept digital forensic tool called the Smartphone Application Data Authenticity Classifier or SADAC. SADAC was developed explicitly as a support tool for digital forensic professionals and is responsible for automating, as well as ensuring the accuracy of the calculated authenticity

classification. The results produced by the SADAC tool is presented using the authenticity classification graph. This visualisation permits digital forensic professionals to more easily compare subsequent results of the same evaluation or compare different evaluation results. The theoretical evaluation of smartphone data based on different attack scenarios confirmed the operative and practical use of the SADAC tool to classify the authenticity of evaluated smartphone data.

The fifth and final contribution of the research is the formulation of a generic process for smartphone data manipulation. The generic process describes stepwise how to manipulate smartphone data on either the Android or iOS platform. Using this generic process makes it possible to derive an attack tree that provides various attack avenues to manipulate smartphone data. Confirming that it is indeed possible to manipulate smartphone data further emphasises the need for and the importance of establishing the authenticity of smartphone data.

9.3 Future Work

Mobile forensics, and especially the field of smartphone forensics, is still a growing discipline. The research conducted in this thesis contributes to both mobile and smartphone forensics by introducing well-established techniques to determine the authenticity of smartphone data. The identification and evaluation of authentic smartphone data can still be further improved by continued investigation of the potential future research avenues described below.

The research narrowly focused on the most prevalent mobile operating system platforms, namely Google Android and Apple iOS. Although these platforms currently occupy almost the entire market share for mobile operating systems with 98.69%, other operating systems are still being continuously supported and developed. Future research can extend the existing reference architecture for smartphone applications and the smartphone data evaluation model to support other mobile operating systems. Potential mobile operating systems to explore can include Tizen and Android Go (Android Oreo Go Edition), as well as other variations of the Android operating system (LineageOS or OxygenOS). Including other mobile operating system platforms will further

enhance the relevance and functionality of both the reference architecture and evaluation model.

Another possible avenue for future research is to investigate the extension of both the existing reference architecture for smartphone applications and the smartphone data evaluation model to support other forms of stored data. The current focus of this research was only aimed at application-related smartphone data that resides in internal storage kept directly on smartphones. However, recent trends show the adoption of cloud services to improve storage capabilities and reduce infrastructure costs. Smartphone application can, therefore, make use of the cloud environment to store application-related data. Extension of both the reference architecture and evaluation model to include data stored in the cloud environment allows for more smartphone applications to be supported.

Furthermore, future research may also consider extending the evaluation and classification models to other software applications. Although the focus here is only for smartphone applications, these newly introduced models can be adapted to support other software applications. Such adaptation is possible by identifying the important aspects of the models and establishing generic frameworks to conduct data evaluation and authenticity classification.

Finally, mobile technology continuously evolves at a rapid rate. This dynamic environment ensures endless changes to existing mobile operating systems, mobile hardware technology, as well as the form of interaction between end-users and smartphones. Such changes will necessitate the continuous review of the proposed smartphone data evaluation and classification models. It will become necessary for future research to continuously investigate the growing field of mobile technology and update or adapt the models accordingly.

Appendix A

Publications

Throughout this research study, key contributions have been published in the following conference and journal papers.

- Pieterse H., Olivier M. (2014) *Smartphones as Distributed Witnesses for Digital Forensics*. In: Peterson G., Shenoi S. (eds) *Advances in Digital Forensics X*, Vol. 433, pp. 237 - 251.
- Pieterse H., Olivier M.S., van Heerden R.P. (2015) *Playing hide-and-seek: Detecting the Manipulation of Android Timestamps*. In: *Information Security for South Africa (ISSA)*, Johannesburg, South Africa, 12-13 August, pp. 1 - 8.
- Pieterse H., Olivier, M.S., van Heerden R.P. (2016) *Reference Architecture for Android Applications to Support the Detection of Manipulated Evidence*. *SAIEE Africa Research Journal*, Vol. 107, No. 2, pp. 92 - 103.
- Pieterse H., Olivier M. (2017) *Evaluating the Authenticity of Smartphone Evidence*. In: Peterson G., Shenoi S. (eds) *Advances in Digital Forensics XIII*, Vol. 511, pp. 41 - 61.
- Pieterse H. (2017) *Assisting Digital Forensics Investigations by Identifying Social Communication Irregularities*. In: *Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS 2017)* Dublin, Ireland, 29-30 June, pp. 322 - 331.

- Pieterse H., Olivier M.S., van Heerden, R.P. (2018) *Smartphone Data Evaluation Model: Identifying Authentic Smartphone Data*. Digital Investigation, Vol. 24, pp. 11 - 24.
- Pieterse H., Olivier M.S., van Heerden, R.P. (2018) *Evaluation of Smartphone Data using a Reference Architecture*. International Journal of Electronic Security and Digital Forensics, Vol. 11, No. 2, pp. 160 - 182.
- Pieterse H., Olivier M.S., van Heerden R.P. (2018) *Detecting Manipulated Smartphone Data on Android and iOS Devices*. In: Venter H., Looock M., Coetzee M., Eloff M., Eloff J. (eds) Information Security. ISSA 2018. Communications in Computer and Information Science, Vol. 973, pp. 89 - 103.
- Pieterse H., Olivier, M.S., van Heerden R.P. (2016) *Evaluation Framework for Detecting Manipulated Smartphone Data*. SAIEE Africa Research Journal, Vol. 110, No. 2, pp. 67 - 76.
- Pieterse H., Olivier M.S., van Heerden R.P. (2019) *Classifying the Authenticity of Evaluated Smartphone Data*. In: Fifteenth Annual IFIP WG 11.9 International Conference on Digital Forensics, Orlando, Florida, 28-29 January. Accepted for publication.

Appendix B

Abbreviations

Below is a list of all the abbreviations used in this work ¹.

ADB Android Debug Bridge

AOSP Android Open Source Project

APFS Apple File System

API application program interface

APK Android package

ART Android Runtime

CPU central processing unit

DVM Dalvik Virtual Machine

EEPROM electronic erasable programmable read-only memory

EXT extended file system

eMMC embedded Multimedia Card

EXIF exchangeable image file format

¹Abbreviations based on the Chicago style.

GPS Global Positioning System

GSM Global System for Mobile communications

HAL hardware abstraction layer

HFS Hierarchical File System

HFS+ Hierarchical File System Plus

HTTP HyperText Transfer Protocol

ICCID integrated circuit card identifier

ID identifier

IMEI International Mobile Equipment Identity

IMSI International Mobile Subscriber Identity

IPO input-process-output

ISP In-System Programming

JTAG Joint Test Action Group

LDN last dialled number

LTE Long-Term Evolution

MPE+ Mobile Phone Examiner Plus

MVC model-view-controller

NAND Not-AND

NDK Native Development Kit

NFC near-field communication

NIST National Institute of Standards and Technology

OHA Open Handset Alliance

OS operating system

OTA over-the-air

PDA personal digital assistant

PIM personal information management

PIN personal identification number

PNG Portable Network Graphic

PUK personal unblocking key

RAM random-access memory

RASPP Reference Architecture for Smartphone aPPlications

ROM read-only memory

SADAC Smartphone Application Data Authenticity Classifier

SDK software development kit

SQL Structured Query Language

SSH Secure Shell

SIM subscriber identity module

SMS short message service

UFED Universal Forensic Extraction Device

UI user interface

UID user identifier

UICC universal integrated circuit card

USB Universal Serial Bus

VoIP Voice over Internet Protocol

WAL write-ahead log

Wi-Fi Wireless Fidelity

XML eXtensible Markup Language

YAFFS Yet Another Flash File System

YAFFS2 YAFFS version 2

Appendix C

Smartphone Application Data Authenticity Classifier

This appendix presents key methods of the SADAC digital forensic tool. The methods describe the calculation of both the authenticity and completeness scores, as well as the construction of the authenticity grading scale. The purpose of the appendix is to offer insight into the inner functionality of the SADAC tool and also further describes the calculation of the final authenticity classification.

C.1 Authenticity Score

The `authenticityScore()` method calculates the authenticity score (A_s) of the smartphone data assessed using the smartphone data evaluation model. Calculation of the authenticity score follows three steps. The first step aggregates the weights of all the evaluated assessment points that produced a positive result per class (pos_c). The second step determines the number of assessments evaluated per class (n_c), regardless of outcome. The result of the first step (pos_c) is then divided by the result of the second step (n_c) and weighed according to the assigned weight (w_c) of the class. The outcomes are aggregated and the final step is to calculate the normalised authenticity score. The normalised authenticity score is more easily comprehended and visualised on the authenticity classification graph.

```
private void authenticityScore() {
    int classA = 0, class1 = 0, class2 = 0, class3 = 0;
    double evaluatedPointsClassA = 0.0, evaluatedPointsClass1 = 0.0,
        evaluatedPointsClass2 = 0.0, evaluatedPointsClass3 = 0.0;

    for(int i = 0; i < receivedAssessmentResults.length; i++) {
        if(receivedAssessmentResults[i] == 1) {
            switch (assessmentPointClassAssignment[i]) {
                case 0: classA++;
                break;
                case 1: class1++;
                break;
                case 2: class2++;
                break;
                case 3: class3++;
                break;
            }
        }
    }

    if(receivedAssessmentResults[i] == 1 ||
        receivedAssessmentResults[i] == 0) {
        switch (assessmentPointClassAssignment[i]) {
            case 0: evaluatedPointsClassA++;
            break;
            case 1: evaluatedPointsClass1++;
            break;
            case 2: evaluatedPointsClass2++;
            break;
            case 3: evaluatedPointsClass3++;
        }
    }
}
```

```
                break ;
            }
        }
    }

    double classAScore = weights[0] *
        (classA/evaluatedPointsClassA);
    double class1Score = weights[1] *
        (class1/evaluatedPointsClass1);
    double class2Score = weights[2] *
        (class2/evaluatedPointsClass2);
    double class3Score = weights[3] *
        (class3/evaluatedPointsClass3);
    double classBScore = class1Score + class2Score + class3Score;
    double authenticityScore = classAScore + classBScore;

    normalisedAuthenticityScore = (authenticityScore) * 100;
}
```

C.2 Completeness Interval

The `completenessInterval()` method represents the calculation of the completeness score (C_s), which describes the completeness of the evaluated smartphone data. The method determines the number of assessment points evaluated per component (see Tables 7.2 to 7.5 in Chapter 7). Each calculated value is then equally weighted and aggregated to determine the completeness score. Normalisation of the completeness score also occurs to align with the normalised authenticity score and allow for better representation of the completeness score on the authenticity classification graph.

```
private void completenessInterval() {
```

```
double core1Count = 0;
double core2Count = 0;
double core3Count = 0;
double core4Count = 0;

double core1Total = 4;
double core2Total = 3;
double core3Total = 6;
double core4Total = 2;

for(int i = 0; i < receivedSelectionArray.length; i++) {
    if(coreComponentAssignment[i] == 1 &&
        receivedSelectionArray[i] == 1) {
        core1Count++;
    }
    else if(coreComponentAssignment[i] == 2 &&
        receivedSelectionArray[i] == 1) {
        core2Count++;
    }
    else if(coreComponentAssignment[i] == 3 &&
        receivedSelectionArray[i] == 1) {
        core3Count++;
    }
    else if(coreComponentAssignment[i] == 4 &&
        receivedSelectionArray[i] == 1) {
        core4Count++;
    }
}

double componentWeight = 0.25;
completenessScore =
```



```

        ((core1Count/core1Total) * componentWeight) +
        ((core2Count/core2Total) * componentWeight) +
        ((core3Count/core3Total) * componentWeight) +
        ((core4Count/core4Total) * componentWeight);

    normalisedCompletenessScore = (completenessScore * 100);
}

```

C.3 Authentic Grading Scale Construction

The final method, `authenticityScaleConstruction()`, is responsible for the creation of the authentic grading scale. The purpose of the authentic grading scale is to assign an appropriate grade of authenticity to the evaluated smartphone data based on the calculated authenticity score. Construction of the authentic grading scale follows three distinct steps. The first step identifies which assessment points were evaluated and the second step determines all possible outcomes pertaining to the evaluation of the selected assessment points. The final step arranges and divides the outcomes into four quartiles, which represents the four distinct grades of authenticity (High, moderate, low or unsatisfactory).

```

private void authenticityScaleConstruction () {
    int [][] bArray = binaryArray(assessmentPointsEvaluated);
    double [] weightSelectionArray =
        new double [assessmentPointsEvaluated];
    int count = 0;
    for (int i = 0; i < receivedSelectionArray.length; i++) {
        if (receivedSelectionArray[i] == 1) {
            weightSelectionArray[count] = weights[i];
            count++;
        }
    }
}

```

```
}

int resultsArraySize =
    (int) Math.pow(2, assessmentPointsEvaluated);
double[] results = new double[resultsArraySize];
for(int j = 0; j < bArray.length; j++) {
    double tempValue = 0;
    for(int k = 0; k < weightSelectionArray.length; k++) {
        tempValue = tempValue +
            (bArray[j][k] * weightSelectionArray[k]);
    }
    results[j] = tempValue;
}

Arrays.sort(results);

upperBound = results[results.length - 1];
middleQuartile = findMedian(results);

int arrayCenter = (int) results.length/2;
double[] lowerHalfArray = new double[arrayCenter];
double[] upperHalfArray = new double[arrayCenter];

if(results.length % 2 == 0) {
    for(int i = 0; i < arrayCenter; i++) {
        lowerHalfArray[i] = results[i];
        upperHalfArray[i] = results[arrayCenter + i];
    }
}
else {
    for(int i = 0; i < arrayCenter; i++) {
```

```
        lowerHalfArray [ i ] = results [ i ];
        upperHalfArray [ i ] = results [ ( arrayCenter + 1 ) + i ];
    }
}

lowerQuartile = findMedian ( lowerHalfArray );
upperQuartile = findMedian ( upperHalfArray );

normalisedLowerQuartile = ( lowerQuartile / upperBound ) * 100;
normalisedMiddleQuartile = ( middleQuartile / upperBound ) * 100;
normalisedUpperQuartile = ( upperQuartile / upperBound ) * 100;
}
```

C.4 Summary

This appendix provided a glimpse into the functionality of SADAC, the proof of concept digital forensic tool, by presenting and describing three key methods. These methods, which included the calculation of both the authenticity and completeness scores, as well as the construction of the authentic grading scale, demonstrate the formulation of the authenticity classification.

Bibliography

- Abadi, S. (2011), Towards a Generic Reference Architecture for Mobile Applications, PhD thesis, University of Gothenburg, Göteborg, Sweden.
- Abalenkovs, D., Bondarenko, P., Pathapati, V. K. and Nordbø, A. (2012), Mobile forensics: Comparison of extraction and analyzing methods of iOS and Android, Master's thesis, Gjøvik University College, Gjøvik, Norway.
- AccessData* (2017). Conquer Mobile Investigations with AccessData. Available from: [<http://www.accessdata.com/products-services/mobile-solutions>] (Accessed: 3 October 2017).
- Adams, C., Whitledge, A. and Sheno, S. (2008), Legal issues pertaining to the use of cell phone data, in 'Ray I., Sheno S. (eds.) *Advances in Digital Forensics IV*', Vol. 285, Springer, Boston, MA, pp. 231–243.
- Afonin, O. and Katalov, V. (2016), *Mobile Forensics - Advanced Investigative Strategies*, Packt Publishing Ltd, Birmingham, UK.
- Ahmad, M. S., Musa, N. E., Nadarajah, R. and Othman, N. E. (2013), Comparison between Android and iOS operating system in terms of security, in '8th International Conference on Information Technology in Asia (CITA), Kota Samarahan, Malaysia, 1-4 July', IEEE, pp. 1–4.
- Ahmed, R., Dharaskar, R. V. and Thakare, V. M. (2014), 'Forensic preservation of digital evidence on mobile devices from the perspective of efficient generalized forensics framework for mobile devices (EGFFMD)', *International Journal of Advanced Research in Computer Science* 5(4), 214–218.

- Al-Hadadi, M. and AlShidhani, A. (2013), ‘Smartphone forensics analysis: A case study’, *International Journal of Computer and Electrical Engineering* **5**(6), 576–580.
- Albano, P., Castiglione, A., Cattaneo, G., De Maio, G. and De Santis, A. (2011), On the construction of a false alibi on the Android OS, in ‘*Third International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, Fukuoka, Japan, 30 November-2 December’, IEEE, pp. 685–690.
- Albano, P., Castiglione, A., Cattaneo, G. and De Santis, A. (2011), A novel anti-forensics technique for the Android OS, in ‘*International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)*, Barcelona, Spain, 26-28 October’, IEEE, pp. 380–385.
- Alghaffi, K. A., Jones, A. and Martin, T. A. (2011), Guidelines for the digital forensic processing of smartphones, in ‘*Proceedings of the 9th Australian Digital Forensics Conference*, Perth, Western Australia, 5-7 December’, pp. 1–8.
- Aloul, F., Zahidi, S. and El-Hajj, W. (2009), Two factor authentication using mobile phones, in ‘*IEEE/ACS International Conference on Computer Systems and Applications*, Robat, Morocco, 10-13 May’, IEEE, pp. 641–644.
- Altshuler, Y., Fire, M., Aharony, N., Elovici, Y. and Pentland, A. S. (2012), How many makes a crowd? On the evolution of learning as a factor of community coverage, in ‘*Yang S. J., Greenberg A. M., Endsley M. (eds.) Social Computing, Behavioural - Cultural Modeling and Prediction*’, Vol. 7227, Springer, Berlin, Heidelberg, pp. 43–52.
- Android Developers* (2017). Platform Architecture. Available from: [<http://developer.android.com/guide/platform/index.html>] (Accessed: 4 October 2017).
- Android Developers* (2018a). Storage Options. Available from: [<http://developer.android.com/guide/topics/data/data-storage.html>] (Accessed: 7 January 2018).

- Android Developers* (2018b). SmsManager. Available from: [<http://developer.android.com/reference/android/telephony/SmsManager.html>] (Accessed: 7 January 2018).
- Android Developers* (2018c). Android debug bridge (adb). Available from: [<http://developer.android.com/studio/command-line/adb.html>] (Accessed: 7 January 2018).
- AndroidVulnerabilities.org* (2018). Available from: [www.androidvulnerabilities.org/index.html] (Accessed: 27 June 2018).
- Apple Developer* (2017a). About the iOS Technologies. Available from: [<https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html>] (Accessed: 29 September 2017).
- Apple Developer* (2017b). iOS Technology Overview. Available from: [<http://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html>] (Accessed: 5 October 2017).
- Ayers, R., Brothers, S. and Jansen, W. (2013), Guidelines on mobile device forensics (Draft), NIST Special Publication 800-101, Technical report, National Institute of Standards and Technology.
- Babin, S. (2008), *Developing Software for Symbian OS: A Beginner's Guide to Creating Symbian OS v9 Smartphone Applications in C++*, 2nd edn, John Wiley & Sons, West Sussex, England.
- Bader, M. and Baggili, I. (2010), 'iPhone 3GS forensics: Logical analysis using Apple iTunes backup utility', *Small scale digital device forensics journal* **4**(1), 1–15.
- Barmpatsalou, K., Cruz, T., Monteiro, E. and Simoes, P. (2018), 'Current and future trends in mobile device forensics: A survey', *ACM Computing Surveys* **51**(3), 1–31.

- Barmpatsalou, K., Damopoulos, D., Kambourakis, G. and Katos, V. (2013), ‘A critical review of 7 years of mobile device forensics’, *Digital Investigation* **10**(4), 323–349.
- Barrera, D. and Van Oorschot, P. (2011), ‘Secure software installation on smartphones’, *IEEE Security & Privacy* **9**(3), 42–48.
- Belenko, A. and Sklyarov, D. (2011), Evolution of iOS data protection and iPhone forensics: from iPhone OS to iOS5, in ‘*Blackhat Abu Dhabi Conference*, Abu Dhabi, United Arab Emirates, 14-15 December’.
- Bergey, J., Campbell, G., Clements, P., Cohen, S. and Jones, L. (1999), Second DOD product line practice workshop report, ESC-TR-99-015, Technical report, Carnegie Mellon University.
- Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, F. and Cowan, J. (2006), Extensible markup language (XML) 1.1 (second edition), Technical report, W3C.
- Brothers, S. (2009), ‘Cell phone and GPS forensic tool classification system’. Available from: [http://www.mobileforensicsworld.org/2009/presentations/MFW2009_BROTHERS_CellPhoneandGPSForensicToolClassificationSystem.pdf] (Accessed: 14 May 2018).
- Caithness, A. (2012), ‘The forensic implications of SQLite’s write ahead log’. CCL-Forensics. Available from: [<https://www.cclgrouppltd.com/the-forensic-implications-of-sqlites-write-ahead-log/>] (Accessed: 10 October 2017).
- Casey, E. (2011), *Digital evidence and computer crime: Forensic science, computers, and the Internet*, 3rd edn, Academic Press, Cambridge, Massachusetts, USA.
- Casey, E. (2013), Reinforcing the scientific method in digital investigations using a case-based reasoning (CBR) system, PhD thesis, University College Dublin, Dublin, Ireland.

- Cellebrite* (2017). Cellebrite - Mobile Forensic Solutions. Available from: [<http://www.cellebrite.com/Mobile-Forensics/Solutions/mobile-forensics-solutions>] (Accessed: 3 October 2017).
- Cheema, A. R., Iqbal, M. M. W. and Ali, W. (2014), An open source toolkit for iOS filesystem forensics, in ‘*Peterson G., Sheno, S. (eds.) Advances in Digital Forensics X*’, Vol. 433, Springer, Berlin, Heidelberg, pp. 227–235.
- Chin, E., Felt, A. P., Greenwood, K. and Wagner, D. (2011), Analyzing inter-application communication in Android, in ‘*Proceedings of the 9th international conference on mobile systems, applications, and services*, Bethesda, Maryland, USA, 26 June-1 July’, ACM, pp. 239–252.
- Chittaranjan, G., Blom, J. and Gatica-Perez, D. (2013), ‘Mining large-scale smartphone data for personality studies’, *Personal and Ubiquitous Computing* **17**(3), 433–450.
- Choi, J. and Lee, S. (2016), ‘A study of user relationships in smartphone forensics’, *Multimedia Tools and Applications* **77**(22), 14971–14983.
- Cohen, F. B. (2012), *Digital forensic evidence examination*, 5th edn, Fred Cohen & Associates, Livermore, CA, USA.
- Conlan, K., Baggili, I. and Breitingner, F. (2016), ‘Anti-forensics: Furthering digital forensic science through a new extended, granular taxonomy.’, *Digital Investigation* **18**, 66–75.
- Curran, K., Robinson, A., Peacocke, S. and Cassidy, S. (2010), ‘Mobile phone forensic analysis’, *International Journal of Digital Crime and Forensics* **2**(2), 15–27.
- Dancer, F. C., Dampier, D. A., Jackson, J. M. and Meghanathan, N. (2013), A theoretical process model for smartphones, in ‘*Meghanathan N. Nagamalai D., Chaki N. (eds.) Advances in Computing and Information Technology*’, Vol. 178, Springer, Berlin, Heidelberg, pp. 279–290.
- Danielsson, A. (2017), Comparing Android runtime with native: Fast Fourier transform on Android, Master’s thesis, KTH - Royal Institute of Technology, Stockholm, Sweden.

- Diniz, H. B. M., Silva, E. C. G. F., Nogueira, T. C. C. and Gama, K. (2016), A reference architecture for mobile crowdsensing platforms, in *'Proceedings of the 18th International Conference on Enterprise Information Systems, Rome, Italy, 25-28 April'*, Vol. 2, SCITEPRESS - Science and Technology Publications, pp. 600–607.
- Distefano, A., Me, G. and Pace, F. (2010), 'Android anti-forensics through a local paradigm', *Digital Investigation* **7**, 83–94.
- Dlamini, Z. I., Olivier, M. S. and Grobler, M. M. (2016), The smartphone evidence awareness framework for the users, in *'Proceedings of the 11th International Conference on Cyber Warfare and Security, Boston, USA, 17-18 March'*, pp. 439–449.
- Do, T. M. and Gatica-Perez, D. (2011), GroupUs: Smartphone proximity data and human interaction type mining, in *'15th Annual International Symposium on Wearable Computers (ISWC), San Francisco, CA, USA, 12-15 June'*, IEEE, pp. 21–28.
- Do, T. M. and Gatica-Perez, D. (2014), 'The places of our lives: Visiting patterns and automatic labeling from longitudinal smartphone data', *IEEE Transactions on Mobile Computing* **13**(3), 638–648.
- D'Orazio, C. J. (2013), iOS forensics, Master's thesis, University of South Australia, Adelaide, Australia.
- Drake, J. J., Lanier, Z., Mulliner, C., Fora, P. O., Ridley, S. A. and Wicherski, G. (2014), *Android hacker's handbook*, John Wiley & Sons, Indianapolis, Indiana, USA.
- Duranti, L. (2009), 'From digital diplomatics to digital records forensics', *Archivaria, The Journal of the Association of Canadian Archivists* **68**, 39–66.
- Duranti, L. and Endicott-Popovsky, B. (2010), 'Digital records forensics: A new science and academic program for forensic readiness', *Journal of Digital Forensics, Security and Law* **5**(2), 45–62.
- Eagle, N. and Pentland, A. S. (2006), 'Reality mining: sensing complex social systems', *Personal and Ubiquitous Computing* **10**(4), 255–268.

- Egele, M., Kruegel, C., Kirda, E. and Vigna, G. (2011), PiOS: Detecting privacy leaks in iOS applications, in ‘*Proceedings of the Network and Distributed System Security Symposium, NDSS*, San Diego, California, USA, 6-9 February’, pp. 177–183.
- Eixelsberger, W., Ogris, M., Gall, H. and Bellay, B. (1998), Software architecture recovery of a program family, in ‘*Proceedings of the 20th International Conference on Software Engineering*, Kyoto, Japan, 19-25 April’, IEEE Computer Society, pp. 508–511.
- Elcomsoft* (2017). Elcomsoft Mobile Forensic Bundle. Available from: [<http://www.elcomsoft.com/emfb.html>] (Accessed: 3 October 2017).
- Epifani, M. and Stirparo, P. (2016), *Learning iOS Forensics*, 2nd edn, Packt Publishing Ltd, Birmingham, UK.
- Epp, S. S. (2004), *Discrete Mathematics with Applications*, 3rd edn, Brooks/Cole - Thomson Learning, Inc., Belmont, CA, USA.
- Farjamfar, A., Abdullah, M. T., Mahmud, R. and Udzir, N. I. (2014), ‘A review on mobile device’s digital forensic process models’, *Research Journal of Applied Sciences, Engineering and Technology* **8**(3), 358–366.
- Farley, T. (2005), ‘Mobile telephone history’, *Teletronikk* **101**(3/4), 22–34.
- Freiling, F., Spreitzenbarth, M. and Schmitt, S. (2011), ‘Forensic analysis of smartphones: The Android Data Extractor Lite (ADEL)’, *Annual ADFSL Conference on Digital Forensics, Security and Law* **5**, 151–160.
- Gadyatskaya, O., Massacci, F. and Zhauniarovich, Y. (2014), ‘Security in the Firefox OS and Tizen mobile platforms’, *Computer* **47**(6), 57–63.
- Garfinkel, S. (2007), Anti-forensics: Techniques, detection and countermeasures, in ‘*2nd International Conference on i-Warfare and Security*, Monterey, California, USA, 8-9 March’, Academic Conferences Limited, pp. 77–84.

- Gartner (2018). Gartner Says Worldwide Device Shipments Will Increase 2.1 Percent in 2018. Available from: [<http://www.gartner.com/newsroom/id/3849063>] (Accessed: 20 April 2018).
- Geng, G., Xu, G., Zhang, M., Guo, Y., Yang, G. and Wei, C. (2012), 'The design of SMS based heterogeneous mobile botnet', *Journal of Computers* **7**(1), 235–243.
- Goadrich, M. H. and Rogers, M. P. (2011), Smart smartphone development: iOS versus Android, in 'Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, Dallas, Texas, USA, 9-12 March', ACM, pp. 607–612.
- Goel, A., Tyagi, A. and Agarwal, A. (2012), 'Smartphone forensic investigation process model', *International Journal of Computer Science & Security (IJCSS)* **6**(5), 322–341.
- Gomez-Miralles, L. and Arnedo-Moreno, J. (2011), Universal, fast method for iPad forensics imaging via USB adapter, in 'Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Seoul, South Korea, 30 June - 2 July', IEEE, pp. 200–207.
- Govindaraj, J., Verma, R., Mata, R. and Gupta, G. (2015), Forensic ready secure iOS apps for jailbroken iPhones, in 'Peterson G., Shenoi S. (eds.) Advances in Digital Forensics XI', Vol. 462, Springer, Cham, pp. 235–249.
- Gronli, T. M., Hansen, J., Ghinea, G. and Younas, M. (2014), Mobile application platform heterogeneity: Android vs Windows Phone vs iOS vs Firefox OS, in '28th International Conference on Advanced Information Networking and Applications (AINA), Victoria, BC, Canada, 13-16 May', IEEE, pp. 635–641.
- Grosskurth, A. and Godfrey, M. W. (2005), A reference architecture for web browsers, in 'Proceedings of the 21st IEEE International Conference on Software Maintenance, Budapest, Hungary, 26-29 September', IEEE, pp. 661–664.
- Grover, J. (2013), 'Android forensics: Automated data collection and reporting from a mobile device', *Digital Investigation* **10**, 12–20.

- Halsey, M. (2015), *Beginning Windows 10: Do More with Your PC*, Apress, New York City, NY, USA.
- Hanna, R., Rohm, A. and Crittenden, V. L. (2011), 'We're all connected: The power of the social media ecosystem', *Business horizons* **54**(3), 265–273.
- Hannon, M. J. (2014), 'An increasingly important requirement: Authentication of digital evidence', *Journal of the Missouri Bar* **70**(6), 314–323.
- Harris, R. (2006), 'Arriving at an anti-forensics consensus: Examining how to define and control the anti-forensics problem', *Digital Investigation* **3**, 44–49.
- Hasan, M. and Haque, A. (2016), 'Mobile application development approaches: Recommendation for E-commerce enterprises'.
- Hassan, A. E. and Holt, R. C. (2000), A reference architecture for web servers, in 'Seventh Working Conference on Reverse Engineering, Brisbane, Australia, 25 November', IEEE, pp. 150–159.
- Hoelz, B. and Maues, M. (2017), Anti-forensics threat modeling, in 'Peterson G., Shenoii S. (eds) *Advances in Digital Forensics XIII*', Vol. 511, Springer, Berlin, Heidelberg, pp. 169–183.
- Holzer, A. and Ondrus, J. (2011), 'Mobile application market: A developers perspective', *Telematics and Informatics* **28**(1), 22–31.
- Hoog, A. (2011), *Android Forensics: Investigation, Analysis and Mobile Security for Google Android*, Syngress, Waltham, MA, USA.
- Hoog, A. and Strzempka, K. (2011), *iPhone and iOS forensics: Investigation, analysis and mobile security for Apple iPhone, iPad and iOS devices*, Syngress, Waltham, MA, USA.
- Husain, M. I., Baggili, I. and Sridhar, R. (2010), A simple cost-effective framework for iPhone forensic analysis, in 'Baggili I. (ed.) *Digital Forensics and Cyber Crime*', Vol. 53, Springer, Berlin, Heidelberg, pp. 27–37.

- Iqbal, B., Iqbal, A. and Al Obaidli, H. (2012), A novel method of iDevice (iPhone, iPad, iPod) forensics without jailbreaking, in '*International Conference on Innovations in Information Technology (IIT)*', Abu Dhabi, United Arab Emirates, 18-20 March', IEEE, pp. 238–243.
- Iulia-Maria, T. and Ciocarlie, H. (2011), Best practices in iPhone programming: Model-view-control architecture - Carousel component development, in '*EUROCON - International Conference on Computer as a Tool*', Lisbon, Portugal, 27-29 April', IEEE, pp. 1–4.
- Jacobs, B. (2015), 'iOS from scratch with swift: data persistence and sandboxing iOS'. Envato. Available from: [<https://code.tutsplus.com/tutorials/ios-from-scratch-with-swift-data-persistence-and-sandboxing-on-ios--cms-25505>] (Accessed: 12 October 2017).
- Jansen, W. and Ayers, R. (2007), Guidelines on cell phone forensics, NIST Special Publication 800-101, Technical report, National Institute of Standards and Technology.
- Jeon, S., Bang, J., Byun, K. and Lee, S. (2012), 'A recovery method of deleted record for SQLite database', *Personal and Ubiquitous Computing* **16**(6), 707–715.
- Jeon, W., Kim, J., Lee, Y. and Won, D. (2011), A practical analysis of smartphone security, in '*Smith M. J., Salvendy G. (eds.) Human Interface and the Management of Information*', Vol. 6771, Springer, Berlin, Heidelberg, pp. 311–320.
- Joorabchi, M. E. and Mesbah, A. (2012), Reverse engineering iOS mobile applications, in '*19th Working Conference on Reverse Engineering*', Kingston, ON, Canada, 15-18 October', IEEE, pp. 177–186.
- Kala, N. and Thilagaraj, R. (2013), 'A framework for digital forensics in i-devices: Jailed and jail broken devices', *Journal of Advances in Library and Information Science* **2**(2), 82–93.
- Kanoi, P. V. and Ingole, P. N. (2013), 'Internal structure of iOS and building tools for iOS apps', *International Journal of Computer Science and Applications* **6**(2), 220–225.

- Karlsson, K. J. and Glisson, W. B. (2014), Android anti-forensics: Modifying CyanogenMod, in ‘*47th Hawaii International Conference on System Sciences (HICSS)*, Waikoloa, HI, USA, 6-9 January’, IEEE, pp. 4828–4837.
- Kessler, G. C. (2007), Anti-forensics and the digital investigator, in ‘*Proceedings of the 5th Australian Digital Forensics Conference*, Edith Cowan University’, pp. 1–7.
- Kim, H.-J. and Kim, J.-S. (2012), Tuning the EXT4 filesystem performance for Android-based smartphones, in ‘*Sambath S., Zhu E. (eds.) Frontiers in Computer Education*’, Vol. 133, Springer, Berlin, Heidelberg, pp. 745–752.
- Köhn, M. D. (2012), Integrated digital forensic process model, Master’s thesis, University of Pretoria, Pretoria, South Africa.
- Kotze, D. (2015), XML accounting trail: A model for introducing forensic readiness to XML accounting and XBRL, Master’s thesis, University of Pretoria, Pretoria, South Africa.
- Kubi, A. K., Saleem, S. and Popov, O. (2011), Evaluation of some tools for extracting e-evidence from mobile devices, in ‘*5th International Conference on Application of Information and Communication Technologies (AICT)*, Baku, Azerbaijan, 12-14 October’, IEEE, pp. 1–6.
- La Polla, M., Martinelli, F. and Sgandurra, D. (2013), ‘A survey on security for mobile devices’, *IEEE Communications Surveys & Tutorials* **15**(1), 446–471.
- Lee, S., Savoldi, A., Lim, K. S., Park, J. H. and Lee, S. (2010), ‘A proposal for automating investigations in live forensics’, *Computer Standards & Interfaces* **32**(5-6), 246–255.
- Lee, X., Yang, C. H., Chen, S. and Wu, J. (2009), Design and implementation of forensic system in Android smartphone, in ‘*The 5th Joint Workshop on Information Security*, Guangzhou, China’, pp. 1–11.
- Lessard, J. and Kessler, G. C. (2010), ‘Android forensics: Simplifying cell phone examinations’, *Small Scale Digital Device Forensics Journal* **4**(1), 1–12.

- Lewis, G., Novakouski, M. and Sánchez, E. (2012), A reference architecture for group-context-aware mobile applications, in ‘Uhler D., Mehta K., Wong J. L. (eds.) *MobiCASE 2012: Mobile Computing, Applications, and Services*’, Vol. 110, Springer, Berlin, Heidelberg, pp. 44–63.
- Lin, F. and Ye, W. (2009), Operating system battle in the ecosystem of smartphone industry, in ‘*International Symposium on Information Engineering and Electronic Commerce*, Ternopil, Ukraine, 16-17 May’, IEEE, pp. 617–621.
- Lin, I. L., Chao, H. C. and Peng, S. H. (2011), Research of digital evidence forensics standard operating procedure with comparison and analysis based on smart phone, in ‘*2011 International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)*, Barcelona, Spain, 26-28 October’, IEEE, pp. 386–391.
- Lohiya, R., John, P. and Shah, P. (2015), ‘Survey on mobile forensics’, *The International Journal of Computer Applications* **118**(16), 6–11.
- Losavio, M. (2005), Non-technical manipulation of digital data, in ‘Pollitt M., Shenoit S. (eds) *Advances in Digital Forensics*’, Vol. 194, Springer, Boston, pp. 51–63.
- Luttenberger, S. and Creutzburg, R. (2011), Forensic investigation of certain types of mobile devices, in ‘*Proc. SPIE 7881, Multimedia on Mobile Devices 2011; and Multimedia Content Access: Algorithms and Systems V*, San Francisco Airport, California, USA, 23-27 January’, pp. 1–12.
- Managing Jailbreak Threats on iOS* (2016), Technical report, Lockout, Inc.
- Mark, D. and LaMarche, J. (2009), *More iPhone 3 development - Tackling iPhone SDK3*, Apress, New York City, NY, USA.
- Martini, B., Do, Q. and Choo, K.-K. R. (2015), Conceptual evidence collection and analysis methodology for Android devices, in ‘Ko R., Choo, K.-K. R. (eds.) *Cloud Security Ecosystem*’, Syngress, an Imprint of Elsevier, pp. 285–307.

- Maus, S., Höfken, H. and Schuba, M. (2011), Forensic analysis of geodata in Android smartphones, in ‘*International Conference on Cybercrime, Security and Digital Forensics*’, pp. 1–11.
- Meier, R. (2012), *Professional Android 4 Application Development*, John Wiley & Sons, Indianapolis, Indiana, USA.
- Mercaldo, F., Nardone, V., Santone, A. and Visaggio, C. A. (2016), Ransomware steals your phone. Formal methods rescue it., in ‘*Albert E., Lanese I. (eds.) Formal Techniques for Distributed Objects, Components, and Systems, FORTE 2016*’, Vol. 9688, Springer, Cham, pp. 212–221.
- Miller, C. (2011), ‘Mobile attacks and defense’, *IEEE Security & Privacy* **9**(4), 68–70.
- Min, J. K. and Cho, S. B. (2011), ‘Mobile human network management and recommendation by probabilistic social mining’, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **41**(3), 761–771.
- Min, J. K., Wiese, J., Hong, J. I. and Zimmerman, J. (2013), Mining smartphone data to classify life-facets of social relationships, in ‘*Proceedings of the 2013 conference on Computer Supported Cooperative Work*, San Antonio, Texas, USA, 23-27 February’, ACM, pp. 285–294.
- MOBILedit* (2017). MOBILedit Forensic. Available from: [<http://www.mobiledit.com/mobiledit-forensic>] (Accessed: 3 October 2017).
- Mohtasebi, S. H. and Dehghantanha, A. (2013), ‘Towards a unified forensic investigation framework of smartphones’, *The International Journal of Computer Theory and Engineering* **5**(2), 351–355.
- Mokhonoana, P. M. and Olivier, M. S. (2007), Acquisition of a Symbian smart phone’s content with an on-phone forensic tool, in ‘*Proceedings of the Southern African Telecommunication Networks and Applications Conference*, Sugar Beach Resort, Mauritius, 9-13 September’, pp. 1–6.

- Morrissey, S. (2010), *iOS forensic analysis for iPhone, iPad, and iPod Touch*, Apress, New York City, NY, USA.
- MSAB (2017). What is XRY?. Available from: [http://www.msab.com/download/product_sheets/en/What_is_XRY.pdf] (Accessed: 3 October 2017).
- Mumba, E. R. and Venter, H. S. (2012), Harmonised digital forensic investigation process model, in ‘*Information Security for South Africa (ISSA)*, Johannesburg, South Africa, 15-17 August’, IEEE, pp. 1–10.
- Mylonas, A., Meletiadiis, V., Mitrou, L. and Gritzalis, D. (2013), ‘Smartphone sensor data as digital evidence’, *Computer & Security* **38**, 51–75.
- Mylonas, A., Meletiadiis, V., Tsoumas, B., Mitrou, L. and Gritzalis, D. (2012), Smartphone forensics: A proactive investigation scheme for evidence acquisition, in ‘*Gritzalis D., Furnell S., Theoharidou M. (eds.) Information Security and Privacy Research, SEC 2012*’, Vol. 376, Springer, Berlin, Heidelberg, pp. 249–260.
- NetMarketShare (2018). Operating System Market Share. Available from: [<https://netmarketshare.com/operating-system-market-share.aspx>] (Accessed: 4 June 2018).
- Nosrati, M., Karimi, R. and Hasanvand, H. A. (2012), ‘Mobile computing: Principles, devices and operating systems’, *World Applied Programming* **2**(7), 399–408.
- Omeleze, S. and Venter, H. S. (2013), Testing the harmonised digital forensic investigation process model-using an Android mobile phone, in ‘*Information Security for South Africa (ISSA)*, Johannesburg, South Africa, 14-16 August’, IEEE, pp. 1–8.
- Osho, O. and Ohida, S. O. (2016), ‘Comparative evaluation of mobile forensic tools’, *International Journal of Information Technology and Computer Science (IJITCS)* **8**(1), 74–83.
- Oxygen Forensics (2017). Oxygen Forensics - Analyst Features. Available from: [<http://www.oxygen-forensic.com/en/products/oxygen-forensic-analyst>] (Accessed: 3 October 2017).

- Padhya, B., Desai, P. and Pawade, D. (2016), 'Comparison of mobile operating systems', *International Journal of Innovative Research in Computer and Communication Engineering* **4**(8), 15281–15286.
- Paraben (2017). Paraben's E3 DS Mobile Device Examination. Available online: [<http://www.paraben.com/producrcs/e3-ds>] (Accessed: 3 October 2017).
- Park, J., Chung, H. and Lee, S. (2012), 'Forensic analysis techniques for fragmented flash memory pages in smartphones', *Digital Investigation* **9**(2), 109–118.
- Park, J. H., Kim, D., Park, J. S. and Lee, S. (2016), 'An enhanced security framework for reliable Android operating system', *Security and Communication Networks* **9**(6), 528–534.
- Parvez, S., Dehghantanha, A. and Broujerdi, H. G. (2011), Framework of digital forensics for the Samsung Star Series phone, in '3rd International Conference on Electronics Computer Technology (ICECT), Kanyakumari, India, 8-10 April', IEEE, pp. 264–267.
- Patodi, P. (2012), Database recovery mechanism for Android devices, Master's thesis, Indian Institute of Technology, Bombay, India.
- Pfleeger, C. P. and Pfleeger, S. L. (2007), *Security in Computing*, 4th edn, Pearson Education, Inc., London, England.
- Pieterse, H., Olivier, M. S. and van Heerden, R. P. (2015), Playing hide-and-seek: Detecting the manipulation of Android timestamps, in 'Information Security for South Africa (ISSA), Johannesburg, South Africa, 12-13 August', IEEE, pp. 1–8.
- Pollitt, M., Casey, E., Jaquet-Chiffelle, D.-O. and Gladyshev, P. (2018), A framework for harmonizing forensic science practices and digital/multimedia evidence, Technical report, OSAC Technical Series 0002R1.
- Punja, S. G. and Mislán, R. P. (2008), 'Mobile device analysis', *Small scale digital device forensics journal* **2**(1), 1–16.

- Quick, D. and Alzaabi, M. (2011), Forensic analysis of the Android file system YAFFS2, in ‘*Proceedings of the 9th Australian Digital Forensics Conference*, Edith Cowan University, Perth, Western Australia’, pp. 100–109.
- Quick, D. and Choo, K. K. R. (2016), ‘Big forensics data reduction: Digital forensic images and electronic evidence’, *Cluster Computing* **19**(2), 723–740.
- Ramabhadran, A. (2007), Forensic investigation process model for Windows mobile devices, Technical report, Security Group - Tata Elxsi.
- Richardson, R. and North, M. (2017), ‘Ransomware: Evolution, mitigation and prevention’, *International Management Review* **13**(1), 10–21.
- Sadun, E. (2012), *The iOS 5 Developer’s Cookbook: Core Concepts and Essential Recipes for iOS programmers*, 3rd edn, Addison-Wesley, Boston, USA.
- Schneier, B. (1999), ‘Attack trees’, *Dr. Dobb’s Journal* **24**(12), 21–29.
- Schölzel, M., Eren, E. and Detken, K. O. (2015), A viable SIEM approach for Android, in ‘*8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems*, Warsaw, Poland, 24-26 September’, pp. 803–807.
- Schölzel, M., Eren, E., Detken, K. O. and Schwenke, L. (2016), ‘Monitoring Android devices by using events and metadata’, *International Journal of Computing* **15**(4), 248–258.
- Shaw, M. and Garlan, D. (1996), *Software architecture: perspectives on an emerging discipline*, Prentice Hall, Upper Saddle River, New Jersey, USA.
- Shen, Z. and Ma, K. L. (2008), MobiVis: A visualization system for exploring mobile data, in ‘*Visualization Symposium, PacificVIS’08*, Kyoto, Japan, 5-7 March’, IEEE, pp. 175–182.
- Simão, A. M. D. L., Sícoli, F. C., Melo, L. P. D., Deus, F. E. G. D. and Sousa Júnior, R. T. D. (2011), ‘Acquisition and analysis of digital evidence in Android smartphones’, *The International Journal of Forensic Computer Science* **6**(1), 28–43.

- Simanta, S., Ha, K., Lewis, G., Morris, E. and Satyanarayana, M. (2012), A reference architecture for mobile code offload in hostile environments, in ‘Uhler D., Mehta K., Wong J. L. (eds.) *MobiCASE 2012: Mobile Computing, Applications, and Services*’, Vol. 110, Springer, Berlin, Heidelberg, pp. 274–293.
- Sokolova, K., Lemercier, M. and Garcia, L. (2014), ‘Towards high quality mobile applications: Android passive MVC architecture’, *International Journal on Advances in Software* **7**(1-2), 123–138.
- Speckmann, B. (2008), The android mobile platform, Master’s thesis, Eastern Michigan University, Ypsilanti, Michigan, USA.
- Sporea, I., Aziz, B. and McIntyre, Z. (2012), ‘On the availability of anti-forensic tools for smartphones’, *International Journal of Security (IJS)* **6**(4), 58–64.
- SQLite* (2017a). Database File Format. Available from: [<http://www.sqlite.org/fileformat.html>] (Accessed: 10 October 2017).
- SQLite* (2017b). About SQLite. Available from: [<http://www.sqlite.org/about.html>] (Accessed: 10 October 2017).
- SQLite* (2017c). Write-Ahead Logging. Available from: [<http://www.sqlite.org/wal.html>] (Accessed: 10 October 2017).
- SQLite* (2017d). Command Line Shell for SQLite. Available from: [<http://www.sqlite.org/cli.html>] (Accessed: 10 October 2017).
- Steele, J. and To, N. (2011), *The Android Developer’s Cookbook - Building Applications with the Android SDK*, Pearson Education, Inc., Boston, MA, USA.
- Tamma, R. and Tindall, D. (2015), *Learning Android Forensics*, Packt Publishing Ltd, Birmingham, UK.
- Tamura, E. and Giampaolo, D. (2016), Introducing Apple file system, Technical report, Apple, Inc.

- Tanenbaum, A. S. (2009), *Modern operating systems*, Pearson Education, Inc., London, England, UK.
- Thomson, L. L. (2013), ‘Mobile devices: New challenges for admissibility of electronic evidence’, *SciTech Lawyer* **9**(3), 32–37.
- Tidrow, R., Boyce, J. and Shapiro, J. R. (2015), *Windows 10 Bible*, John Wiley & Sons, Indianapolis, Indiana, USA.
- Tracy, K. W. (2012), ‘Mobile application development experiences on Apple’s iOS and Android OS’, *IEEE Potentials* **31**(4), 30–34.
- Tsavli, M., Efraimidis, P. and Katos, V. (2015), ‘Reengineering the user: privacy concerns about personal data on smartphones’, *Information & Computer Security* **23**(4), 394–405.
- Vaibhav Kumar, S. (2013), Customization of Android and performance analysis of Android applications in different environments, Master’s thesis, Computer Science and Engineering Department, Thapar University, Patiala, India.
- Valjarevic, A. and Venter, H. S. (2012), Harmonised digital forensic investigation process model, in ‘*Information Security for South Africa (ISSA)*, Johannesburg, South Africa, 15-17 August’, IEEE, pp. 1–10.
- Vashisht, G. and Vashisht, R. (2014), ‘A study on the Tizen operating system’, *International Journal of Computer Trends and Technology* **12**(1), 14–15.
- Vaughan-Nicholas, S. J. (2003), ‘OSs battle in the smart-phone market’, *Computer* **36**(6), 10–12.
- Verma, R., Govindaraj, J. and Gupta, G. (2014), Preserving dates and timestamps for incident handling in Android smartphones, in ‘*Peterson G., Sheno S. (eds) Advances in Digital Forensics X*’, Vol. 433, Springer, Berlin, Heidelberg, pp. 209–225.
- Vidas, T., Zhang, C. and Christin, N. (2011), ‘Toward a general collection methodology for Android devices’, *Digital Investigation* **8**, 14–24.

- Weiss, G. M. and Lockhart, J. W. (2011), Identifying user traits by mining smart phone accelerometer data, in ‘*Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data*, San Diego, California, USA, 21 August’, ACM, pp. 61–69.
- Yang, C.-H. and Lai, Y.-T. (2012), ‘Design and implementation of forensic system for Android devices based on cloud computing’, *Applied Mathematics & Information Sciences* **6**(1), 243–247.
- Yates, I. (2010), Practical investigations of digital forensics tools for mobile devices, in ‘*Information Security Curriculum Development Conference*, Kennesaw, Georgia, 1-3 October’, ACM, pp. 156–162.
- Yu, X., Jiang, L., Shu, H., Yin, Q. and Liu, T. (2009), A process model for forensic analysis of symbian smart phones, in ‘*Ślęzak D., Kim, T., Kiumi A., Jiang T., Verner J., Abrahão S.(eds.) Advances in Software Engineering*’, Vol. 59, Springer, Berlin, Heidelberg, pp. 86–93.
- Yun, J. J., Won, D. and Park, K. (2016), ‘Dynamics from open innovation to evolutionary change’, *Journal of Open Innovation: Technology, Market, and Complexity* **2**(1), 7.
- Zdziarski, J. (2008), *iPhone forensics: recovering evidence, personal data, and corporate assets*, O’Reilly Media, Inc., Sebastopol, CA, USA.
- Zhang, X., Jeong, S., Kunjithapatham, A. and Gibbs, S. (2010), Towards an elastic application model for augmenting computing capabilities of mobile platforms, in ‘*Cai Y., Magedanz T., Li M., Xia J., Giannelli C. (eds.) Mobile Wireless Middleware, Operating Systems, and Applications, MOBILWARE 2010*’, Vol. 48, Springer, Berlin, Heidelberg, pp. 161–174.
- Zheng, M., Mingshen, S. and Lui, J. (2014), DroidRay: A security evaluation system for customized Android firmwares, in ‘*Proceedings of the 9th ACM symposium on information, computer and communications security*, Kyoto, Japan, 4-6 June 2014’, ACM, pp. 471–482.

Zimmermann, C., Spreitzenbarth, M., Schmitt, S. and Freiling, F. C. (2012), Forensic analysis of YAFFS2, in 'Suri N., Waidner M. (eds.) *Sicherheit 2012*', Vol. 195, pp. 59–69.