Digital Forensic Readiness Architecture for Cloud Computing Systems

by

Dirk J. Ras

Submitted in fulfilment of the requirements for the degree

Magister Scientiae (Computer Science)
in the Faculty Engineering, Built Environment and Information Technology, University of Pretoria

Information and Computer Security Architecture Research Group
Department of Computer Science
University of Pretoria
South Africa

August, 2018

Every contact leaves a trace

Locard's exchange principle

Digital Forensic Readiness Architecture for Cloud Computing Systems

by

Dirk J. Ras

Submitted for the degree of Magister Scientiae (Computer Science)
August, 2018

Abstract

Cloud computing underpins many of the current emergent and established technologies. As a result, cloud computing has an impact on many components of our daily lives, be it from online shopping and banking to usage of mobile apps. Because of this ubiquity, crime related to cloud systems is an ongoing concern. There are, however, many factors that, while enabling cloud systems to function, also make digital forensic investigations on such systems very challenging. While processes and standards are defined for digital forensics, these processes often do not work when applied to cloud systems. Forensic investigations are, by their nature, very disruptive to the operation of a system. This is often unacceptable in a cloud environment. One way to mitigate the risk of a forensic investigation is to proactively prepare for such an event by achieving forensic readiness. This leads to the research conducted for this dissertation.

The central question is whether it possible to achieve forensic readiness in a cloud environment, so that a digital forensic investigation can be conducted with minimal or no disruption to the operation of said cloud environment.

This dissertation examines the background information of cloud computing, digital forensics and software architecture in order to get a clear understanding of the various research domains. Five possible models for the acquisition of data in a cloud environment are proposed, using the NIST cloud reference architecture as a baseline. A full, technology neutral, architecture for a cloud forensics system is then generated. This architecture allows for the acquisition of forensic data within a cloud environment. The architecture ensures that the data is kept forensically stable and enables the proactive analysis of the captured data.

Using one of the acquisition models, a proof of concept implementation is done of the architecture. Experiments are run to determine whether the system meets the set functional requirements and quality attributes to enable forensic readiness in a cloud system. The architecture and implementation are evaluated against the experimental results and possible improvements are suggested. The research is then concluded and possible future avenues of research in the field of cloud forensics are suggested.

Keywords: Cloud computing, digital forensics, cloud forensics

Supervisor	**:**	Prof. H.S. Venter
Department	**:**	Department of Computer Science
Degree	**:**	Magister Scientiae

Plagiarism Declaration

Name	Dirk Jacobus Ras
Student number	23148978
Topic of Work	Digital Forensic Readiness Architecture for Cloud Computing Systems

1. I understand what plagiarism is and am aware of the University's policy in this regard.

2. I declare that this dissertation is my own original work. Where other people's work has been used (either from a printed source, Internet or any other source), this has been properly acknowledged and referenced in accordance with the requirements as stated in the University's plagiarism prevention policy.

3. I have not used another student's past written work to hand in as my own.

4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.

Signature

Acknowledgements

I would like to thank the following people and organisations that helped in making this study possible.

- Professor H.S. Venter for his support, guidance and wisdom in navigating the world of academics.
- My long suffering parents, who always supported me.
- My mother-in-law, Ina, who came to my rescue at a critical time.
- My friends Andrich van Wyk, Waldo Delport and Leandi Steyn-Delport for always reminding me that they were already done with their degrees and that I should get a move-on or face further mocking.
- My colleagues at my places of work at SAP Africa, University of Pretoria and Dariel Solutions.
- Finally, my wife Aret, who actually married, and stayed married to me, while I spent the long nights working on this. Also, for supplying me copious amounts of tea, coffee, and necessary amounts whiskey.

Thank you to the National Research Foundation for funding part of this study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

In the early 1980s, computers with internal memory were still large cumbersome behemoths [3], with minimal storage capacity and tightly controlled access. During this time, the age of the personal computer dawned and computers found their way from the basements of large corporations and universities into the homes and offices of ordinary people.

When the Internet became available to the general public during the mid 1990s, the personal computer became a networked device with access to a large volume of information. From the time when personal computers found their way into the home and office, to them being connected to the Internet, the power of computer hardware increased dramatically. From the humble IBM PC, launched in 1981 [4], to the Pentium 4, launched in 2001 [5], the CPU speed went from 4.77MHz to 1.4GHz. Similarly, hard drive capacity rocketed from about 10MB to 20GB at the same time. As of 2017, CPUs with multiple cores and high capacity solid state hard disk drives have become commonplace.

With this powerful hardware, it became possible to run computers within computers. The concept of virtualization became a reality. However, single powerful computers running a few virtual machines could not be scaled up. The answer to this dilemma was the concept of cluster computing. By clustering a large number of computers together, it becomes possible to run numerous virtual machines. Add to this a model for dynamic resource allocation and self scaling, and the concept of cloud computing becomes a reality.

With the proliferation of computers came the rapid growth in computer crime. These crimes include anything from computer viruses to denial of service attacks, and many others. Some way of investigating these crimes was therefore needed. Computer forensics is a means of investigation and could provide a solution to the problem of computer crime.

However, at the best of times, computer forensics is an arduous process when investigating a single device. When an investigation must be performed on a massive computer system, such as a cloud computing system, it becomes a truly daunting task, both in terms of the time taken to conduct an investigation, and the logistics involved to capture large volumes of data [6–9]. For cloud computing systems, traditional computer forensics methods will not be sufficient to conduct a timely and accurate investigation [10].

Cloud computing systems have several traits that allow them to operate successfully. These traits, however, are also the cause of challenges with digital forensics in cloud systems. These general problems are [6, 8, 11]:

- Scale of cloud systems: Cloud computing systems can range in size from small minimal node systems, to vast commercial and corporate cloud systems. These corporate systems can easily span hundreds of physical nodes, the physical scale of which makes the systems hard to manage in a forensic investigation.

- Volume of data: Combined with the physical size of the systems, comes the problem of volume of data. Since the advent of high capacity hard disk drives, data centres with petabytes of storage have become commonplace. Having to capture and analyse all this data makes cloud forensics a practically daunting task. It is therefore required that the relevant data be isolated somewhere in the cloud system.

- Isolation of relevant data: As mentioned above, with the vast volumes of data, it is difficult to isolate the relevant data. This is due to the fact that data can migrate through the cloud system depending on its architecture. The data relevant to the cloud forensic investigation must therefore be isolated in some way.

- Integrity of captured data: The isolation and capturing of the forensic data create a problem in itself, as the integrity of the data must be maintained. Failure to maintain the integrity of the data could render the data inadmissible in a court of law or damage the data, rendering it useless. With this in mind, some method of maintaining the data integrity must be implemented.

- Availability of cloud system: Standard forensic investigation protocol is to disconnect the investigated system from the power supply and then do the investigation off-line. Assuming that the investigation is not related to some critical scenario, for example government security, it will not be possible to shut down an entire cloud data centre for an investigation.

These general problems shed some light on the enormity of the challenge of cloud forensics. From these challenges, a specific problem can be identified that will be addressed in this dissertation. This problem is discussed in the next section.

1.2 Problem statement

The problem with cloud forensics is that, due to the volatility, volume and distributed nature of data in cloud computing systems, standard digital forensic processes, such as the ISO 27043 standard, cannot be applied. The reason for this is that the standard digital forensic processes, such as described in the ISO 27043 standard, are severely disruptive to the live systems under investigation [12].

When considering a cloud system, data volatility is the first major challenge that has to be overcome in order to perform a forensic investigation. Cloud systems have a pool of computing resources that can be used to create virtual machines to perform some function [13, 14]. When a virtual machine is no longer needed, it is either saved and powered down, or destroyed. In the above cases the computing resources are returned, in part or in their entirety, to the resource pool for reuse.

This has the effect that valuable data might be lost, making it unavailable as digital evidence in a digital forensic investigation.

In addition, the volume of data in a cloud environment can be vast. Cloud environments can range from small home clouds of a few nodes, to huge enterprise clouds with petabytes of storage [8,14]. When dealing with these massive clouds for a digital forensic investigation, there is no efficient method of locating data within them without obtaining substantial additional information of where the required data is located [6].

Finally, the data in cloud systems can be fragmented and distributed throughout the cloud environment. Cloud systems are based on clustering technology that, by its nature, causes data to be fragmented [8]. In large enterprise clouds running large virtual machines, data can be fragmented over multiple physical disks operating on multiple different servers. Should a digital forensic investigation be required, multiple servers, where the affected data is located, would have to be taken off-line. These machines would each have to be investigated and the relevant evidence gathered in fragmented form and rebuilt into a coherent form, while maintaining the forensic integrity of the information. This would also result in undesirable downtime and might affect virtual machines that are not part of the digital forensic investigation. Virtual machines can also migrate within the cloud environment, depending on their resource needs and the operational parameters of the cloud system.

These technical problems also create a procedural problem in that commercial cloud systems cannot be taken off-line for long periods of time to conduct a forensic investigation. When using standard digital forensic procedures, it is common practice to remove the power from the system that needs to be investigated [15]. This would be impossible in a commercial cloud system, as it could lead to considerable losses in revenue and possible legal action by customers.

Taking into consideration the above-mentioned problems, it is also extremely challenging to ensure forensic readiness in a cloud computing system. Forensic readiness has two objectives, namely: Maximising an environment's ability to collect credible digital evidence, and minimising the cost of forensics in an incident response

[16]. Given the problems mentioned, it is clear that cloud systems fail both these objectives.

Taking into account these problems, a research question can be formulated to address the challenges facing digital forensics of cloud systems.

The research question that this dissertation addresses is the following:

Is it possible to achieve forensic readiness, in a cloud environment, so that a digital forensic investigation can be conducted with minimal or no disruption to the operation of said cloud environment?

This study aims to show that this is possible through the application of proactive digital forensics. Proactive forensics is defined as "the ability to proactively collect, trigger an event, and preserve and analyse evidence to identify an incident as it occurs. In addition, an automated preliminary report is generated for a later investigation ..." [17]. This system will be referred to as the Forensic Ready Cloud (FRC).

This study is limited to the basic cloud reference architecture [6] and specialised cloud systems will not be considered.

In the next section, the objectives of this study are stated.

1.3 Objectives

In order to realise the concept of forensics in a cloud computing system, certain criteria must be met. This dissertation defines and discusses said criteria and thus the goals of this dissertation are:

- To generate theoretical architectural models for conducting forensics in a cloud computing environment. The models will form a baseline from which the full system can be extrapolated and designed.
- To conduct experiments in order to determine if it is feasible to implement the theoretical architectural models. There is little use designing an architecture

that cannot be implemented, as it will not address the problem of cloud forensics.

- To implement one of the theoretical architectural models. The implemented architectural model serves as a test bed to determine if it is possible that such a model can be applied to a base cloud architecture.
- To conduct an experiment in order to determine if the implemented model is a feasible solution with regard to the problem of cloud forensics.
- To conduct a critical evaluation that addresses the feasibility and impact of the architectural models and to determine what can be done to improve them.

In order to meet these objectives, certain methodologies are used.

1.4 Methodology

This dissertation contains the following methodologies of research: A literature review, the generation of theoretical architectural models, implementation of one of the generated models and experimentation, and finally a conclusion as to the feasibility of the forensic readiness of a cloud computing system.

1.4.1 Literature review

A literature review is conducted to establish the current state of research with regard to the topic of computer forensics of cloud computing systems. This literature review serves as a baseline for the author to identify where a contribution can be made to the field. The literature review also gives the necessary background information of the topics presented in this dissertation.

1.4.2 Theoretical architectural models

Theoretical architectural models are developed to address the problem of forensic readiness of cloud computing systems. These model architectures are based on the standard cloud hardware virtualization architecture [18].

1.4.3 Full architecture

Using the Use-Case Responsibility Driven Analysis and Design (URDAD) methodology [19], a full architecture for the FRC system is developed. This is done by analysing the functional and non-functional requirements, generating the different architectural views and, finally, applying the relevant architectural structural patterns and tactics. By applying these techniques, a fully-fledged architecture is generated.

1.4.4 Implementation and experiments

The generated architecture is implemented using applicable technologies. A set of experiments is designed to measure the relevant metrics such as, among others, CPU and memory performance. These experiments are also used to either confirm or disprove the hypotheses.

1.4.5 Critical evaluation

The results are discussed and a conclusion is drawn from the available experimental results. Simultaneously, possible future avenues of research are also explored.

There are, however, certain elements that are outside the scope of this research. The scope and limitations are discussed in the next section.

1.5 Scope and limitations

This dissertation focuses on the acquisition and forensic soundness of data in a cloud computing environment. The acquisition of data concerns data captured from a virtual machine deployed in a cloud environment. The forensic soundness ensures that the captured data can be kept in a state where it would be admissible in a court of law as digital evidence. The data must be kept in this forensically sound state even while it is moved in the cloud environment.

With an industrial implementation of the proposed forensic system, one would be concerned with the both the core system for data acquisition and the ancillary components making the system more efficient. These ancillary components are outside the scope of this research. These components are networking components, data storage, implementation scaling, forensic analysis of data and analysis tools.

When regarding networking components, cloud computing systems are built with high performance and expensive networking hardware. The author does not have access to this level of infrastructure and thus uses a standard networking setup as an analogue. The network only functions to transfer data within the system and has no effect on the outcome regarding the success of the proposed system, as it is assumed that the network is set up and working normally.

As stated in section 1.1, the volume of data is a challenge in the field of cloud forensics. While the proposed system has mechanisms that attempt to mitigate the challenge of large volumes of data, data and storage management is not the focus of this research. Data that is captured must be stored somewhere in a cost-effective manner; however, the implementation of the storage system is outside the scope of this research.

The proposed system architecture is designed as a "bolt-on" system for existing cloud environments. Thus, some tactics can be applied to ensure that the proposed system can scale to the required size. While the proposed system should effectively scale in a cloud environment, this research does not analyse the feasibility of scaling the proposed system to full cloud environment.

Finally, the analysis of data and the analysis tools that can be used in a forensic investigation is not in the scope of this research. While a analysis tool is used in the implementation of the proposed system, it is merely to illustrate the working and feasibility of the data acquisition component of the proposed system. The proposed architecture is not dependent on the selection of analysis tool. The tool selection is only dependent on the requirements of a specific implementation.

In order to communicate the background effectively, research contribution and conclusion of this research, the dissertation layout is given in the next section.

1.6 Layout

This dissertation is divided into four parts. *Part 1* serves as an introduction to the rest of the dissertation. *Part 2* contains the literature review of this study. *Part 3* contains the author's contribution made to the field of study. Finally, *Part 4* contains the conclusion and possible future work. Figure 1.1 gives a visual representation of the flow of this dissertation.

The motivation driving this structure is that the background chapters give an overview of the fields of cloud computing and computer forensics. From this background, cloud forensic models can be generated that serve as the basis of the FRC architecture. With these models in place, a full requirements analysis can be done, taking into account all the relevant functional and non-functional requirements. An architecture, the Forensic Ready Cloud architecture (FRC), can then be generated that fulfils the identified requirements. Physical implementation and experimentation is done to verify that the FRC architecture is viable in real hardware. The results are analysed and critically evaluated to determine if the FRC architecture indeed addresses the problems as stated and what can or need to be improved.

The following is a breakdown of the chapters and short descriptions of the individual chapters of this dissertation.

1.6.1 Part 1: Introduction

Chapter 1 **Introduction:** (current chapter), serves as an introduction. It includes the research question, methodologies, objectives and layout of the dissertation.

1.6.2 Part 2: Background

Chapter 2 **Cloud computing:** contains the literature review which is done to determine the state of the art for the field of cloud computing. A study is done on the history of cloud computing from its origins as grid computing

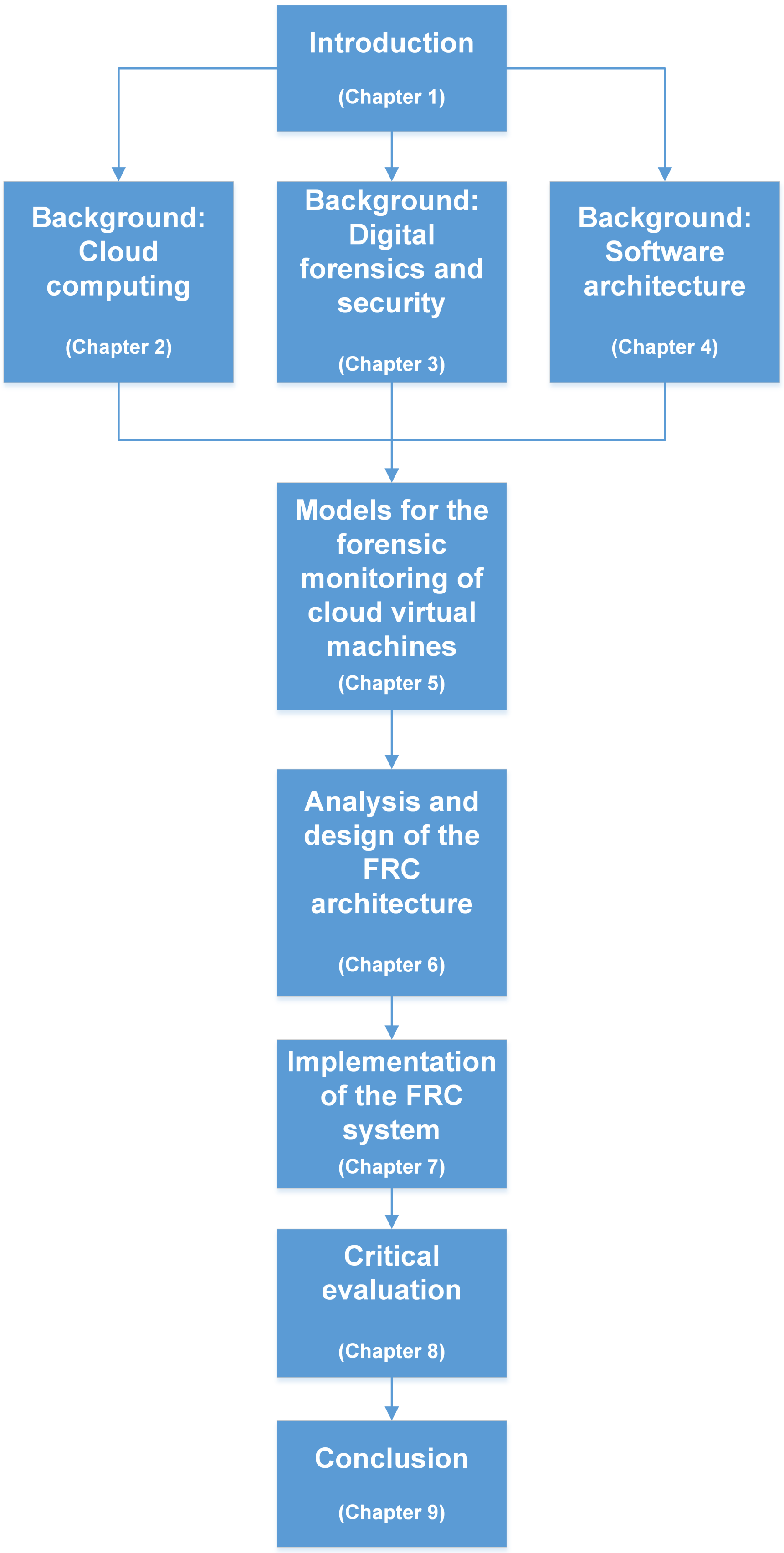

Introduction
(Chapter 1)
Background: Cloud computing
(Chapter 2)
Background: Digital forensics and security
(Chapter 3)
Background: Software architecture
(Chapter 4)
Models for the forensic monitoring of cloud virtual machines
(Chapter 5)
Analysis and design of the FRC architecture
(Chapter 6)
Implementation of the FRC system
(Chapter 7)
Critical evaluation
(Chapter 8)
Conclusion
(Chapter 9)

Figure 1.1: Dissertation layout

and clustering, to where it exists today, with the use of virtualization as self-provisioning of services. This chapter also introduces the National Institute of Standards and Technology (NIST) reference model for cloud computing on which this dissertation is based.

Chapter 3 **Digital forensics and security:** contains the literature review which is done to determine the state of the art for the field of computer forensics. A study is done of the history of computer forensics. This includes the current state of computer forensics, the standards associated with computer forensics, the current processes used in computer forensics and, finally, the state of cloud forensics.

Chapter 4 **System architecture:** contains the literature review which is done to determine the state of the art for the field of system architecture. A study is done on software architecture and how it influences the design, interaction and execution of software systems. This includes a software architecture description and the mechanisms through which such a description is achieved. These mechanisms include architectural viewpoint, architecture description languages, patterns and tactics.

1.6.3 Part 3: Research contribution

Chapter 5 **Models for the forensic monitoring of cloud virtual machines:** is the chapter that discusses the theoretical architectural models for forensics in cloud computing systems. It expands on the NIST reference model and introduces the concept of forensic monitoring and proactive forensics. From these concepts, the models for proactive computer forensics in cloud computing systems are derived and described. The developed models are compared and contrasted to determine which model will best meet the requirements of the FRC system.

Chapter 6 **Analysis and design of the FRC architecture:** is done in this chapter to define the functional and non-functional requirements in order to realise a proactive forensic cloud system. It is then possible to design an architecture that will fulfil these requirements and allow the system to be implemented.

Chapter 7 **Implementation of the FRC system:** contains the physical implementation process for the selected model and the experiments run on this model. These experiments include the effect of the forensic monitoring on the cloud system in terms of performance and whether the forensic monitoring is an adequate measure to extract forensic data. This is the main contribution to the field of digital forensics, in that the working implementation of the FRC system serves as proof of the concept of proactive forensics in cloud systems that, by extension, achieve forensic readiness in cloud computing systems.

1.6.4 Part 4: Conclusion

Chapter 8 **Critical evaluation:** contains the discussion of the results described in the previous chapter. This includes an analysis of how the architecture and implementation address the problems identified in section 1.2 Problem Statement, and possible improvements that can be made to address these problems better.

Chapter 9 **Conclusion:** contains the conclusion of the dissertation and possible future work. This final chapter summarises the results that were obtained and relate these results to the stated problems. Finally, possible future work in the field of cloud forensics is explored.

1.7 Conclusion

This chapter served as an introduction to the dissertation. The next chapter will contain the background and literature study of this dissertation. The following chapters will constitute the rest of the dissertation, including the architectural models, experiments, discussion and final conclusion.

Chapter 2

Cloud computing

2.1 Introduction

Cloud computing is one of the pre-eminent technologies in the current era of computer technology. It is the amalgamation and evolution of many different technologies and concepts spanning almost 4 decades. Ultimately the goal of cloud computing is for users to benefit from these technologies, without the need for a deep understanding of said technologies.

In this chapter, the different characteristics that define cloud computing will be examined. Next, the service and deployment models are discussed. Finally, the cloud reference architecture and core technology concepts are examined and the chapter is concluded.

2.2 Characteristics of cloud computing

As cloud computing has become clearly defined, certain characteristics have emerged that differentiate cloud computing as a distinct technology. These characteristics are as defined by the National Institute of Standards and Technology (NIST) [7]: on-demand self service, broad network access, resource pooling, rapid elasticity and measured service. Each of these characteristics are described briefly below.

On-demand self service. A consumer of the cloud service must be able to provision computing resources, such as server time and network storage, in an automated manner, without the need for human intervention.

Broad network access. The capabilities of the cloud system must be available over the network and accessed in some standard manner that can be used by multiple client platforms, such as workstations, laptops, tablets and mobile phones.

Resource pooling. A multi-tenant model is used to provide computing resources to multiple consumers from a pool of computing resources held by the cloud provider. These resources can be dynamically assigned and re-assigned, depending on the need of the consumers. The customer is generally unaware of the exact location of the computing resources and may only be able to, at best, assign a high-level physical location, e.g. data centre, city or country. The computing resources that can be utilised include processing power, data storage, memory and network bandwidth. The cloud provider is fully examined in section 2.5.1.2.

Rapid elasticity. Depending on the demand of the customer, computing resources can be scaled dynamically outward or inward. From the perspective of the customer, the computing resources appear to be unlimited and said resources can be used at any time.

Measured service. Computing resources in a cloud system are automatically controlled and optimised by leveraging a metering capability. The metering takes place at some level of abstraction, depending on the type of service that is used, whether active user accounts, storage, bandwidth or processing. Metering allows resource usage to be tracked, thus allowing the resources to be controlled and to report on usage. This provides transparency for both the cloud provider and consumer of the services.

In addition to these characteristics, a cloud infrastructure is a collection of hardware and software that functions to enable the core cloud characteristics. A cloud infrastructure consists of both a physical layer and an abstraction layer, which work in concert to provide the cloud services. The physical layer, consisting of servers, network components and storage, provides the hardware resources for the cloud system. The abstraction layer manifests the cloud services, and thereby the essential cloud characteristics, via the software that is deployed on it.

In order for cloud systems to provide as much flexibility as possible, different service models were developed. These models are discussed in the next section.

2.3 Service models

The service paradigms for cloud computing services have their basis in that of server-orientated architecture, which advocates the concept of *everything-as-a-service* [20]. Cloud computing, however, separates the services into distinct models, with the three distinct models being Software-as-a-service (SaaS), Platform-as-a-service (PaaS) and Infrastructure-as-a-service (IaaS) [7, 21]. Each of the models offers a different level of abstraction, thereby giving the service consumer flexibility.

2.3.1 Software-as-a-service

Software-as-a-service (SaaS) delivers a complete software application to a customer, using a subscription model. The customer needs only to configure certain aspects of the software, while not needing to interact directly with the underlying cloud platform or infrastructure. The customer can access said software using either a program interface or a thin client interface, e.g. a web browser [7, 22].

2.3.2 Platform-as-a-service

Platform-as-a-service (PaaS) allows a customer to deploy applications that are either self-created or purchased, onto cloud infrastructure, without needing to manage the underlying cloud infrastructure. The cloud system therefore removes the need for a customer to manage the low-level infrastructure, including servers, networking, storage, operating systems (OS) and so forth. Since the underlying infrastructure is managed, software developers have little to no control of the low level of the deployment environment [21, 22].

2.3.3 Infrastructure-as-a-service

Infrastructure-as-a-service (IaaS) allows the consumer to obtain fundamental computing resources, such as processing, storage, networking and so forth. However, the consumer has no control of the underlying cloud infrastructure, only over the choice of operating systems, storage and which software is deployed. Depending on the configuration of the cloud system, the consumer might also have limited control of networking components [21, 23].

In addition to the service models, cloud systems have different deployment models where software, platforms or infrastructure can be deployed as required. The following section deals with these models.

2.4 Deployment models

Cloud deployment models are defined to categorise who may use a certain cloud system. The deployment models are defined by the NIST as: public cloud, community cloud, public cloud and hybrid cloud [7]. Each of these deployment models is discussed below.

2.4.1 Public cloud

The infrastructure of a public cloud can be used by anyone in the general public. These services offered by the public cloud may be free of charge or based on some form of payment model. The cloud may be operated, managed and owned by institutions such as governments, academic institutions or businesses. These institutions, in turn, offer the cloud services to the general public. The physical cloud infrastructure resides on the premises of the cloud provider [7, 24].

2.4.2 Community cloud

In the case of a community cloud, the infrastructure is provided for some specific community of users with some common requirement. This requirement may range from some policy, mission, security requirement or compliance concern. An example of a community cloud is a cloud for tertiary education institutions. These institutions have similar educational, administrative and informational services that are typically spread over a wide geographical area. Using a community cloud would cut IT operating expenses, as the individual institutions would not be required to run large individual IT departments [25].

The cloud may be operated, managed and owned by one or more members of the community it services. It might also be owned by some third party, or a combination of community members and third parties. These third parties can range from dedicated cloud providers to telecommunication companies, to anyone who has the capability to host a cloud system. Because of these diverse ownership roles, the infrastructure might reside on or off the premises of any of the owners [7, 24].

2.4.3 Private cloud

Private cloud infrastructure is provisioned for the sole use by one organisation. The internal departments of the organisation are the consumers of the cloud services. The physical infrastructure might be owned by the organisation itself, a third party or a combination of both [7, 24]. As many organisations are moving to the model of outsourcing IT infrastructure, it stands to reason that the same can be done for cloud systems. An organisation might choose to have a minimal footprint IT infrastructure on-site and have the rest of the infrastructure hosted in a private cloud off-site [26].

2.4.4 Hybrid cloud

A hybrid cloud model may consist of two or more of the deployment models. Each component remains unique, but they are bound by some technology that allows both data and application portability [7, 24]. A scenario of an implementation of

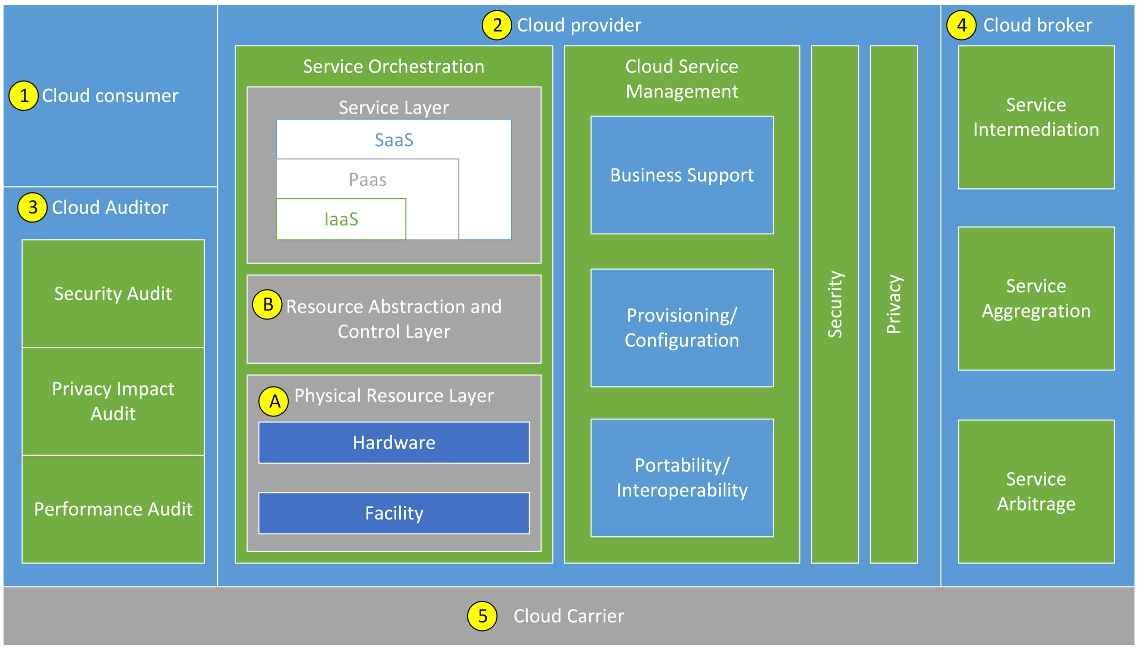


Figure 2.1: NIST Cloud reference architecture [1]

a hybrid cloud would be one of an organisation that already has IT infrastructure running as a private cloud for normal operational purposes. Should the organisation need temporary additional computing resources for computationally heavy tasks, for example bulk report running, such tasks can be completed using the additional computing power of a public cloud. After the need for such resources ends, the organisation can revert to using only its private cloud.

With the characteristics, service and deployment models defined, a reference architecture and core technological concepts can complete the picture of a cloud computing system. The NIST reference architecture is discussed in the next section.

2.5 Cloud reference architecture

The NIST provides a reference architecture in which all the major components, activities and functions of cloud systems are defined [1]. Figure 2.1 depicts the full reference architecture and Table 2.1 all the different role players of the architecture. The different stakeholders (actors), components and core technological concepts are discussed below.

Table 2.1: Actors in Cloud Computing

Actor	**Definition**
Cloud consumer	A person or organisation that maintains a business relationship with, and uses service from, Cloud Providers
Cloud provider	A person, organisation, or entity responsible for making a service available to interested parties
Cloud auditor	A party that can conduct independent assessment of cloud services, information system operations, performance and security of the cloud implementation
Cloud broker	An entity that manages the use, performance and delivery of cloud services, and negotiates relationships between Cloud Providers and Cloud Consumers
Cloud carrier	An intermediary that provides connectivity and transport of cloud services from Cloud Providers to Cloud Consumers

2.5.1 Cloud components and stakeholders

Figure 2.1 depicts the full reference architecture for cloud systems and Table 2.1 the stakeholders (actors) for the cloud components as described by the NIST [1]. These stakeholders are the cloud consumer, provider, auditor, broker and carrier. Each is discussed in the sections below.

2.5.1.1 Cloud consumer

The cloud consumer (Figure 2.1 block ①) is the main stakeholder of the cloud computing service. The cloud consumer can be an organisation or person who uses a service supplied by a cloud service provider. This service is selected from s service catalogue and a contract is set up between the cloud service provider and the cloud consumer. The cloud consumer is billed according to the service that is utilised.

A service level agreement (SLA) between the cloud service provider and the cloud consumer is needed to specify, among other parameters, performance, security, quality of service and remedies for service outages. The SLA must also explicitly state the limitations and obligations of the service contract. Included in this must also be the pricing policy of the cloud service provider [27].

The usage scenarios are dependent on the services selected by the cloud consumer. For example, SaaS cloud consumers can have access to software applications as end users. These consumers can be billed per number of users, time of use, network bandwidth consumed, volume of data stored or duration of storing the data.

PaaS consumers have tool and execution resources provided to them by the cloud service provider, in order to deploy and manage applications within the cloud environment. These consumers can range from application developers who design and implement software applications, to application administrators who monitor and configure applications on the cloud platform. Billing for these consumers can be according to processing resources consumed, network resources utilised, database storage or platform usage.

IaaS consumers are given access to virtual computers, network infrastructure and storage, as well as any other fundamental computing resources they could require. These consumers are able to run any software that they choose. These consumers can be anyone from system developers to IT managers looking for computing resources for IT operations. Billing usually occurs according to the computing resources consumed, including CPU time, data volume, network bandwidth consumed or IP addresses used.

All these services are supplied by the cloud provider.

2.5.1.2 Cloud provider

The cloud provider (Figure 2.1 block ②) is the organisation or person who makes the services available to the cloud consumer. It is the responsibility of the cloud provider to acquire, manage and maintain the IT infrastructure in order to provide services via network access to the cloud system. These services differ per service model.

For SaaS, it is the responsibility of the cloud provider to deploy, maintain and update the software applications on the cloud system so that the services can be offered to the cloud consumers. This results in limited control of the applications by the cloud consumer and the responsibilities for the management and control of the infrastructure resting with the cloud provider.

In the case of PaaS, the cloud provider is tasked with running the platform and its underlying software, including the software execution stack, databases and middleware components. The PaaS cloud provider can also maintain and support the tools used by the cloud consumer to develop, deploy and manage software applications. These tools can include integrated development environments (IDEs), development version of cloud software, software development kits (SDKs), deployment and management tools. The cloud provider retains exclusive access to the underlying infrastructure such as networks, servers, operating systems and storage. The cloud consumer, in turn, has control over applications and in some cases hosting environment settings.

Finally, for IaaS, it is the responsibility of the cloud provider to acquire the physical computing resources. These resources can include servers, networks, storage and other hosting infrastructure. These resources are abstracted into virtual machines and virtual network interfaces. This is achieved via the cloud software that makes the abstracted resources available to the cloud consumer. The cloud consumers can use these resources to fulfil their computing needs. This can be done on a fundamental level, as opposed to the SaaS and PaaS deployment model. In the case of IaaS, the cloud consumer has control over components such as the OS and network. The cloud provider ensures that the provisioning infrastructure is maintained by maintaining the physical hardware and cloud software.

The provided services and how they are utilised might require independent examination from time to time. This is done by the cloud auditor.

2.5.1.3 Cloud auditor

The cloud auditor (Figure 2.1 block $\textcircled{3}$) is an independent organisation or person who examines and expresses an opinion on cloud service controls. Audits on clouds are conducted in order to verify that said cloud conforms to set standards. These audits can be regarding security controls, privacy impact, performance and so forth.

Auditing is important for cloud providers, as it should be possible for third parties to assess the security controls [28]. The confidentiality, integrity and availability (CIA) of the cloud system and its information are protected by security controls, including management, operational and technical safeguards [29]. The CIA concept is examined in section 3.10. The auditor makes an assessment of the security controls and whether these controls are implemented correctly, operational and fulfilling their requirements within the cloud system. Included in this audit should also be an assessment of whether the system is in compliance with the security policy and regulations. These policies can include data retention, access control, that fixed content cannot be modified, and so forth. All this must be done to comply with legal and business requirements.

Closely linked to the security audit, is a privacy audit that must ensure that personal information is kept in compliance with local laws and regulations and that such information is kept in adherence of the CIA principle. This must be enforced at every stage of development and operation of the cloud system.

The multitude of services offered by cloud providers requires some intermediary to manage their integration. This is the role of the cloud broker.

2.5.1.4 Cloud broker

With the rapid evolution of cloud systems, it can easily become too complex for cloud consumers to manage the integration of cloud services. A cloud broker (Figure 2.1 block $\textcircled{4}$) may act as an intermediary between the cloud consumer and cloud provider. Thus, the cloud broker manages the use, performance and delivery of cloud services, as well as negotiating the relationship between the cloud consumers and cloud providers.

In general, the cloud broker provides services in three categories [1, 30]:

Service intermediation. Specific services are enhanced by improving some capability, thus adding value to services for the cloud consumer. These improvements can range from access control management, identity management, performance reporting, security enhancements, and so forth.

Service aggregation. Multiple services can be combined and integrated into one or more new services. Data integration is provided by the broker, as well as ensuring secure data movement between the multiple cloud providers and the cloud consumer.

Service arbitrage. As with service aggregation, services are combined. However, in the case of service arbitration, the services are not fixed. In other words, the broker has the option to choose the services from multiple cloud providers.

In order for the services to reach the cloud consumer, some intermediary is needed to provide connectivity. This is done by the cloud carrier.

2.5.1.5 Cloud carrier

Cloud providers (Figure 2.1 block ⑤) need some method of connecting cloud consumers to the offered services. This requirement is fulfilled by the cloud carrier, in that it connects the cloud consumer to the cloud provider via a network, telecommunication network or some other access device. Devices such as computers, laptops, mobile phones, etc. can be used by the cloud consumer to access cloud services [7]. Access and distribution to these services are usually provided by some network or telecommunication carriers. The cloud carrier is distinct from the cloud broker, in that the cloud carrier is only concerned with the data communication between the cloud provider and cloud consumer. The services offered by cloud providers and how they are integrated, are the responsibility of the cloud broker.

The different stakeholders of cloud systems show each of their different views regarding their role in cloud systems. However, certain technologies underpin cloud systems to enable their operation. These technologies are discussed in the next section.

2.5.2 Core technological concepts

Cloud computing has its underpinnings in a number of different technologies. These technologies are: clustering, virtualization and hypervisors. Each of these technology concepts is discussed below.

2.5.2.1 Clustering

The clustering (Figure 2.1 block Ⓐ) of computers occurs when two or more computing systems work together to perform some function [31]. The aim of this is to have a scalable solution that gives flexibility in terms of computing power, redundancy or availability. As cloud computing relies on large pools of resources to operate, the practice of clustering many nodes becomes a very attractive proposition. This is because it allows many redundant nodes to form the resource pool.

2.5.2.2 Virtualization

Virtualization (Figure 2.1 block Ⓑ) is one of the core technologies used in cloud computing infrastructure [32–34]. This enables users to have access to scalable, on-demand computing systems [23]. Virtualization works by abstracting computing resources from their physical counterparts [35] into a resource pool from which the resources can be drawn by users. A virtual machine can thus be provisioned by a user, as long as the resources required for the virtual machine do not exceed that of the physical host machine. This allows multiple virtual machines to run on a single host [36, 37].

2.5.2.3 Hypervisor

A hypervisor or virtual machine manager (Figure 2.1 block Ⓑ) is a piece of software that allows the simultaneous running of one or more operating systems known as virtual machines (VM) [38, 39]. The virtual machines running on the system are called the guest operating systems. Each guest is allocated resources by the hypervisor from the host resource pool. Hypervisors are divided into Type 1 and Type 2 hypervisors.

- Type 1: This type is known as a bare metal or native hypervisor, which runs directly on the hardware without a host OS.
- Type 2: This type is known as a hosted hypervisor, which runs on a host OS. The host OS is installed on the host system and the hypervisor is installed on the host OS.

From the points above, it can be seen that the main difference between Type 1 and Type 2 hypervisors is that Type 1 hypervisors run directly on hardware and Type 2 hypervisors run on additional software, e.g. the host operating system. Type 1 hypervisors tend to more efficient, as the hypervisor is built into the firmware of the computer on which it runs. It also provides better security, availability and performance, according to IBM [40]. Since Type 2 hypervisors run on a host operating system, it can be installed on existing servers, with no need to reformat such devices.

With the concept of cloud computing defined and examined, digital forensics and information security must also be investigated.

2.6 Conclusion

This chapter shed some light on the concepts underpinning cloud computing. These concepts include the characteristics of cloud computing, service models, deployment models, the NIST reference architecture for clouds and cloud forensics. This background on cloud computing allows the cloud component of the FRC system to be designed. As the Forensic Ready Cloud (FRC) system is designed to enable proactive forensics in cloud systems, some background regarding digital forensics and computer security is required. In the next chapter, digital forensics and computer security will be examined.

Chapter 3

Digital forensics and security

3.1 Introduction

Forensic science has its origin in 1879 France, where a Parisian police clerk with the name Alphonse Bertillon introduced the notion of documenting crime scenes by photographing corpses and other items left behind at crime scenes [41]. With his novel approach, he laid the foundation of forensic science, in that he provided a photographic record of a crime scene, along with accurate measurements and cataloguing of corpses. Bertillon's radical notion of investigating crimes via science and logic had a profound effect on one of his students, Edmond Locard.

Locard, in turn, codified the exchange principle. This principle states that "a criminal action of an individual cannot occur without leaving a mark" [41]. In other words, any criminal act will leave behind some form of trace evidence due to the force required to perpetrate such an act [42]. The work of Locard thus gave rise to the core concepts that aid investigators to this day. These concepts are: crime scene documentation, suspect identification and the discipline of trace analysis. These concepts are just as relevant to digital forensic science as they are to traditional forensic sciences such as toxicology, ballistics, fingerprinting and DNA analysis.

This chapter contains background information regarding the field of digital forensics, specifically computer and cloud forensics. Firstly, the history of digital forensics is reviewed, leading to a definition of digital forensics. Next, the concept of digital evidence is discussed. With the concept of digital evidence, methods of how such evidence is obtained are discussed, including live and dead or postmortem forensics, as well as proactive and reactive forensics. Following this, the different branches of digital forensics are discussed, with emphasis on cloud forensics. In addition, the digital forensic process is discussed, including the Cohen model and ISO 27043 standard. Some concepts of computer security are also discussed. Computer security is related to computer forensics in that a forensic investigation might be required because of a failure in computer security. Finally, the chapter is concluded.

3.2 History of digital forensics

The earliest crimes perpetrated against computing systems were mostly physical damage that was committed, in general, by dishonest or disgruntled employees. This was the prevalent threat until the 1980s, until programmers began creating malicious software [43]. With the Internet becoming more widely available during the 1990s, so too did crime against poorly protected systems. These systems were targeted for vandalism, political action and financial gain. In the latter part of the 1990s, the attack patterns changed as operating systems were made more resilient against attack [43, 44]. Computer crime is, and will continue to be, present in society. Currently it seems as though computer crime has moved to a more organised level, including large-scale fraud and other undertakings, such as international drug trafficking, arms trading and human trafficking, requiring high-tech communications. Nation states are also having to deal with information warfare and cyber-terrorism [44].

The first organisation dedicated to digital forensics, called the International Association of Computer Investigative Specialists (IACIS), was founded in 1989 as a collaboration between law enforcement individuals in the United States [45]. While the organisation received little support in the beginning, the work of the individuals who formed the IACIS, formed the basis of modern digital forensics [46]. The FBI held the First International Conference on Computer Evidence in 1993, which was attended by representatives of 26 countries. A second conference was held in 1995 and the International Organisation of Computer Evidence (IOCE) was founded [47].

While the early cases focused mostly on data recovery from stand-alone computers, much of the computer crime related to hacking of telephone networks in order to gain free access to telephony services [48].

During the period of 1995 to 2005, three main drivers gave rise to the growth in the field of digital forensics, especially in the United States. These drives were [46]:

- The explosion of computer technology. From the mid 1990s to the mid 2000s, computer technology became a ubiquitous and indispensable part of life. As computers, mobile devices and the Internet became omnipresent, so too did criminal activities related to these technologies.
- The explosion of child pornography cases. While investigating the case of George Stanley Burdynski Jr. in 1993, investigators found that computers were used to traffic illegal images of minors. This led to the FBI establishing the Innocent Images operation in 1995 to investigate cases of child pornography [49]. This led to the seizure of ever-increasing volumes of digital evidence, resulting in the growth of digital forensics.
- The rise of global terrorism, including the attacks on September 11, 2001, also known as the 9/11 attacks. While the 9/11 attacks did not involve computer crime directly, a wealth of digital evidence was found on computers around the world. This prompted law enforcement, military and intelligence communities to improve their digital forensic capabilities.

During this time period, the emphasis also changed from the concept of the "resident expert", to a more formal digital forensic investigator. This was due to the fact that the drive behind digital forensics was taken over by professional organisations and government agencies, as opposed to individuals. Between 1999 and 2000, the G-8 High Tech Crime Subcommittee and Scientific Working Group on Digital Evidence (SWGDE) and IOCE published principles for digital forensics [50–53]. Along with this, in 2004, the American Society of Crime Laboratory Directors Laboratory Accreditation Board (ASCLD-LAB), with the SWGDE, formalised digital evidence as a laboratory discipline [54].

The field of digital forensics remains a rapidly growing field, with information security professionals viewing digital forensics as core to their skill set. Digital forensic practitioners are now formally trained and many academic programmes have emerged across the globe [46]. In addition to this, digital forensics is an active field of academic research, with conferences like The Digital Forensic Research Workshop (DFRWS), the International Federation for Information Processing (IFIP) Working Group 11.9 on Digital Forensics and the International Conference on Systematic Approaches to Digital Forensic Engineering (SADFE) [55–57].

With this brief historical overview of digital forensics, it is possible to define digital forensics.

3.3 Definition of digital forensics

Digital forensics can be defined as "the use of scientifically derived and proven methods toward the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorised actions shown to be disruptive to planned operations" [58–60].

From this definition it becomes apparent that the core concept of digital forensics is that of digital evidence. The next section examines digital evidence in more detail.

3.4 Digital evidence

Digital evidence is defined by Casey as "probative information stored or transmitted in digital form that a party to a court case may use at trial" [61,62], or by Boddington as "information in digital form found on a wide range of computer devices; in fact, it is anything that has a microchip or has been processed by one and then stored on other media" [41]. For the purposes of this research, digital evidence is defined as "information that is stored or transmitted in a digital or analogue format, via a computer or other transmission or storage system, that has relevance to a forensic investigation".

Digital evidence can take many forms including, but not limited to, e-mails, digital pictures, videos, audio recordings, instant messages, electronic documents, spreadsheets, databases, system logs and so forth. This type of evidence can be referred to as real evidence, similar to physical documents or photographs, but also IT evidence, depending on the relevant jurisdiction [41]. Regardless, most digital evidence is presented in the form of verbal testimony, in a court, by the forensic practitioner.

Digital evidence is used in a similar manner as documentary evidence, in that it is used to support or refute claims made in legal cases. It may be used to support of refute some key part of circumstantial evidence, in a similar manner as documentary evidence. Because of this, it is of the utmost importance that the evidence is reliable. Digital evidence can play a part in both criminal and civil cases and, in some instances, it might be the only evidence that is presented.

When referring to reliable evidence, the concept of digital forensic soundness must be explored. The term "reliable evidence" is often used interchangeably with the term "digital forensics soundness of evidence". McKemmish defines the term forensically sound as "the application of a transparent digital forensic process that preserves the original meaning of the data for production in a court of law" [63]. When analysing this definition, certain terms must be taken into account. Firstly, "transparent" in this context means that the forensic process that is followed must be reliable and accurate. It must also be possible to be tested and verified. Next, the phrase "preserves the original meaning" means that the captured data must be captured by means of a forensic process, so that the interpretation of the captured data maintains the meaning of the original data. Finally, the term "digital forensic process" refers to both the employed methodology and the used technology.

For digital evidence to be presented, it must first be acquired by some means. In the next section, live and postmortem forensics will be viewed as mechanisms for the acquisition of digital evidence.

3.5 Postmortem and live digital forensics

Postmortem and live forensics are the two methods of acquiring and analysing data in a digital forensic investigation. Both these methods are discussed below.

3.5.1 Postmortem forensics

Postmortem or dead forensics is the "traditional" method of gathering forensic evidence [41, 58, 64]. With this method, common practice is to remove the power source from the computer that is under investigation. This results in the computer being shut off instantly without the risk of changing any of the data on the hard drive. The resulting investigation is thus done on a "dead system", with a data autopsy being performed [61,62,65]. The primary focus is that of data recovery and analysis of the information that is stored on the hard drive or other data storage device, depending on the type of system under investigation.

The major disadvantage of postmortem forensics is that all volatile data is lost when the computer system is shut down. To mitigate this problem, the concept of live forensics is used.

3.5.2 Live forensics

Live forensics is the capture and analysis of digital evidence from a computer system that has not been shut down prior to the forensic investigation taking place [65,66]. The main advantage of live digital forensics is the ability to preserve volatile data, such as data stored in the RAM of a computer. In some cases, this might also result in the ability of the investigator to observe what is happening on the system during an incident [64].

In an inversion to the problem with postmortem forensics of destroying volatile data, live forensics runs the risk of changing the data being investigated. Because the system is still active, the actions of the investigator can have an effect of the data under investigation [41, 58].

Linked closely to the concepts of postmortem and live digital forensics, are the concepts of proactive and reactive digital forensics. These concepts are discussed in the next section.

3.6 Reactive and proactive digital forensics, and digital forensic readiness

The collection of digital evidence via live and postmortem forensics, can be closely linked to the process of analysing the captured digital evidence. Reactive digital forensics is a manual process of acquiring and analysing digital evidence after an incident has occurred. Proactive digital forensics, in turn, is where the forensic acquisition process is done in a live manner and the initial analysis is done by an automated system [17]. Finally, forensic readiness is policies and procedures put in place to enable a successful digital forensic investigation. Each of these concepts is discussed in the sections below.

3.6.1 Reactive digital forensics

Reactive digital forensics is the traditional forensic process of postmortem forensics. In this case, the forensic investigation is conducted after an incident occurs [60]. In the case of a reactive digital forensic investigation, two types of evidence are collected. These evidence types are active and reactive evidence. Active evidence refers to evidence that is still in a live or active state, for example processes still in memory. Reactive evidence, on the other hand, refers to static or dead evidence, like the image of a hard drive [17]. The detailed process of reactive digital forensics is discussed in section 3.8.

The contrast to reactive digital forensics is proactive digital forensics.

3.6.2 Proactive digital forensics

Proactive digital forensics can be described as the capability to collect, preserve and analyse digital evidence proactively, initiated by some triggering event. In addition to this, a preliminary report is automatically generated to aid in the later investigation [17]. The evidence that is gathered in this proactive manner relates to a specific incident or event while it is occurring [67].

Proactive digital forensics has phases linked to the steps of the forensic process, namely proactive collection, event triggering function, proactive preservation, proactive analysis, and preliminary report [17].

- Proactive collection is the live collection of predefined digital evidence according to a requirement set by an organisation. The evidence is collected in order of volatility and priority.
- Event triggering function is a function that is triggered by some suspicious event that starts some process to aid the forensic process.
- Proactive preservation is the forensic preservation of the captured digital evidence, done via a process of hashing.
- Proactive analysis is the live analysis of digital evidence via some automated system using forensic techniques. This analysis can be used to create an initial hypothesis of the incident.
- Preliminary report is generated from the actions of the proactive system.

It should be noted that the proactive collection of digital evidence is not the same as a traditional intrusion detection system (IDS). The evidence collection process ensures that the digital evidence is captured in a manner that is forensically sound. The analysis of the captured digital evidence must be done in a manner that ensures that it is admissible in a court of law [17].

In order to enable proactive digital forensics, controls in the form of policies and procedures must be set in place. These controls lead to forensic readiness.

3.6.3 Forensic readiness

Pangalos, Ilioudis and Pagkalos define forensic readiness as “the state of an organisation where certain controls are in place in order to facilitate the digital forensic processes and to assist in the anticipation of unauthorised actions shown to be disruptive to planned operations" [68]. Forensic readiness is one of the main factors that will allow an organisation to use digital evidence that was gained from a digital forensic investigation successfully, while minimising the cost associated with such an investigation [68,69].

Tan defines two concepts that forensic readiness would cover [16]:

- Maximising an environment's ability to collect credible digital evidence; and
- Minimising the cost of forensics during an incident response.

From these two points, five factors can be extrapolated that can affect digital evidence, in terms of evidence preservation and investigation time. These factors are [70]:

- How is logging done;
- What is logged;
- What is done in terms of intrusion detection and intrusion detection systems;
- Forensic acquisition; and
- Evidence handling.

Forensic readiness should be a corporate goal, as it will maximise the use of digital evidence in case it is needed. This can range from supporting a legal process to system disaster recovery [71,72]. The corporate actions necessary to enable forensic readiness are both technical and non-technical.

Rowlingson defines ten key activities for implementing forensic readiness. These steps are [70]:

1. Define the business scenarios that require digital evidence.
2. Identify available sources and different types of potential evidence.
3. Determine the evidence collection requirement.

4. Establish a capability for securely gathering legally admissible evidence to meet the requirement.

5. Establish a policy for secure storage and handling of potential evidence.

6. Ensure monitoring is targeted to detect and deter major incidents.

7. Specify circumstances when escalation to a full formal investigation (which may use the digital evidence) should be launched.

8. Train staff in incident awareness, so that all those involved understand their role in the digital evidence process and the legal sensitivities of evidence.

9. Document an evidence-based case describing the incident and its impact.

10. Ensure legal review to facilitate action in response to the incident.

With the application of these steps, forensic readiness can be achieved in an organisation [68].

Proactive and reactive digital forensics allow digital forensic investigations to be conducted and forensic readiness puts in place the control to make the investigation successful. There are, however, many different branches of digital forensics, each with its unique challenges. In the next section, these different branches of digital forensics are investigated.

3.7 Branches of digital forensics

Just as digital forensics is a branch of forensic science, digital forensics have a number of sub-branches that focus on more specific areas of the field. These sub-branches are computer forensics, database forensics, network forensics, mobile device forensics and cloud forensics [73]. There are, of course, many more branches of digital forensics, such as embedded systems forensics, but they are not relevant to this research.

3.7.1 Computer forensics

Noblett et al. define computer forensic science as “the science of acquiring, preserving, retrieving, and presenting data that has been processed electronically and stored on computer media” [74]. From this definition, it can be seen that computer forensics is the process of acquiring and analysing digital evidence from computers and computing systems when these systems were used in some form of computer crime [75]. The investigation should be conducted by an investigator who is skilled in computer forensics.

Digital evidence acquired from the computer during the application of a computer forensic process can take the form of files, applications, hidden information or the operating system. The hidden information can be information that is intentionally hidden or information hidden due to it being in unallocated hard drive space or files that have been deleted.

Computer forensics can be done using live or postmortem techniques. Using live forensic techniques, data stored in RAM can be captured and analysed as potential digital evidence [62]. The more common method of computer forensics is postmortem forensics. In this case, the power is removed from the computer and the data is captured from the hard drive, using a write blocker to ensure evidence integrity [62].

3.7.2 Database forensics

Database forensics, as the name suggests, deals with the forensic analysis of databases and meta data related to databases [76]. As with computer forensics, a forensic process is followed, but it is limited to the content of the database and related meta data. Database forensics is primarily a reactive form of digital forensics, however, cached information might still remain in RAM, thus requiring live analysis.

The primary mechanism for database forensics is that of log analysis. Time-stamped transaction logs can be used to recreate the incident resulting in changes or damage to the database [77].

One of the main challenges regarding database forensics is that of the variety of different database management systems available. When looking at relational databases, the logical structures are similar, but the specific implementations can vary significantly. As such, a great deal of knowledge is needed by the investigator [78].

3.7.3 Network forensics

Network forensics is the investigation of an incident regarding activities on a digital network. This can involve monitoring the network and capturing network traffic related to digital forensic investigations, in a manner that the captured evidence is admissible in a court of law [79]. Network forensics is distinct from computer forensics in that network forensics deals with volatile data and data is acquired using live forensics. Also, network forensics must be done in some proactive manner. Network traffic must be captured actively, lest it be permanently lost after the transmission is completed [70].

Another component of network forensics is log analysis. Logs stored on computers and network devices, such as switches and routers, can be invaluable to a digital forensic investigation. Logs are time-stamped and thus can be put in sequential order. Using such logs, an event timeline can be reconstructed to a certain extent [79]. Logging is, however, not standardised and is vendor-specific. This results in increased complexity for the investigator, as logs must be correlated to build an accurate sequence of events.

3.7.4 Mobile device forensics

Mobile device forensics relate to forensics done on a mobile computerised device. These devices can range from mobile phones, portable music players, pagers and so forth. Mobile devices have the characteristics of having smaller storage capacity and less computational power than their non-mobile counterparts. It is also quite likely that more modern devices will have some form of network connection, usually via a form of wireless technology such as WiFi, Bluetooth or Near-field communication (NFC).

Evidence from mobile devices can be obtained from three primary sources. The sources are internal memory, external memory and service provider logs. In the case of internal memory, evidence is stored within the device, mostly in flash or solid state memory [80]. This memory is integrated into the device and cannot be removed easily. External memory, in the form of Secure Digital (SD) cards, subscriber identification module (SIM) cards, CompactFlash (CF) cards and Memory sticks, can be removed from the mobile device easily [81, 82]. Finally, service provider logs are external to the mobile device and kept by the mobile service provider, in the case of mobile phones. Evidence contained within internal and external memory is treated as primary evidence, as it is directly linked to the mobile device. On the other hand, service provider logs serve as supplementary evidence. This evidence can include call logs, text messages, cell tower connections and, in some cases, location of the mobile device [81, 82].

3.7.5 Cloud forensics

Cloud forensics refers to a digital forensic investigation that is conducted on a cloud computing system [9]. Cloud computing forensic science is defined by the US National Institute of Standards and Technology (NIST) as "the application of scientific principles, technological practices and derived and proven methods to reconstruct past cloud computing events through identification, collection, preservation, examination, interpretation and reporting of digital evidence" [6, 7]. Due to the complex nature of cloud computing, a digital forensic investigation has proven over time to be a complicated endeavour [9, 83, 84]. Because of the dynamic environment of cloud systems, investigations can be highly complex for an investigator.

As cloud computing systems have unique characteristics that enable them to perform unique functions, so too are there unique needs in terms of cloud forensics. The NIST defines 9 areas of major concern regarding cloud forensics. These areas are: architecture, data collection, analysis, anti-forensics, incident first responders, role management, legal issues, standards and training [6].

Architecture. Just as the offerings of cloud systems are diverse, so too are the architectures on which they are built. These architectures can vary between cloud providers. This results in problems regarding data compartmentalisation and isolation, system proliferation, data storage locations, provenance of data and infrastructure seizure without disruption to other cloud tenants. Of particular concern is the provenance of data, as the provenance is required to maintain the chain of custody, but can be made difficult due to issues with multi-tenancy and data segregation.

Data collection. In order for a digital forensic investigation to be conducted, data must be collected from the system under investigation. However, with cloud systems, such data can be located in large, distributed and dynamic cloud environments. The collection of volatile data must therefore occur in a multi-tenant environment, from virtual machines, where the data could be shared between multiple users, across multiple locations and may be accessible by multiple parties. This can create an issue where the confidentiality of other tenants in the cloud may be compromised in the collection of forensic artefacts [8].

Analysis. Data captured from cloud systems can be challenging to analyse, in that the forensic artefacts might need correlating across multiple cloud providers. In addition to this, the reconstruction of events from virtual machines and virtual storage can be very difficult. Finally, challenges exist in verifying the integrity of meta data and the analysis of log data timelines.

Anti-forensics. To achieve anti-forensics, a set of techniques is used in order to mislead or prevent forensic analysis [85]. The challenges faced can range from data hiding and data obfuscation to the use of malware. The ultimate goal remains to compromise the integrity of the evidence.

Incident first responders. Issues with first responders generally revolve around competence and trustworthiness from the cloud provider toward the first responder, in his or her ability to perform tasks such as data collection, performing initial triage and processing large volumes of forensic artefacts.

Role management. In cloud systems, correctly and uniquely identifying the owner of an account can be a challenge. This is due to the decoupling that occurs between the users and the credentials they use, and the physical person associated with said credentials. As it is easy to create a fake identity online, determining the exact owner of data can be problematic.

Legal. As with most investigations, legal issues must be taken into account. In the context of cloud computing, these issues can include in which jurisdiction data is located, problems with international communication channels of investigations, cooperation and communication of cloud providers in the case where forensic data is spread across said multiple cloud providers, missing contracts and SLAs, issuing of subpoenas without adequate knowledge of the physical location of data relevant to the investigation and that the business continuity of other cloud tenants could be interrupted with the confiscation and seizure of cloud resources.

Standards. Standards remain lacking in the field of cloud forensics. Even limited standard operating procedures can be lacking, much less to say interoperability between cloud providers and standardisation of tools.

Training. Training for forensic investigators can be an issue, as investigators often try to apply digital forensic techniques that are not suited or applicable to cloud systems. This stems from a lack of training and expertise in cloud forensics for both investigators and instructors.

While each of the different branches of digital forensics has its unique challenges, the commonalities between them allow for a standardised process for digital forensic investigations to be defined. This process is discussed in the next section.

3.8 Digital forensic process

The ISO 27043 standard describes the idealised structure for an investigation process across various investigation scenarios [12]. The process classes are the readiness, initialisation, acquisitive, investigative and concurrent process classes. These processes are analogous to the seizure, acquisition, analysis and reporting steps in other process models such as proposed by Casey [61] and Cohen [15], among many more. As ISO 27043 is the standard that is now applied to the digital forensic process, only it will be considered for the purposes of this research.

3.8.1 Readiness processes class

The readiness class includes the processes that enable an organisation to be organised in such a way that should the need for the collection of digital evidence arises, the potential of successful collection is maximised, while the cost and time of the investigation is minimised. This class of processes is optional and dependent on the individual organisations to implement readiness measures.

Four goals are associated with the readiness class: to have the digital evidence used to its maximum potential, to minimise the cost implications of the investigation, both in terms of financial cost and cost of impact to the affected system, to have minimal impact on the business processes of the affected organisation, and to have the level of information security of the organisation's systems improved or preserved. Enabling of these goals is divided into process groups. These process groups are: planning, implementation, and assessment. With each of these process groups, certain processes are accosted. For the planning group, the processes are:

- Process definitions of scenarios;
- Process for identifying potential evidence;
- Process for pre-incident gathering;
- Process for the handling and storage of data that may be digital evidence;

- Process that defines the pre-incident analysis of data that may be digital evidence;
- Process planning for incident detection; and
- Process defining the system architecture.

For the implementation processes group, the processes are:

- Implementation of the system architecture process;
- Implementation of the pre-incident gathering, handling and storage of potential digital evidence processes;
- Implementation of the pre-analysis of data representing potential digital evidence processes; and
- Implementation of the incident detection process.

Finally, for the assessment processes group, the processes are:

- Assessment of the implementation process; and
- Implementation of the results of the assessment.

All the processes are iterative and can be redone and refined to further bolster the readiness of the system. Should the need arise, these processes can aid in the initialisation processes.

3.8.2 Initialisation processes class

The initialisation processes class consists of the processes which should be in place should a digital forensic investigation need to be initialised. These processes are:

- Process for incident detection;
- Process for first response;
- Planning process; and

- Preparation process.

Once the initialisation processes have been completed, it is possible to start the acquisition of digital evidence.

3.8.3 Acquisitive processes class

The acquisitive processes class deals with the capturing of potential digital evidence for the digital forensic investigation. The processes in the acquisitive processes class are:

- Process to identify potential digital evidence;
- Process to collect potential digital evidence;
- Process to acquire potential digital evidence, as an optional process;
- Process to transport potential digital evidence; and
- Process to store potential digital evidence.

When the potential digital evidence has been acquired, the investigative processes can begin.

3.8.4 Investigative processes class

The investigative processes class deals with the actual investigation of an incident. In this class, the digital evidence is analysed, results from the analysis are interpreted, the results are reported and the investigation is closed. Thus the processes in the investigative processes class are:

- Process to acquire potential digital evidence, as an optional process;
- Process to examine and analyse the potential digital evidence;
- Process to interpret the digital evidence;
- Process on reporting the results of the investigation;

- Process to present the findings of the investigation; and
- Process to close the investigation.

With these processes completed, the forensic investigation can come to an end.

In order to compare the Forensic Ready Cloud (FRC) system to other architectures, these related architectures and strategies must be examined. In the next section a survey is done regarding these related architectures.

3.9 Related forensic readiness work

When examining the problem of forensic readiness in cloud systems, multiple approaches can be followed. In this section the work of De Marco et al. [86], Alenezi et al. [87] and, Kebande and Venter [88] is examined. This is done to determine what possible solutions these authors propose, with regard to the problem of cloud forensics. The work of other authors relating to forensic readiness is also examined.

De Marco et al. also recognise similar challenges related to cloud forensics as stated in section 3.7.5. As such a reference architecture is proposed to enable cloud forensic readiness. A cloud reference architecture is also used and is defined as the Cloud Forensic Readiness System (CFRS). The CFRS focuses on monitoring and collecting sensitive data which results in savings in time and money relating to a forensic investigation. The design is stated not to be constrained by any specific cloud architecture and does not alter the existing cloud architecture.

Alenzi et al. propose a framework regarding the factors that influence the readiness of an organisation to perform a digital forensic investigation on a cloud computing system. The framework is divided into three categories. These categories are technical, legal and organisational. Each of the categories is described below. The technical factors relate to the technical aspects relating to readiness in cloud computing systems. The technical factors include:

- Cloud infrastructure;
- Cloud architecture;

- Forensic technologies; and
- Cloud security.

The legal factors relate to the agreements between regulatory authorities, multi-jurisdictions and between providers and consumers. The legal factors include:

- Service Level Agreement (SLA);
- Regulatory; and
- Jurisdiction.

Finally, the organisational factors relate to the organisational characteristics and how its employees can enable forensic readiness. The organisational factors include:

- Management support;
- Readiness strategy;
- Governance;
- Culture;
- Training; and
- Procedures.

Kebande and Venter propose an Agent Based Solution (ABS) model in order to achieve forensic readiness in a cloud computing system. The proposed model is named the Cloud Forensic Readiness (CFR) model. The CFR model utilises a modified botnet to collect evidence in the cloud system. The agent is deployed within the cloud system by installing the agent on the VMs deployed in the cloud system. The CFR model is based to the SaaS service model in order to facilitate deployment of the forensic agent to the VMs.

Grobler et al. [89] identify six dimensions of digital forensics and also identify certain goals for proactive digital forensics. With these goals and dimensions identified, the authors propose a theoretical framework as a guide to implement proactive forensics in an organisation. Complementing this work, Elyas et al. [90] [91]

identify factors that can contribute to achieving digital forensic readiness in an organisation and develop a conceptual framework as an aid to said organisations.

Valjarvic and Venter [92] propose the Digital Forensic Investigation Readiness (DFIRP) model as implementation guidelines. This model consists of three processes, namely planning, implementation and assessment. This model was added to the ISO 27043 standard [12]. The model aims to implement forensic readiness in organisations to enable efficient and effective forensic investigations that would result in digital evidence admissible in a court of law.

Trenwith and Venter [93] propose a model for digital forensic readiness in a cloud environment. In this model, data collection is accelerated with the use of remote and central logging of data. Other forms of data that might be relevant to a digital forensic investigation, are also addressed by this model.

Sibiya et al. [94] propose a model that can be used by cloud service providers to enable forensic readiness. This model enables the cloud service provider to administer the data that might be used in a digital forensic investigation. This model is, however, limited to examining the readiness of data for forensic analysis in a cloud environment.

With the concepts of digital forensics examined, it is also necessary to examine computer security as a measure to avoid the need for a forensic investigation. A digital forensic investigation might be required due to the failure in the security of a computer system [43].

3.10 Computer security

Computer security and digital forensics are closely related; security as a preventative measure and digital forensics as a reactionary measure to an incident. Security can come in many forms and has multiple avenues of specialisation, but for this section only the core concepts will be discussed.

The first concept is the CIA triad. The CIA triad is the concepts of confidentiality, integrity and availability [95]. These three pillars can be considered the core principles of information security [96]. Confidentiality relates to the security of information which is enforced by applying appropriate access levels to information. This is achieved by organising information into collections and categorised according to who should have access to it. An example of confidentiality management is the Unix file permissions and access control systems.

Integrity relates to data integrity. The key to maintaining data integrity is to secure data against unauthorised modification or deletion. In addition to this, should data be modified or deleted by accident, the data should be recoverable. Examples of maintaining data integrity is that of a version control system such as Git or SVN, where data can be modified by authorised users and easily recovered in case of accidental modification or deletion.

The last component in the CIA triad is availability. This refers specifically to the availability of data to users. For this to be realised, the access channels and authentication mechanisms must be of such a nature that the data is accessible to individuals authorised to access it. On the other hand, individuals who are not authorised to access the data must be blocked from doing so. Systems designed with high availability in mind, are designed to compensate for events such as power outages and hardware failures. These systems use tactics such as high availability clusters, failover systems and rapid disaster recovery [97].

Closely related to the confidentiality component of the CIA triad, is the concept of least privileges. This concept states that users must only be granted enough access to information in order for them to do their jobs [96].

In terms of network security, two paradigms are commonly used. These are perimeter security and layered security. With the perimeter approach, only the borders of the network are secured against intrusion. Focus is placed on firewall, proxy servers and polices that make network intrusions less likely to occur [96, 97]. However, little or no effort is put into securing systems within the boundary of the network. The major flaw in the perimeter approach is that, should the network be compromised, there are no further safeguards against attacks.

A better approach would be that of layered security. When the layered security approach is used, individual systems within the network are also secured. This can be done via network segmentation to mitigate possible damage if the outer perimeter of the network is compromised [96].

Finally, security of systems must be measured in terms of reactive or proactive security. This is done by measuring if a system, both in terms of policies and procedures, and technological measures, is more geared to responding to a threat or more towards preventative solutions. Along with this, it must be measured if the system relies on passive security, which makes little or no effort to prevent an attack, or proactive security, where steps are taken to prevent an attack from occurring. An example of the above-mentioned is that of an intrusion detection system (IDS). The IDS can warn a system administrator if an attempted breach is in progress, regardless of the success of the breach. It can also give vital information regarding possible attack vectors to the system, which can then be compensated for, thus preventing possible attacks before they are launched [96, 97].

With the concept of digital forensics and information security defined and examined, software and system architecture must also be investigated, specifically how architecture can be used to enable proactive forensics.

3.11 Conclusion

In this chapter, the field of digital forensics was examined. This included a brief history of digital forensics that allowed a definition of digital forensics to be synthesised. Next, digital evidence was examined, as well as the methods of acquiring such evidence via postmortem and live digital forensics.

Related to the acquisition methods are the analysis methods of reactive and proactive digital forensics. Enabling these analysis methods, is the concept of digital forensic readiness. Next, the different branches of digital forensics, such as computer, database, network, mobile and cloud forensics, were examined. The digital forensics process, as defined by the ISO 27043 standard, was also examined.

In addition to background on digital forensics, a short section regarding computer security was also included.

With this background information on digital forensics, the forensic component of the Forensic Ready Cloud (FRC) system can be defined. In order to define the FRC system accurately, an architectural process must be followed. In the next chapter, the concepts of computer software and system architecture are examined.

Chapter 4

Software architecture

4.1 Introduction

As with all complex human endeavours, some form of structure is needed to describe how such an endeavour should be approached. Such structures can include the blueprint for a building, a circuit board layout and component list for a piece of electronics, or in the case of software engineering, a software architecture. Software architecture brings some form of order to the potential chaos of designing, building and maintaining software systems. For this reason, a software architecture concerns the high level abstraction, fundamental concepts and constraints of a software system.

In this chapter, the core concepts of software architecture are examined. These concepts are the definition of software architecture, the common structural patterns of software architectures, quality attributes, architecture tactics to achieve the relevant quality attributes and reference architectures. This chapter also contains an overview of the Use-Case, Responsibility Driven Analysis and Design (URDAD) methodology [98], which is used to design the FRC system. Finally, the chapter is concluded.

4.2 Definition of software architecture

There seems to be no single agreed upon definition of software architecture. The ISO defines software architecture as *"the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and the principles of its design and evolution"* [99]. The Institute of Electrical and Electronics Engineers (IEEE) defines software architecture as *"the structure or structures of a system which comprise software elements, their externally visible properties and the relationships amongst them"* [100].

From these definitions it can be seen that software architecture is concerned with the components of a system, their visible properties and how these components interact with each other. With a working definition of software architecture established, the next section gives an overview of the concepts that make up software architecture.

4.3 Overview of software architecture

Figure 4.1 depicts an overview of the components commonly associated with a software architecture. Table 4.1 lists the components, the block number of the component in Figure 4.1 and the section in which the description is located for each component.

Table 4.1: Architecture components

Component	**Block in Figure 4.1**	**Section number**
Structural patterns	1	4.3.1
Quality attributes	2	4.3.2
Architecture tactics	3	4.3.3
Reference architecture	4	4.3.4
Integration architecture	5	4.3.5
Architecture representation	6	4.3.6
Architectural design techniques	7	4.3.7

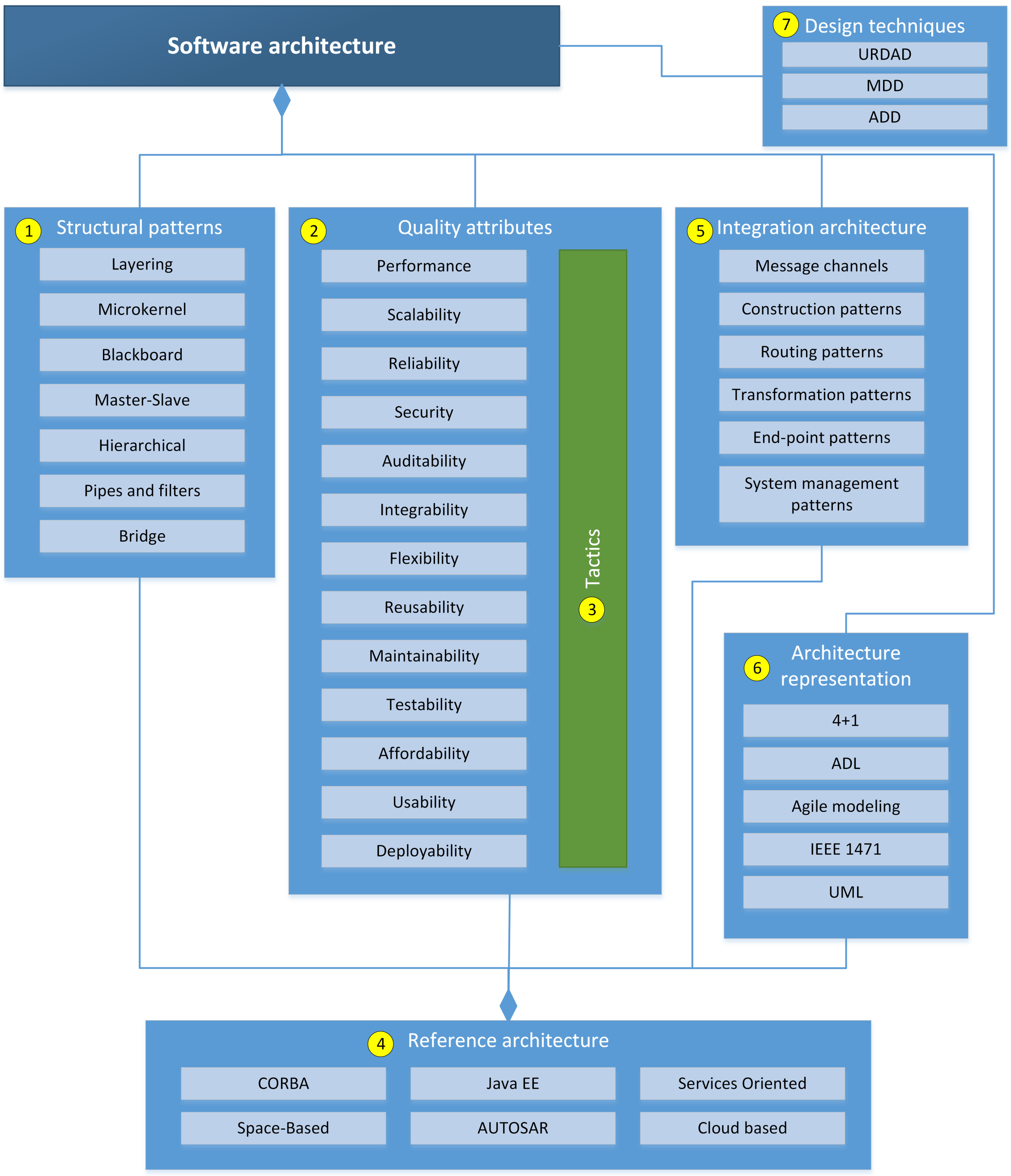


Figure 4.1: Components of a software architecture

Each of these components is examined in more detail in the following sections, starting with architectural structural patterns.

4.3.1 Structural patterns

Structural patterns (Figure 4.1 block 1) can be defined as *"a template solution for a structure which has been shown to be able to address specific architectural concerns"* [101,102]. The choice of structural patterns for the architecture of a given system will likely have wide-ranging consequences on the said system [101]. The reason for this is that the structural pattern is aligned with certain quality attributes. This results in certain quality attributes being enhanced, to the detriment of other quality attributes.

Structural patterns improve the understanding of the software architecture, as certain patterns are known to focus on certain quality attributes. This also eases the process of assessing software architectures and comparing alternate software architectures.

Examples of structural patterns are: layering, hierarchical, master-slave, blackboard, pipes and filters, model-view-controller and microkernel [101,102]. The patterns relevant to the FRC system are the layering and the microkernel patterns. These patterns are discussed in the sections below.

4.3.1.1 Layering

The layering pattern has the system components organised in layers, with each layer having some responsibility in the system. The exact responsibilities of each layer are dependent on the requirements of the system. The primary constraint when using the layered pattern is that a layer can only access components on the same layer or one layer down.

Figure 4.2 depicts a basic 2-tier implementation of the layering pattern. In this case, the client-server model is used. The client layer contains the client application which facilitates the human-computer interaction with the system. The client later, in turn, connects to the server layer via, for example, a web server. Within the server layer, the web server connects to a database server for data persistence.

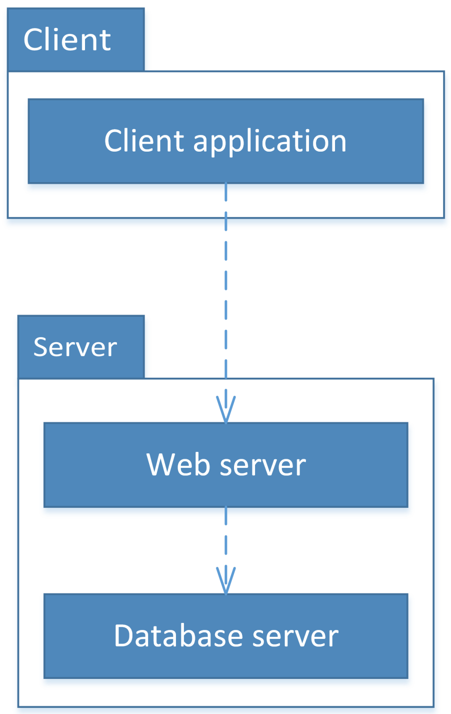


Figure 4.2: Layering structural pattern

Some of the benefits of the layering pattern include [101]: having pluggable layers, improved cohesion in that each layer has a defined responsibility, resulting in a reduction of complexity, reusable high-level components and improved maintainability. Conversely, there are also a number of constraints associated with the layering pattern. These constraints are inflexibility due to the structure of the layers and communication constraints, communication overheads due to the possible need for re-coding data for communication between layers, and high maintenance cost due to the impact changes that lower layers have on higher layers.

4.3.1.2 Microkernel

The microkernel pattern consists of a service bus that connects internal and external services, via adaptors, that facilitates client or user interfacing [102]. Figure 4.3 depicts a simple version of the microkernel pattern. The service bus forms the backbone of the system. The servers from which this backbone is constructed are assumed to be stable, reliable and slow to be changed. Similarly, the internal services form the core of the system functionality. The external services generally provide

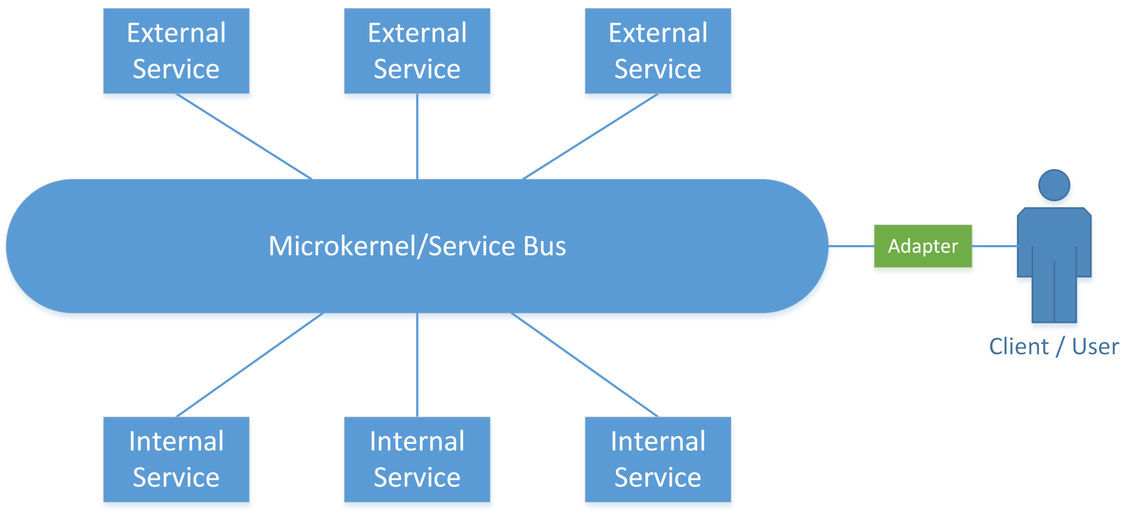


Figure 4.3: Microkernel structural pattern

higher level services and must thus be more flexible as they are more likely to be client facing. A single access point is provided to a user via an adaptor into the system.

The most popular implementation of the microkernel pattern is the Service Oriented Architecture (SOA). In the case of SOA, the Enterprise Service Bus (ESB) is the implementation of the microkernel. SOA is discussed in more detail in section 4.3.4, as it serves as reference architecture.

The microkernel provides certain distinct benefits as it provides for flexible client facing services [102]. The first benefit is that of simplified integration. The different services need not integrate with each other, only with the service bus. Next, flexibility is improved due to the pluggable nature of the service bus that can be connected to via adaptors. Also, since adaptors are used, the technology that is used to implement the connecting systems becomes unimportant. Finally, the system is more maintainable because the low- and high-level services are separate. This separation makes changing or upgrading low-level components much simpler.

Certain trade-offs must, however, be made that present certain challenges to the microkernel pattern. Firstly, due the overhead created by the communication and data routing, performance can be degraded. Next, the service bus can be a single point failure that might degrade the system reliability. Finally, the microkernel pattern does not provide for process management, making said process management more complex.

There are many more structural patterns which offer unique attributes. These patterns are, however, not relevant to the FRC system and are therefore not included. Structural patterns offer a baseline for a software architecture, but all systems have their own unique requirements. The non-functional requirements of such a system are described as the quality attributes. These quality attributes are discussed in the next section.

4.3.2 Quality attributes

Quality attributes (Figure 4.1 block 2) are requirements that must be met for a software system to fulfil its purpose [103]. Quality attributes form part of the non-functional requirements of a software system, thus describing how said system should function. It should be noted that quality attributes should be specific as to how the requirement must be addressed.

When dealing with software architectures, certain quality attributes are emphasised, depending on the requirements. The full set of quality attributes are: maintainability, reuseability, integrability, performance, reliability, scalability, security, testability, auditability, flexibility and affordability [101]. From this list of quality attributes, auditability, security, integrability, and affordability are relevant to the FRC system and are briefly examined below.

Auditability. Auditability is measure of how simple it is to acquire accurate information from a software system to perform an effective audit. Such an audit can be of the system itself or the information contained within the system.

Security. Security or securability is a characteristic of being able to secure a system. This includes the ease with which security measures can be introduced into a system, while maintaining the usability, having multiple levels of security and the effectiveness of security measures.

Integrability. Integrability is a characteristic of being able to aggregate sub-systems into a functional full system that fulfils some set of requirements. The scale can be from the integration code level components to the integration enterprise level software packages into the infrastructure of an organisation.

Affordability. Affordability refers to the cost of some computing component, be it a piece of hardware or software. It is a measure of the cost, in terms of initial acquisition cost, cost of operation and cost of maintenance.

In order to achieve the defined quality attributes of a software architecture, certain concrete implementation methods are defined. These methods, or more appropriately architectural tactics, are discussed in the next section.

4.3.3 Architecture tactics

The concept of architecture tactics (Figure 4.1 block 3) was introduced by Bass, Clements and Kazman, and they define architecture tactics as a *"design decision that influences the control of a quality attribute response"* [104]. Thus, an architectural tactic represents a design decision that addresses a certain quality attribute with the use of some concrete pattern. For example, the tactic of clustering can be used to improve the scalability and reliability quality attributes of a system.

Architectural tactics can therefore assist with the following [101]:

- Finding methods to address quality attributes;
- Explicitly making trade-off decisions;
- Gaining a better understanding of implemented architectures, frameworks, or vendor products;
- When using reference architectures and frameworks, deciding which features are applicable; and
- Communicating architectural decisions.

Architectural tactics can be applied to all quality requirements and will enable and enhance said quality attributes in some way. When applied to the FRC system, the quality attributes of auditability, security, integrability and affordability are defined as the primary quality attributes. Thus, architectural tactics are required to enhance said quality attributes. The tactics for each of these identified quality attributes are examined in the sections below.

4.3.3.1 Auditability tactics

In order for a system to be auditable, it must be possible to extract information from said system easily, accurately and in a timely fashion. The information must also be simple to interpret for an effective audit to be performed.

In order for information to be extracted easily from a system, the auditor must have adequate access to the system in question. This is handled on the policy and procedure level of the organisation that is being audited. From a system perspective, adequate access must be given in terms of access rights. This can be achieved by granting the necessary log-in credentials to the auditor. Easy information extraction will also aid in the timely completion of an audit.

Accurate information is vital to an information audit. It must be verified that the information obtained from a system is a true representation of the information contained in the system. This can be achieved by using hashing algorithms. Since each piece of information can be hashed to a unique value, a copy of the information can be verified by using the same hashing algorithm. Should the hash value be the same, it can be assumed that the information is identical to that on the audited system.

4.3.3.2 Security tactics

In terms of security tactics, three main goal categories are commonly used. These categories are detecting attacks, resisting attacks and recovery from attacks. For each goal, certain strategies can be implemented to achieve them.

Detecting attacks allows for action in a proactive manner to limit damage. The strategies associated with attack detection are event logging and analysis, auditing and monitoring of sensitive events, verification of message integrity, maintaining a database of attack signatures and scanning for them.

Should an attack occur, there must also be tactics to resist said attacks. To achieve the goal of resisting attacks, two main strategy categories are the limiting of access and limiting of exposure. The strategies for limiting access are to use authentication and authorisation mechanisms, minimise access channels and access domain, and enforce confidentiality. Similarly, the strategies to limit exposure are to add additional layers of security for valuable systems and data, ring-fence resources, minimise the use of external resources and assume such resources cannot be trusted, enforce secure default settings and, in the case of a failure, fail in a secure manner.

In the case of a successful attack, strategies are needed to recover from such an attack. These strategies can include dropping requests and connections, updating access rules and restoring to a previous secure state.

4.3.3.3 Integrability tactics

The integrability quality attribute has three common goals associated with it. These goals are publishing of provider existence, publishing of offered services and provision of access.

To achieve the first goal of publishing the provider existence, the strategies of marketing the provider can be employed. Alternatively, the provider can be published in a naming service.

Next, the services offered by a provider must also be published. This can be done by publishing the service contracts or by publishing the services to a service broker.

Finally, access must be provided for service access. Strategies for such access can be implemented by providing proxies, supporting service agents or brokers, support of standard communication channels and protocols, and support of a communication bus.

4.3.3.4 Affordability tactics

Affordability is always a concern regarding commercial computer systems. Affordability can be achieved by keeping the total cost of ownership to a minimum.

This total cost of ownership refers to the initial acquisition cost, cost of operation and cost of maintenance.

Acquisition cost cannot be addressed by architecture tactics, because it falls outside the architecture and is a business matter. Therefore tactics must be used to minimise the cost of operation and the cost of maintenance.

Cost of operation can be minimised by ensuring that licence fees for proprietary software and licences are kept to a minimum. Using open-source software, if at all possible, would be the most viable option. Efficient code can also keep the cost down, especially in scenarios where prices are charged per machine cycle.

Cost of maintenance can be minimised by introducing mechanisms to simplify processes like patching and updating. Over-the-air patching is a popular example of one of these mechanisms. It allows the patch to be run without human intervention on remote systems. On a code level, tactics such as coding standards and best practices can greatly simplify code maintenance, thus reducing cost in terms of manpower.

Just as architecture tactics address the realisation of specific quality attributes, certain well-defined reference architectures achieve this with full system architectures. In the next section, the concept reference architectures are examined.

4.3.4 Reference architecture

A reference architecture (Figure 4.1 block 4) can be defined as *"a domain-specific architectural template which aims to address architectural concerns for a particular class of problems"* [105]. In other words, the purpose of a reference architecture is to give a high-level standard model from which it is possible to design a specific solution when applied to a specific domain [106].

There are many examples of reference architectures in various problem domains. These domains can range from business software to engineering. Examples of reference architectures in different domains are:

AUTOSAR, for the software architectures in the automotive domain.

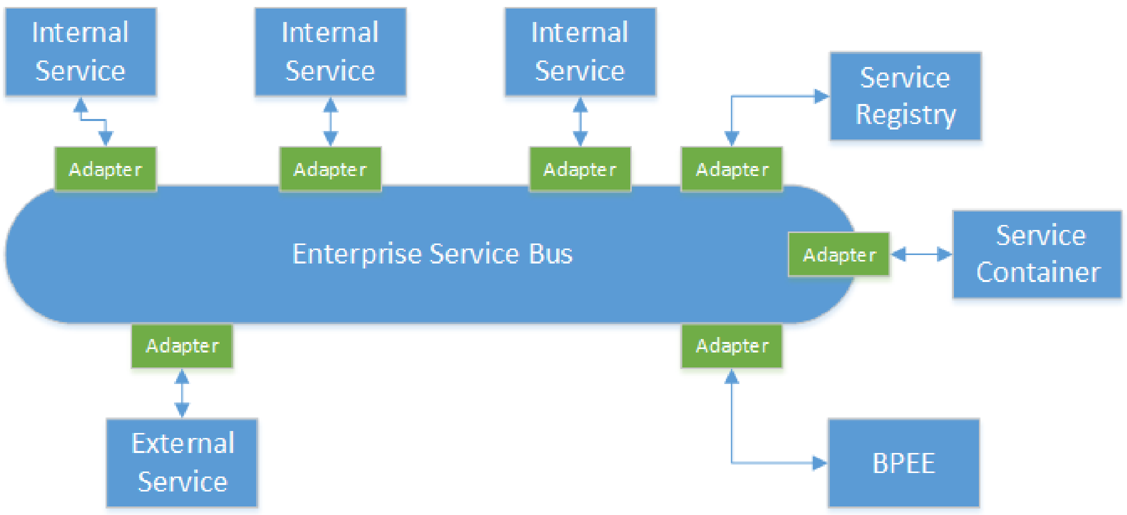


Figure 4.4: Simplified SOA reference architecture

Java EE, template solution for enterprise applications using a layered architecture. Examples of implementations are: JBoss, Glassfish and Apache Geronimo.

SOA reference architecture, for service orchestration of services that are discoverable, stateless, and self-healing. Examples of implementations are: Open ESB, Mule and Apache Axis.

These are a few of the existing reference architectures and there are many others. However, for this research, the Service Orientated Architecture (SOA) is of particular interest.

The SOA is a concrete application and reference architecture of the microkernel pattern. Figure 4.4 depicts a simplified view of the SOA reference architecture. Within SOA, the Enterprise Service Bus (ESB), along with the Business Process Execution Engine (BPEE), is the microkernel. The internal and external systems connect to the ESB where the respective services are registered in the service registry and indexed in the service container.

The main aims of SOA are to decouple systems and technologies, increase flexibility, provide infrastructure for service reuse, and to provide the infrastructure service provider with management and governance capabilities. For these reasons, the SOA reference architecture is also tailored to enhance certain quality attributes. These quality attributes are scalability, reliability, flexibility, performance, auditability, security and integrability.

While a reference architecture gives a blueprint to create an architecture, with its constituent components, to meet a certain set of requirements, some form of communication is also needed between the components. These communication methods are defined by the integration architecture, which is discussed in the next section.

4.3.5 Integration architecture

Integration architecture (Figure 4.1 block 5) is one of the primary components of any software architecture, since the decisions made are likely to have a considerable impact on the system [19]. It might have a particular impact on quality attributes such as performance, scalibility, reliability and security. As modern systems are typically running in an integrated environment, it is necessary to provide access via a range of access channels. With this comes certain challenges, including accessibility and quality requirement challenges, integration complexity and infrastructure constraints. In order to compensate for these constraints, it is necessary to apply the relevant integration concepts and integration patterns. These concepts and patterns are discussed in the following sections.

4.3.5.1 Integration concepts

The challenges of accessibility and quality requirement challenges, integration complexity and infrastructure constraints are compensated for by using concepts such as service publication, protocols, integration mechanisms and integration approaches [101]. Each of these concepts are briefly discussed below.

4.3.5.1.1 Service publication

To make services available for use, they are published with the information required on how to be able to use them. The information published is generally the service that is offered and its location, supported access channels and protocols, and the message patterns that are required for information exchange.

4.3.5.1.2 Communication protocols

Communication protocols can be defined as *"the specification of rules governing the syntax, the semantics and the synchronisation of communication* [107]. Protocols may specify the following:

- The method of finding communication end point;
- The method of how a communication session is established;
- The method of negotiating communication characteristics, e.g. compression, encryption, error detection and error correction;
- Which languages or message structures are used during communication;
- The format for a message start and end;
- The method of detecting whether or not a communication link is still active;
- Fault handling of improperly formatted or corrupted messages; and
- The method of terminating a communication session.

4.3.5.1.3 Integration mechanisms

Integration mechanisms are the global strategy that is followed in order to integrate various components within an architecture [108]. These mechanisms are:

- User-interface integration;
- Database-based integration;
- Service-request-based integration;
- Space-based integration; and
- Messaging.

4.3.5.1.4 Integration approaches

In order to implement the above-mentioned mechanisms, certain approaches can be followed. These approaches include:

- Sharing a common resource. In this case, a common resources is accessible and modifiable by one or more role players. This form of integration requires state protection of concurrent access, typically done via resource locking.

- Transferred resource. In this case, a resource is passed with either a message or a resource with request parameters. The resource might be specific for the particular service request and is commonly routed by a controller. The request can either be routed directly or passed from process to process using pipes and filters.

- Request-based integration. In the case of request-based integration, a request is sent to a service provider which then supplies the relevant service. In this case the request must be sent using a predefined format that is defined by a service contract.

- Document-based integration. In the case of document integration, the recipient is responsible for deciding how the supplied information must be processed. This method is advantageous, as it is more flexible in that multiple operations can be performed on the information as opposed to a single requested operation.

From multiple implementations of integration concepts, certain patterns emerged. These patterns are discussed in the next section.

4.3.5.2 Integration patterns

Integration patterns can be defined as *"patterns that are proven, reusable, generic solution components, i.e. they represent specific best practice solutions to integration problems"* [108]. These patterns are commonly used in combination with each other to achieve the desired integration. Key to these patterns is messaging.

Messaging has a number of distinct benefits over direct integration. These benefits are:

Load balancing. Message consumers can be added to the queue as needed.

Processing resource optimisation. In conjunction with load balancing, the number of processing elements can be utilised fully across each step in the messaging pipeline.

Reliability. Messages can be stored; they can be buffered until it is possible to deliver them.

Auditability. Due to the fact that each point in the message pipeline can be monitored, it is possible to log the inputs and outputs for each processing stage.

Decoupling. Since the message producer and consumer are unknown to one another, either may be changed should the need arise. It is also possible to have multiple producers and consumers.

Using messaging, it is possible to implement a range of integration patterns. Different classes of patterns are listed below. Each of these classes has multiple specific patterns associated with it. Hohpe and Woolf define an entire dictionary of integration patterns [108]. However, an exhaustive list of the specific patters is omitted for the sake of brevity. Some example patterns are given for each pattern class.

- Message channels, e.g. publish-subscribe channel, point-to-point channel.
- Message construction, e.g. request-reply pattern, event message pattern.
- Routing, e.g. content-based routing, message filter.
- Transformation, e.g. content enricher, content filter.
- End-point patterns, e.g. polling consumer, selective consumer.
- System management, e.g. control bus, smart proxy.

As architectures can become very complex, it is necessary to have methods of communicating said architecture accurately and efficiently. For this reason, various architectural representation techniques exist. These techniques are discussed in the next section.

4.3.6 Architecture representation

While the components of software architecture describe the system that is designed, some method must also be employed to communicate the architecture to different stakeholders. Some form of architectural representation is therefore required (Figure 4.1 block 6).

The most common technique for architecture representation is that of the so-called architectural view [109]. Different views are used to communicate different information of the architecture to the relevant stakeholder. A guide for describing complex, software-intensive systems is given in the IEEE 1471 Standard [100].

Another commonly used representation is the model proposed by Kruchten, the so-called 4+1 view model [110]. In this model, the architecture is split into views for the end-users and UX designers (logical view), programmers (implementation view), system engineers (deployment view), and system integrators (process view). What connects these views, the “+1” view, is the use of case view for analysts and testers.

The Unified Modelling Language (UML) is a general purpose modelling language that provides a standard method of visualising a system design [111]. UML has become an ISO standard for system design visualisation [112]. UML offers blueprints for diagrams to visualise architectural elements such as activities, components, component interaction, user interaction and so forth.

In the same way that UML is a formal language, there are other languages that can be used to describe an architecture. These architecture description languages (ADL) are usually domain specific and focus on the representation of components. ADLs give more functionality regarding the system behaviour by using components with properties defined and semantics of connections. The biggest drawback of ADLs is that most are very specific in their application domains.

There are, of course, many other representation methods, but these are irrelevant to this research.

With all the different components of a software architecture defined, it is necessary to have methods of systematically designing an architecture. In the next section, these architectural design techniques are examined.

4.3.7 Architecture design techniques

In order for an architecture to be designed, some method must be used that can provide a usable architecture. These methods are generally placed in two classes. The first is formal methods that will yield a design that can be proven as correct and reliable [113]. The trade-off, however, is that applying such a methodology is difficult and expensive. As such, formal methods are only applied to a class of problems where reliability and safety are of the utmost importance, e.g. aviation and medical science.

On the other hand, agile methods assume that the initial set of requirements are volatile, incomplete and possibly incorrect [114,115]. For this reason, agile methods attempt to provide a process which can operate in these conditions. While modelling is done in agile methods, the primary output is that of working code.

Model-driven development (MDD) uses both aspects from formal methods and agile methods [116]. With MDD, the agile component is used to address business and system requirements, whereas the formal component is used for model transformation and code generation.

As with MDD, the Use Case, Responsibility Driven Analysis and Design (URDAD) methodology uses aspects of both formal and agile methodologies [98]. It is a service-orientated analysis and design methodology that focuses on delivering a technology-neutral design. It also aims to provide a repeatable engineering process for generation model-driven artefacts.

The URDAD methodology requires that analyses are performed across multiple levels of granularity. At each level the requirements are examined and a design is done of high-level services that are composed of lower-level services. When the design of one level is complete, the next lower level of granularity is examined and the process is repeated. Services are thus recursively constructed until generic services are reached. These generic services can include numerical addition or data persistence.

Each level of granularity has an analysis and design phase associated with it. The analysis phase first identifies the stakeholders that have requirements for a specific use case. Next, the quality requirement, and pre- and post-conditions are defined. After the definition, the aggregated service requirements are encapsulated in a service contract which, in turn, is visualised using UML. URDAD does require that each service request has a single request and response. This is done to enhance maintainability in that, should a change request be received from a stakeholder, only the affected service needs to be changed. This isolated change mitigates problems that can arise by changing service contracts unintentionally.

When proceeding to the design phase, the pre- and post-conditions of the required service are identified and grouped with similar services. Service contracts are then assigned to the different groups and the processes to realise the services, at a lower level of granularity, are put in place. This process is repeated until the lowest level of granularity is reached. For this low level, the system can be implemented.

With the concepts of software architecture defined, it is now possible to continue with the design of the FRC system.

4.4 Conclusion

This chapter gives a high-level overview of software architecture and its various components. These components include structural patterns, quality attributes, architecture tactics, reference architectures integration architectures, architecture representations and architecture design techniques. With an understanding of these components, it is possible to generate a software architecture to fulfil a purpose. In the next chapter, models to facilitate the proactive acquisition of forensic data are proposed and compared.

Chapter 5

Models for the forensic monitoring of cloud virtual machines

5.1 Introduction

In the previous chapters, the background is discussed regarding cloud computing, digital forensics and software architecture. In this chapter, five models are proposed to address the problem as discussed in Chapter 1. The models are based on the National Institute of Standards and Technology (NIST) of the United States [6] reference architecture for cloud computing. Each of the models is defined and examined regarding its strengths and weaknesses. Table 5.1 provides a summary and comparison of all the models.

5.2 Motivation

In order to realise the goal of proactive computer forensics, one must first look at the general implementation of cloud systems and how it can be adapted to enable proactive forensics. The core tenets to enable proactive computer forensics are that of the NIST [1] reference architecture for cloud systems and the concept of forensic monitoring.

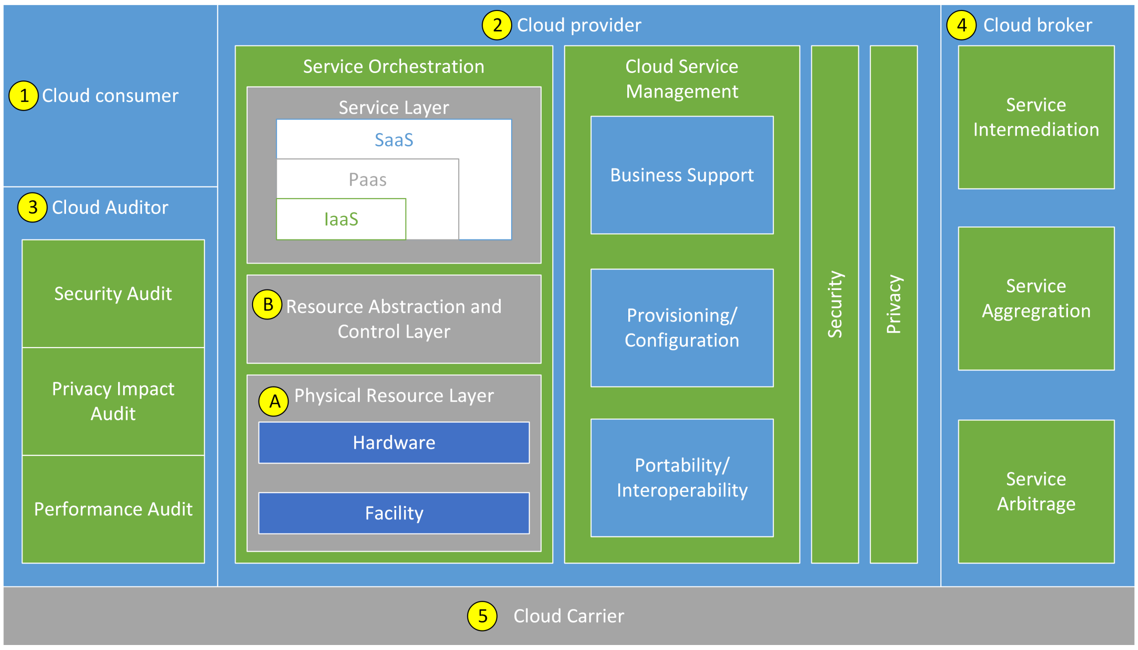


Figure 5.1: NIST Cloud reference architecture [1]

5.2.1 Cloud reference architecture

Figure 5.1 depicts the fully realised cloud reference architecture provided by the NIST. The full NIST architecture is very extensive and contains many components not relevant to this research. Thus, a simplified reference architecture is used to show only the relevant components.

Figure 5.2 depicts the simplified reference architecture of cloud computing systems. In this architecture, hardware nodes are clustered together to form one large logical machine, and thus a resource pool. The resource pool is administered by the cloud operating system. Finally, the hypervisor administers the virtual guest operating systems that can be hosted on the cloud. All of the following models use the base architecture and expand on it in order to facilitate proactive digital forensics.

Naturally, various different implementations of cloud systems have different variations on the specific structure of said systems. However, all well-known cloud systems follow the basic structure similar to that of the NIST reference architecture [1]. Thus, the NIST architecture can be used as a reference point on which to build the proactive forensic system.

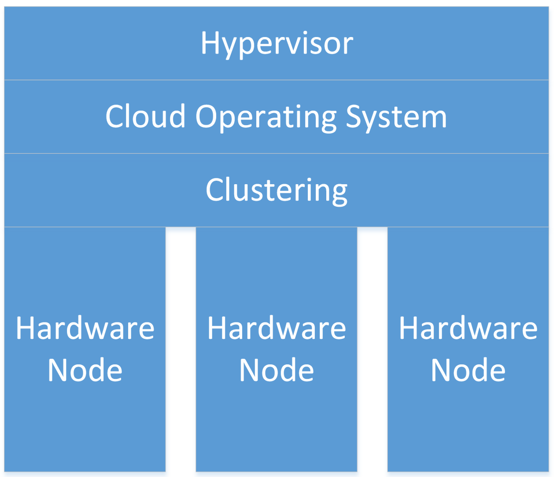


Figure 5.2: Simplified cloud reference architecture

5.2.2 Forensic monitoring

In order to enable proactive forensics, some forensic system or systems must be active during normal operation of a cloud system. This is necessary to capture data that might be relevant to a forensic investigation. A Forensic Monitor can therefore be defined as follows:

A Forensic Monitor is a piece of hardware or software that can be configured to perform some forensic function in a system while the system is running, in order to aid in a computer forensic investigation. The main function of the Forensic Monitor is to actively capture data from a system and transmit the captured data to a system where it can be analysed forensically [18, 117].

The exact working of the Forensic Monitor depends on the type of implementation of the cloud forensics system and is discussed for each of the cloud forensic models. In each case, the Forensic Monitor captures data via some mechanism, then either performs some basic forensic operation and finally transmits the captured data to a central forensic analysis system. The Forensic Monitor must be kept as lightweight as possible in terms of performance requirement, in order not to have an adverse impact on the performance of the cloud.

In order for the Forensic Monitor to be implemented, a specific implementation point within the cloud reference architecture must be identified. In the following

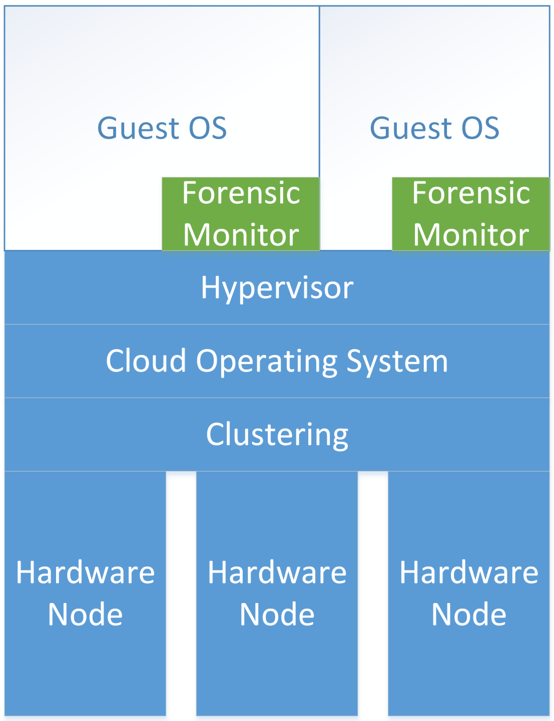


Figure 5.3: Operating system embedded forensic monitor model

section, the different options regarding implementation points are examined as possible models for the FRC architecture.

5.3 Forensic models

The proposed models are classified in terms of their implementation type, implementation complexity, level of data segregation, tamper resistance and as a service model as described in the section 1.2 Problem Statement. The models are the operating system embedded, hypervisor embedded, communication layer, single tenant forensic virtual machine, and multi-tenant forensic virtual machine. Each of these models is discussed in the sections below.

5.3.1 Operating system embedded forensic monitor model

Figure 5.3 shows the operating system embedded forensic monitor. The forensic monitor forms part of the deployed guest OS. For this method to be feasible, the guest OS must be supplied by the cloud provider. The reason for this is that the

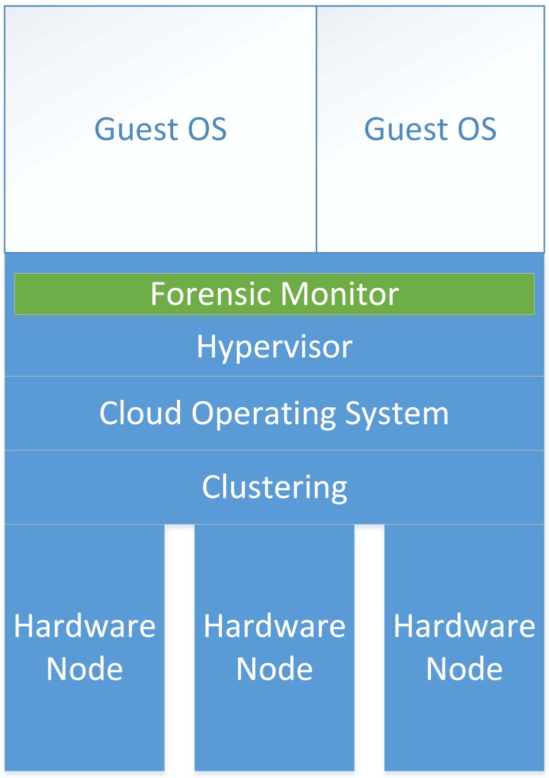


Figure 5.4: Hypervisor embedded forensic monitor model

user must not be able to access the forensic monitor lest it be tampered with or disabled. As the forensic monitor is integrated with the guest OS, it gains access to the functions provided by the guest OS. It would thus be possible to capture the relevant forensic data via the guest OS and store it for later use. In terms of data segregation, this model is well suited, as the forensic monitor is deployed within each deployed guest OS. Should the forensic data be required, it can be uniquely identified by its originating OS. This model is most suited to the software-as-a-service (SaaS) data delivery paradigm, as the user only consumes the software that is delivered while having no access to the underlying OS. This will hold true only if the OS is secured. The OS might still be vulnerable to attack via exploits such as worms, viruses, etc.

5.3.2 Hypervisor embedded forensic monitor model

Figure 5.4 depicts the hypervisor embedded model. With this model, the forensic monitor forms part of the hypervisor. As the guest OSs must communicate via

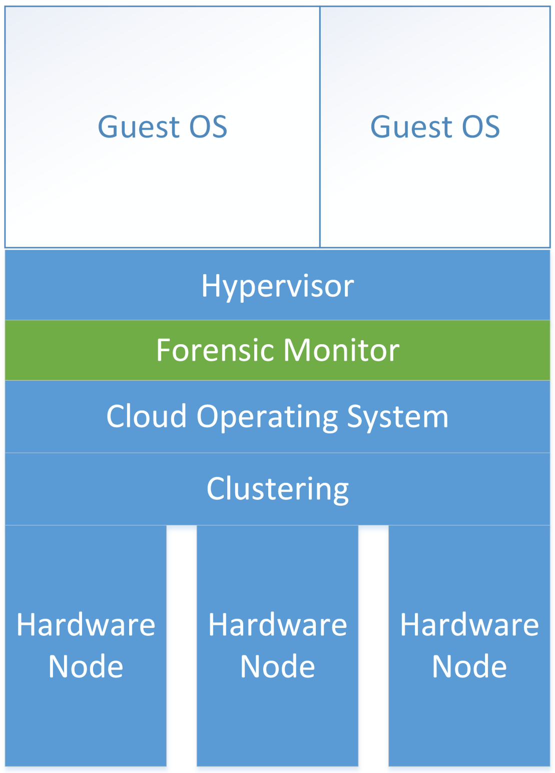


Figure 5.5: Communication layer forensic monitor model

the hypervisor with the physical hardware, all traffic can be intercepted. The hypervisor could be engineered in such a manner that the data relevant to forensics is captured and stored. As the monitor is tightly coupled to the hypervisor, the monitor must be written as an integral part of the hypervisor. Data segregation should be considered in the design of the forensic monitor. Since the hypervisor is a multi-tenant system, the data from the various guest OSs running on said hypervisor must be clearly segregated. Since the forensic monitor is integrated into the hypervisor, the user is not limited by which guest OS to use. Unlike the OS embedded monitor model, this model is agnostic to the deployed guest VM, making it feasible for the platform-as-a-service data delivery paradigm.

5.3.3 Communication layer forensic monitor model

Figure 5.5 depicts the communication layer model for forensic monitoring. In this model, the forensic monitor is layered between the cloud OS and the hypervisor. With the hypervisor embedded model, the data has to pass from the guest OS to the hardware via the hypervisor and the cloud OS. This model places the forensic monitor as part of the cloud OS, thus forming the communication layer between the cloud OS and the hypervisor. Since all data must pass through the cloud OS to the underlying hardware, the data can be monitored forensically. With this model, the forensic monitor should form part of the cloud OS to enable the capture of communication between the cloud OS and the hypervisor. This would require that the cloud OS either be designed with the forensic model as part of the system, or that the forensic monitor runs as a separate application on the cloud OS. Data segregation might become a problem with this model. The data should be clearly differentiated by its originating guest OS so that it can be catalogued correctly. Should the data be mixed, it could call into question its forensic usefulness. In the case that the forensic monitor is tightly coupled with the cloud OS, this model would be suited to the infrastructure-as-a-service data delivery paradigm. This is because the forensic monitor forms part of the cloud OS and cannot be removed. However, in the case of the forensic monitor being a separate application running on the cloud OS, the platform-as-a-service data delivery paradigm would be used. This is due to the fact that as a separate application, the forensic monitor can much easier be tampered with or disabled.

5.3.4 Single tenant forensic virtual machine model

Figure 5.6 depicts the single tenant model. This model has the monitoring done by a forensic VM running on the hypervisor. The forensic VM serves as the host to the guest VM and the forensic monitor. This creates a nested set of VMs running on the hypervisor. For this model, each guest VM is hosted by a separate forensic VM. As the guest OS is contained within the forensic VM, it is completely isolated from the underlying architecture, making tampering with the forensic monitor very difficult. The forensic monitor gains access to the data of the guest VM, since the data must pass through the forensic VM in order to reach the hardware layers. In terms of implementation, since the hypervisor is agnostic to the guest OS it hosts, the forensic VM can be run atop the hypervisor. Part of the forensic VM would then contain a hypervisor to enable the hosting of the guest VM. The forensic VM

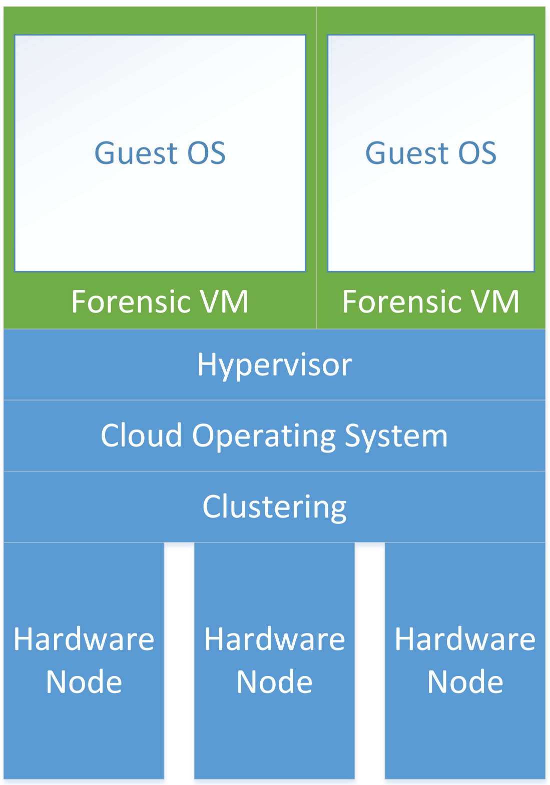


Figure 5.6: Single tenant forensic virtual machine model

OS must be written in such a way that it can fulfil its function of capturing the forensic data. A possible problem with this model could be that since the VMs are nested, the computational overhead required to run the forensic VM might cause degradation in performance. As the forensic VM contains an OS running the forensic monitor, it would allow all the functions associated with such an OS. Thus, captured data can be catalogued and stored for easy access should the need for forensics arise. With this, data can be transported to another system for safekeeping. As each guest VM runs in a separate forensic VM, data segregation is not an issue. All data is automatically kept separate between the forensic VMs, since each forensic VM runs only a single instance guest VM. As any OS can be deployed in the forensic VM, the user is not constrained in choosing a desired guest OS. Thus, this model would be well suited to the platform-as-a-service data delivery paradigm, as the user can freely select the appropriate guest OS.

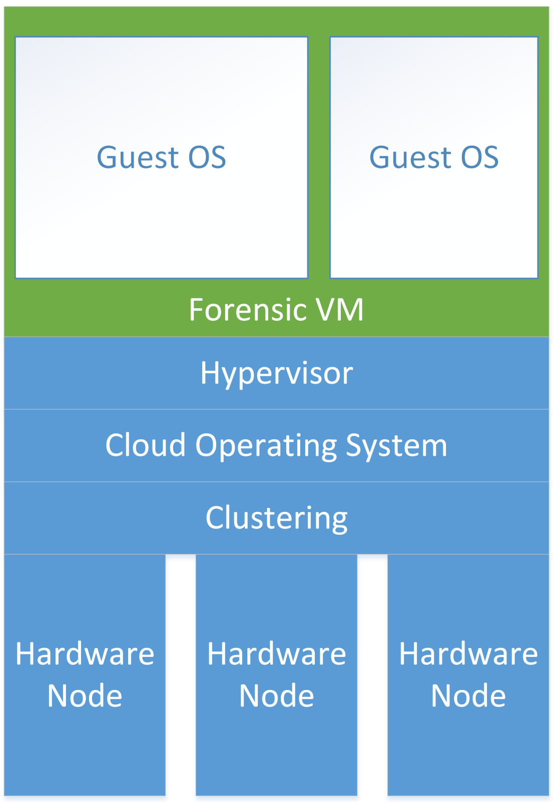


Figure 5.7: Multi-tenant forensic virtual machine model

5.3.5 Multi-tenant forensic virtual machine model

Figure 5.7 depicts the multi-tenant model. As with the single tenant model, the multi-tenant VM model has a nested set of VMs running on the hypervisor. In this case, however, the forensic VM hosts multiple guest VMs. These hosted guest VMs are isolated from the underlying architecture by the forensic VM. The forensic VM captures the relevant forensic data from the hosted guest VMs and stores it for later use. What is unique about this model is that the guest VMs can interact directly with each other, creating a virtual ecosystem. This entire ecosystem can then, in turn, be monitored by the forensic VM, as it would facilitate the communications between the guest VMs. Data segregation with this model can be an issue; since the forensic VM is a multi-tenant system, the data must pass through the forensic VM to the underlying architecture. Some method must therefore be considered to differentiate between the data from the different guest VMs. The user can decide on the OS which is to be deployed in the forensic VM, as the forensic VM is agnostic to what it hosts. Thus the multi-tenant model is well suited to the platform-as-a-service data delivery paradigm.

Having defined all the models, Table 5.1 in the next section gives a side-by-side comparison of the models.

5.4 Summary

Table 5.1: Summary of proposed models

	Operating System Embedded	**Hypervisor Embedded**	**Communication layer**	**Single tenant VM**	**Multi-tenant VM**
Implementation	Built into OS. Can be ran as an OS service, daemon or program.	Built into Hypervisor	Built into cloud OS/Application on cloud OS	Runs on Hypervisor as a VM tenant.	Runs on Hypervisor as a VM tenant.
Implementation Complexity	Medium. Dependent on the security required.	High. Must form part of the Hypervisor implementation.	High. Must be on the protocol layer requiring rewrites of said protocols.	Low. Nested VMs	Low. Nested VMs
Data segregation	High. Per OS instance, can easily be differentiated.	High (best case). Managed by Hypervisor, dependent on implementation.	Low to Medium. Dependent on implementation. All data must pass via the cloud OS.	High. One guest OS per instance of forensic VM	Medium. Multiple guest tenants in forensic VM.

	Operating System Embedded	**Hypervisor Embedded**	**Communication layer**	**Single tenant VM**	**Multi-tenant VM**
Tamper resistance	Dependent on implementation of monitor in guest OS	High. Cloud user does not have access to Hypervisor	Dependent on implementation. High when embedded. Low when a separate application.	High. Cannot be accessed or detected by cloud user.	High. Cannot be accessed or detected by cloud user
As a Service model	Software-as-a-service	Platform-as-a-service	Infrastructure-/ Platform-as-a-service, depending on implementation	Platform-as-a-service	Platform-as-a-service

5.5 Conclusion

With the description of the different models, it is now possible to design a detailed architecture that will enable proactive digital forensics. This architecture is an expansion to a full system of which the base models form part. In the next chapter, a detailed requirements analysis is performed to enable the architectural design of the proactive forensic system. This system is referred to as the Forensic Ready Cloud (FRC) system.

Chapter 6

Analysis and design of the FRC architecture

6.1 Introduction

To implement a system for proactive cloud forensics, it is necessary to define an architecture that describes the system and all its components accurately. While the proposed forensic models aid in the acquisition of forensic information, the FRC system as a whole must be modelled in order to create an implementable and technology agnostic architecture.

Cloud computing systems are dynamic by design and can be outright chaotic when viewed from the perspective of conducting a digital forensic investigation. In contrast to the possible chaos of cloud systems, the digital forensic investigation process must follow a very rigid set of guidelines. For this reason, an architecture is needed to bridge the gap between the chaos of a cloud system and the rigidness of a digital forensic investigation. To design such an architecture, some method must be followed.

The Use-Case, Responsibility Driven Analysis and Design (URDAD) method is used to define the architecture, as discussed in Chapter 4 [98]. The URDAD method is systematic and consists of an analysis phase and a design phase. The phases are iterative and can be repeated for each level of granularity. This makes it an ideal method to adapt the dynamism of the cloud to the rigidness of forensics, as the concepts at each level of granularity can be analysed and the relevant strategies can be applied to generate a workable architecture.

As an analysis is done on different levels of granularity, it is possible to ensure that each level conforms to the requirements of the digital forensic process, thus making the implementation of the system certifiable as forensically sound. In addition to this, should faults be found in the system, it is easier to drill down to the cause of such faults.

This chapter is divided into the analysis phase, which includes the goals and constraints, functional requirements, non-functional requirements and model selection, and the design phase, which includes the architectural views and integration architecture. It also contains the evaluation of the architecture, to ensure that all architecture goals are met.

6.2 Analysis phase

The analysis phase aims to elicit, verify and document the stakeholder requirements [101]. This section describes the goals, constraints, quality attributes, and functional and non-functional requirements of the FRC system.

6.2.1 Goals, constraints and quality attributes

In order to create and evaluate an architecture for the FRC, a clear set of goals and constraints must be defined. The architecture must be of such a nature that the goals can be achieved within the given constraints. In addition to the goals and constraints, the primary quality attributes must be defined and aligned with said goals and constraints. These quality attributes are used to measure the performance and alignment of the system to the defined architecture.

6.2.1.1 Architecture goals

In order for the FRC system to be successful, goals must be defined against which the architecture can be designed and serve as a guide to determine if the FRC system fulfils its intended purpose. In other words, the architectural goals can be tested, via proof of concept implementation, to determine whether or not the system meets these goals. The architectural goals of the FRC architecture are:

- To have a system that can capture data from a virtual machine deployed in a cloud environment;
- To keep the captured data in a forensically sound state, in case it is needed for a forensic investigation. Forensically sound data refers to data that is captured using recognised forensic acquisition techniques and that can be used as digital evidence by fulfilling the requirement that the data is authentic, reliably obtained, and admissible. A more in-depth definition is provided in Chapter 3 section 3.4;
- To automate the initial process, as defined in Chapter 3 section 3.8, of the forensic investigation; and
- To have a forensically sound copy of the forensic data available. Data in a cloud environment is volatile due to the shared resources used to create a cloud system. Thus, when conducting a digital forensic investigation on a cloud computing system, it is desirable to have access to data that was captured and kept forensically sound should the original be destroyed, either by accident or maliciously.

With the goals defined, it is also necessary to define the constraints under which the system must operate.

6.2.1.2 Constraints

As with all systems, the FRC system has certain constraints in which it must operate. These constraints must be catered for in the architectural design in order for the final system to meet its requirements. These constraints are:

- The FRC system must be a “bolt-on” solution for cloud systems. While it would be more effective to create a bespoke cloud system that enables easy forensics in said cloud system, for this scenario such a system would not be desirable. One of the core attributes of the FRC system is that it can be used with an existing cloud hosting system. In the case where such an existing system would be operational, changing over to a bespoke system would incur cost of running the systems in parallel while a migration is in process, and the risk of data loss during the switch-over process. The other concern would be that of the cost of developing and implementing a bespoke system. Should a bespoke system be hardware-based, it could entail significant changes to the deployment architecture of the cloud system.

- The concept of Confidentiality, Integrity and Availability (CIA) must be adhered to. CIA is one of the core tenets of computer security and is examined fully in Chapter 3 section 3.10. In the FRC system, captured data must be kept confidential, the integrity of the data must be ensured for a forensic investigation and the FRC system must be available to be used. Data that is captured might be of a sensitive nature to the owner of the data and must thus be kept confidential lest it causes damage to the owner. The data could include medical records, financial records or other potentially sensitive information. For the data to be usable as digital evidence, the data must be only that of the target system. In addition to this, the captured data must be kept secure.

- The FRC system must not unduly affect the performance of the cloud hosting system. One of the most popular pricing models used by cloud service providers, is pricing by use of processing resources [118]. For the FRC to operate, it requires a certain amount of resources from the cloud resource pool. Thus the cost of the used resources must be carried by either the cloud service provider or the user of the cloud system. For this reason, the resources used by the FRC must be kept to a minimum. Should the FRC system require large amounts of system resources, it could affect the performance of the cloud system, especially if the cloud system hosts a number of virtual machines.

With the above goals and constraints defined, the relevant quality attributes can be selected to achieve the said goals and compensate for the given constraints.

6.2.1.3 Quality attributes

In order for the FRC system to be realised, the core quality attributes must be identified. The quality attributes associated with the FRC system architecture are auditablility, security, integrability and affordability. These quality attributes are discussed below.

Auditability is the single most important attribute of the FRC system. In order to have a system that can capture forensic data and transport it to another system for analysis, an audit trail is of the utmost importance for evidentiary purposes. Should any of the chain of sub-systems in the FRC system be found to have distorted data, it could call into question the forensic integrity of all captured forensic data.

For instance, a photographic or other digital image is captured from a virtual machine and must be copied, over a network, for digital forensic analysis to be performed on it. However, because of the large size of the image, it is compressed using a lossy compression algorithm, resulting in a difference in the file that was captured and the file that was transmitted. This would break the forensic integrity of the image, as some of the information contained in the image is lost. Should the image have contained information embedded using steganography, this information would be lost. It is thus of the utmost importance that all processes that the FRC system performs must be auditable to ensure the integrity for the forensic chain of evidence.

Security is a necessary addition to ensure the auditability of the FRC system. Making the system more secure, ensures that it is resilient to attack and thus ensures the integrity of the data. This is in line with the core concepts of computer security, namely Confidentiality, Integrity and Availability (CIA). All these concepts are relevant to the FRC system, as mentioned in section 6.2.1.2.

Integrability ensures that the system is easily deployed into the relevant cloud environment. The architecture aims to be technology agnostic and thus special care should be taken to ensure that it is possible to integrate it with various cloud platforms.

Affordability is always a factor when designing and implementing a software system. In the case of the FRC system, the objective is that the entire system must be software based, i.e. no specialised hardware should be required for implementation of the system. This is in line with the constraint of having a "bolt-on" solution.

How these quality attributes are achieved by using certain architectural tactics, is discussed in section 6.3.2.3. With the architectural concerns addressed, it is necessary to define the functional requirements that describe the function of the FRC system.

6.2.2 Functional requirements

The functional requirements define what the system is supposed to do. These requirements will be the driving factor behind the inputs, outputs and behaviours of the FRC system. The acquisition of viable forensic data is the primary requirement of the FRC system. Viable forensic data is forensically sound and will stand up to scrutiny in a court of law. Forensically sound data is examined in Chapter 3 section 3.4.

The following list defines the functional requirements of the FRC system.

- Data must be captured from a virtual machine in a cloud environment for the purposes of digital forensics. Due to the problems discussed in Chapter 1 section 1.2, the FRC system must be able to capture data from virtual machines deployed in a cloud environment.
- The captured data must be kept in a state that is forensically sound, should the need for a forensic investigation arise. In order for a digital forensic investigation to be performed, the captured data must be stored in a manner that maintains its integrity. Further, the integrity must be verifiable to show that the data has not been inadvertently changed or tampered with due to

malicious intent. Should either case be true, then the evidentiary validity, and thus the utility, of the captured data can be called into question.

- The captured data must be attributed with meta data relevant to the forensic investigation. Attribution is one of the core steps of the digital forensic investigation process [12, 58], and must be done regardless of whether the system is cloud-based or not. In the case of the cloud system, because of the possible large number of VMs in the system, data must be attributed, at the very least, in terms of its VM origin, time captured, number of files captured, names, sizes and hashes of captured files. This would be the minimum amount of meta data required to conduct a worthwhile digital forensic investigation.

- The captured data must be transferred to an isolated location where an automated preliminary forensic investigation can be conducted. Since data in cloud systems is by its very nature volatile, it cannot be analysed at the point of capture. This could alter the data in some way or leave it susceptible to attack from a malicious source. The data must be transported to a dedicated secure system where an initial, automated, forensic processes can be applied. With this done, some form of basic reporting can be done to alleviate the workload of a forensic investigator.

A core function of the FRC system is the acquisition of forensic data. In order to enable the acquisition, a model for forensic monitoring, as defined in Chapter 5, must be selected.

6.2.3 Model selection

Table 5.1 summarises the different monitoring models described in Chapter 5. From this it is possible to choose the most applicable model for implementation of the FRC system. As the implementation is a proof of concept, the model that can show the most features defined by the FRC architecture should be selected. For this reason, the single tenant VM model is selected for implementation.

The reasons for this choice are: the model would enable auditability, as all captured data can be attributed directly to both the guest and the forensic VM of origin. The characteristic of direct attribution is the reason why the single tenant model is chosen over the multi-tenant model. Since the multi-tenant model hosts multiple guest VMs within a single forensic VM, the attribution of data is more challenging, because the origin of the data must be correctly attributed.

The single tenant model has a high level of tamper resistance, increasing security. It is a pure software implementation, meeting the quality requirement of affordability. Finally, the single tenant model has a low complexity regarding implementation, making it ideal for a proof of concept.

With the goals, constraints, quality attributes and functional requirements defined, it is now possible to enter the design phase and define the relevant architectural views.

6.3 Design phase

In the design phase, the service contracts are assigned to the different identified responsibilities from the analysis phase, as well as the business processes that are required to execute said contracts. To show the different components clearly, different views are used to illustrate different facets of the FRC architecture. With the defined views, architectural patterns and tactics used to realise the FRC system are selected.

6.3.1 Architectural views

The architectural views show the various perspectives of the FRC system in terms of logical layout, process flow, user interaction and physical deployment.

6.3.1.1 Logical view

The logical view describes how the different components of the system are distributed in relation to each other. Figure 6.1 depicts the full logical layout of the FRC system. Each block numbered ① to ⑤ is a sub-system or connecting system.

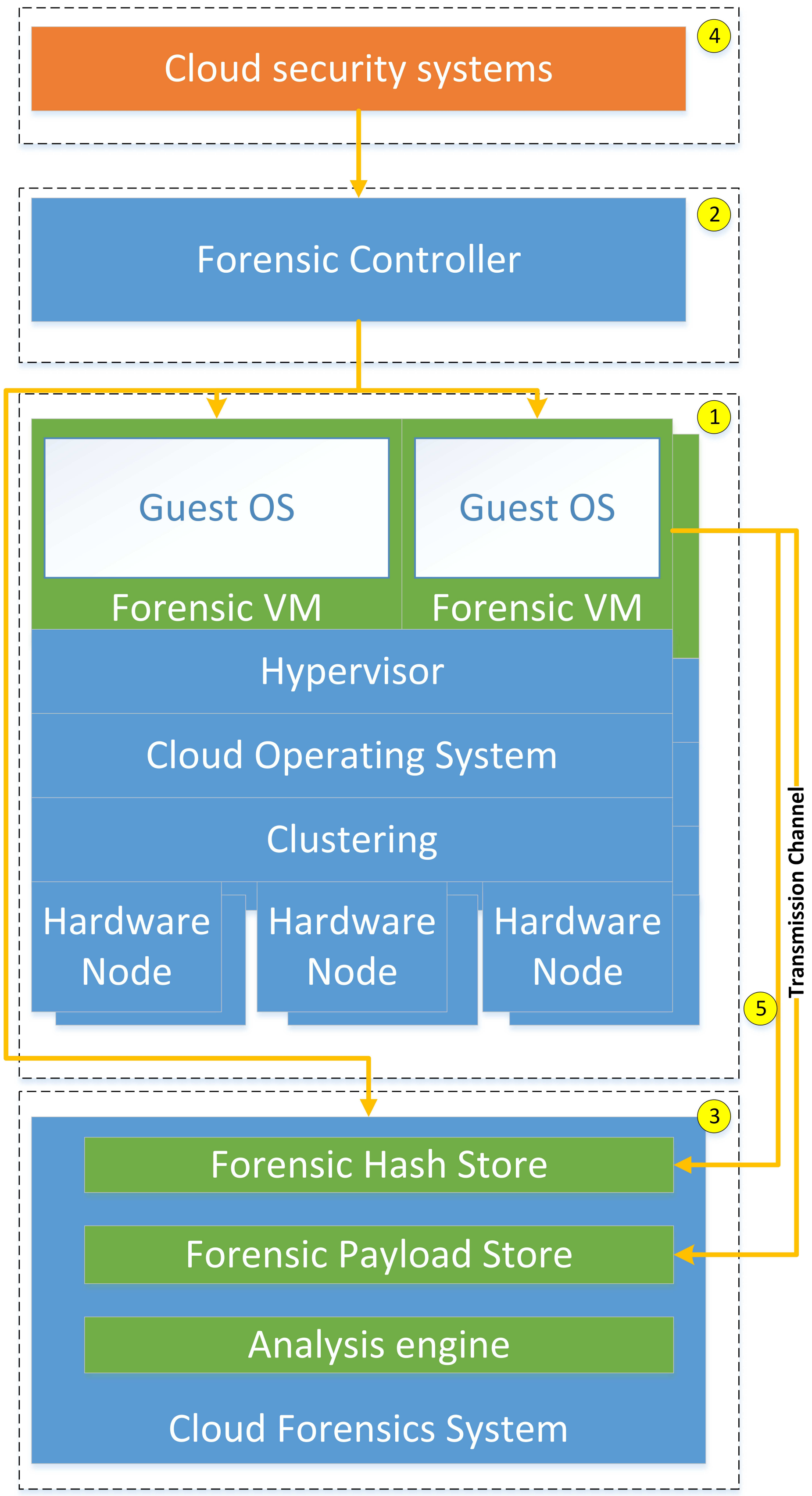


Figure 6.1: Logical view of the FRC system

6.3.1.1.1 Cloud hosting system

The single tenant forensic monitor model uses nested VMs to enable forensic monitoring and is shown in Figure 6.1 block $\textcircled{1}$. The reasoning for selecting this model is discussed in section 6.2.3. With this model, a single guest OS is contained within a single forensic VM. Thus, the forensic VM acts as a wrapper for the guest OS, isolating it from the underlying cloud structure. In this case, the forensic VM runs on top of the hypervisor. This is done so that the implementation of any existing cloud system need not be changed. This model therefore requires that all data passes from the guest OS to the rest of the cloud stack via the forensic VM. Data is intercepted and captured by the forensic VM where some automated forensic analysis can be performed. Hashing, for instance, can then take place. The function of the forensic VM is to capture data and transmit it to the cloud forensic system. This is done to minimise the required computational overhead in the cloud hosting system. The operation of the forensic VM is transparent to the user. From the user's point of view, the provisioned guest operating system runs directly on the cloud hosting system.

6.3.1.1.2 Forensic controller

The forensic controller (Figure 6.1 block $\textcircled{2}$) governs the forensic VMs in the cloud system. The forensic controller is connected to the security system of the cloud infrastructure. The controller has the function of assigning the level of forensic data acquisition to the different forensic VMs. In order to save both space and computational overhead, the forensic controller can be configured to different levels of data acquisition. At the lowest level, the VM only captures minimal data, for example log files, whereas on the highest level, the entire guest VM can be captured for forensic analysis.

This level of data acquisition is determined by the configuration set by the cloud service provider. The forensic controller receives input from the cloud security system. Should a specific VM be targeted by some external threat, the cloud security system detects the threat and signals the forensic controller. The forensic controller can escalate the level of data acquisition by the forensic VM for the targeted guest

VM. This can be done either for a single VM or for a group of VMs, depending on the detected threat. The captured forensic data is transported via a secure channel to the cloud forensics system, where automated analysis can be performed.

6.3.1.1.3 Cloud forensic system

The cloud forensic system (Figure 6.1 block ③) is the destination for data captured by the forensic VMs. It is a separate system that stands apart from the cloud hosting system. The cloud forensic system consists of a forensic hash store, a forensic payload store and an analysis engine. The forensic hash store contains the hash signatures that are generated by the forensic VMs. These hashes are catalogued and stored in such a way that it is easy to differentiate their origins. The hashes serve the purpose that, in the case of an investigation, the forensic hashes can be compared to the hashes generated by the investigator in order to determine where changes to data took place. It can also be used as a short-cut by eliminating the need to generate the hashes from scratch.

The forensic payload store contains the forensic data collected from the different forensic VMs. When the forensic VM receives a signal to increase the level of data acquisition, the acquired data is transmitted to the forensic payload store via a secure channel. The acquired data is catalogued and sorted in order to easily determine its origin from within the cloud hosting system. The data is segregated and secured should it be needed in an investigation. The analysis engine performs a predetermined analysis on the forensic data. This can be configured to such tasks as cataloguing and analysis of images, analysis of communications and anomaly detection. The proactive analysis data can save time for the investigator and speed up the investigation.

In order for the data capture process to be initiated, a signal must be received from the cloud security system.

6.3.1.1.4 Cloud security system

The cloud security system (Figure 6.1 block ④) is the standard security systems that are present in a cloud environment, for example firewalls and anti-virus systems. As these standard systems are already optimised to their purposes, it stands to reason that they should not be replaced. The only addition to this existing security system would therefore be the interface to the forensic controller.

In order for the data to be transferred from the cloud hosting system to the cloud forensic system, some form of transmission channel is required.

6.3.1.1.5 Transmission channel

The transmission channel (Figure 6.1 block ⑤) is merely the carrier that allows the captured data to be sent from the cloud hosting system to the cloud forensics system. The channel must be encrypted to ensure that possible data interception would be useless. This channel has both a physical and logical level of operation. On the physical level, the hardware infrastructure of the cloud data centre can be used to facilitate the data communication between the cloud hosting system and the cloud forensics system. This can take the form of a local area network, fibre network or some other communications medium. On the logical level, some direct connection must be made via the encrypted channel between the cloud hosting system and the cloud forensic system. This can be done via direct connection, virtual private network or some other mechanism.

6.3.1.2 Process view

The process view describes the dynamic attributes of a system. In this case, the process view of the FRC system is a flow diagram (Figure 6.2) describing different processes and the sequence of their execution. Figure 6.3 illustrates the detailed sequence of the hashing process, message transmission and message verification.

6.3.1.2.1 Process flow

Figure 6.2 depicts the process flow of the system. Starting with the initial state, input from the cloud security system, the forensic controller ① receives a signal that an intrusion was detected and the system must capture data. The forensic controller then signals the forensic VM ② of the relevant guest VM. The system proceeds to capture the relevant data ③ that is to be used in the forensic investigation. While the data is captured, hashes are generated ④ for the verification of the captured data. The hashes are stored in the local hash store ⑤, and are transmitted ⑥ to the cloud forensic hash store ⑦. The hashes stored in the local hash store are then also transmitted to the forensic hash store, where they are compared ⑧ to verify ⑨ that the hashes are identical. Should the hashes not be identical, the process is repeated from ④ and the hashes are recalculated and resent. If the hashes are identical, the hashes are written to the forensic hash database ⑩.

After the hash for a particular file has been sent, the actual captured data (payload) is also transmitted ⑪ to the cloud forensic payload store ⑫ and written to the database containing the captured data ⑬. The hash for the transmitted file is calculated ⑭ and compared to the hash value in the forensic hash store ⑮. If the calculated hash matches the hash in the forensic hash store ⑩, the file is loaded to the analysis engine ⑯ from the stored captured data ⑬, using the load captured data process ⑰. The data is then analysed ⑱ and the results are written to the forensic report database ⑲. The process is then at an end.

6.3.1.2.2 Hashing sequence

Figure 6.3 depicts the hashing sequence and data verification process of the FRC system. This process is of particular importance to the system, as it addresses the primary functional requirements.

The process begins in the forensic VM that generates a hash value via a hash generator ①. The hash value is encrypted ② and a message is built ③ to be sent to the cloud forensics system. The message is also encrypted ④. The message is transmitted ⑤ to the cloud forensics system.

When the message is received by the cloud forensics system, a receipt message is sent back the forensic VM to acknowledge receipt of the encrypted data ⑤. In the cloud forensics system, the message and hash are decrypted ⑥. The hash is stored in the forensic hash store ⑧. At the same time in the forensic VM, the payload of data is encrypted for transmission ⑦.

The payload data is transmitted ⑨ to the cloud forensic system and, as with ⑤, a receipt is sent. The payload is decrypted ⑩. Once the payload has been decrypted, a hash value is generated for the payload ⑪. This generated hash value is compared ⑬ to the hash value that was transmitted earlier and stored in the

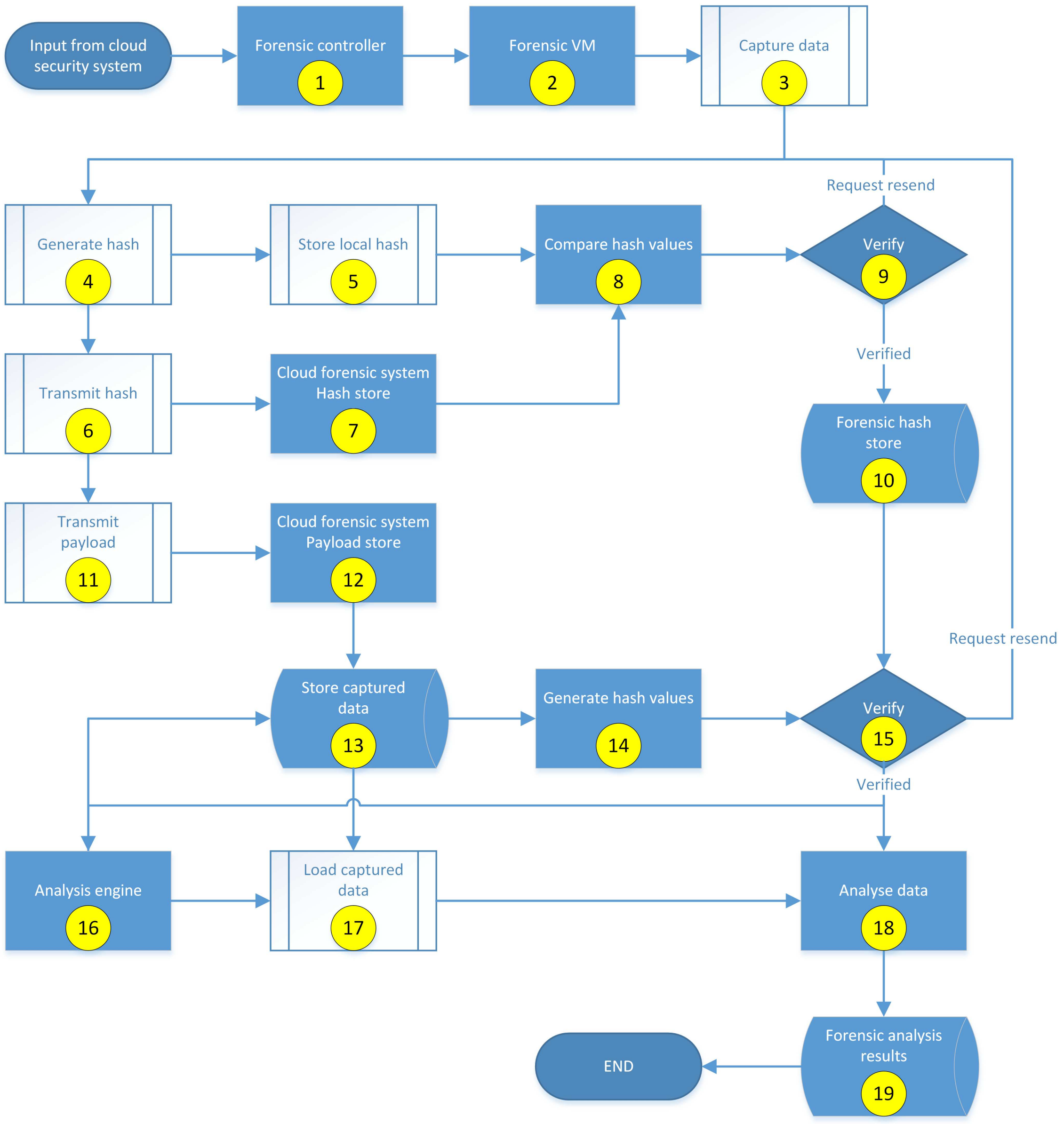


Figure 6.2: Process flow of the FRC system

hash store ⑫. Should the hashes match, the payload is stored for analysis ⑭. In the event that they do not match, the process is repeated.

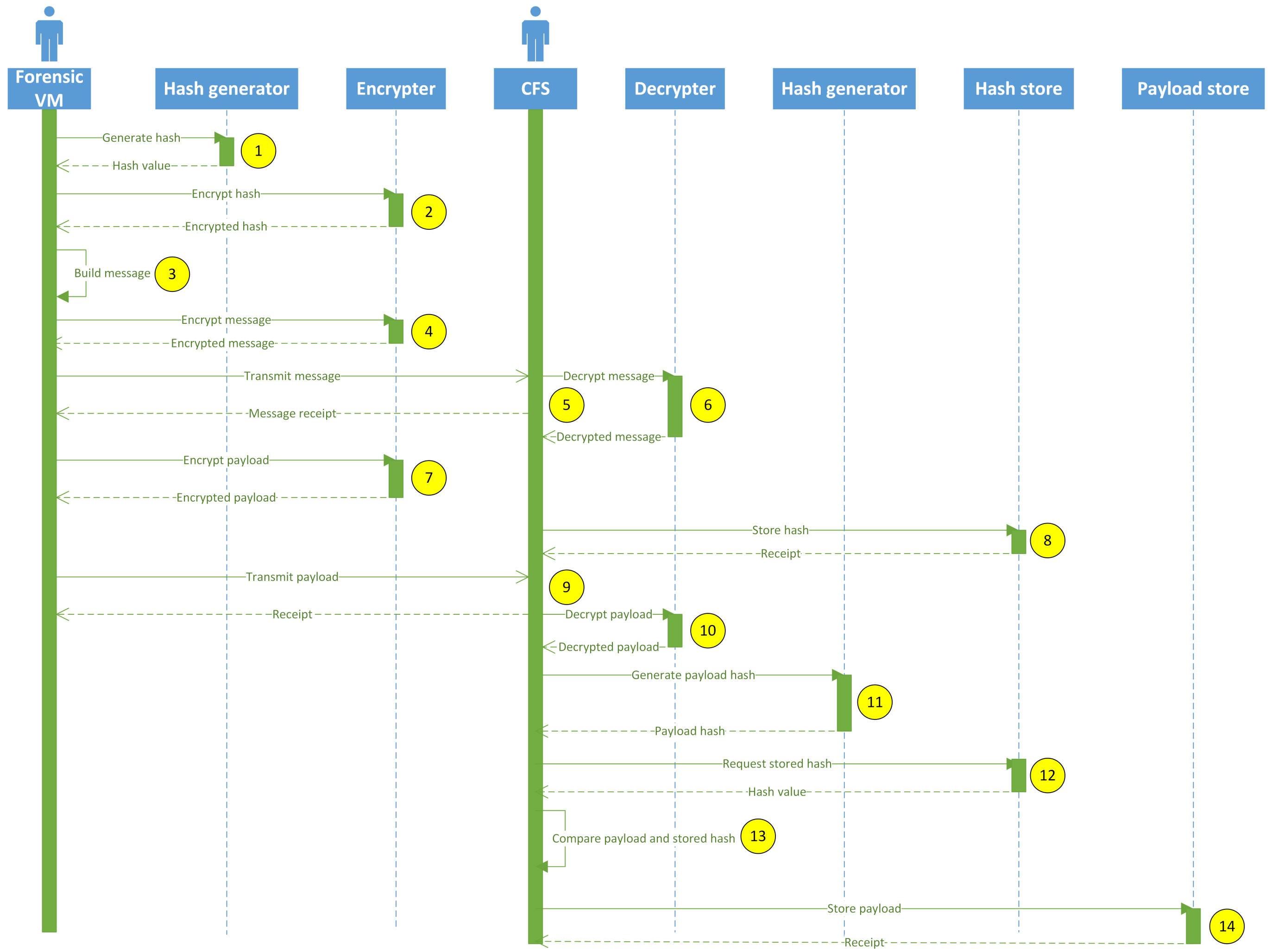


Figure 6.3: Hashing and data verification sequence diagram

6.3.1.3 Use case view

The Use Case view, in Figure 6.4, gives an overview of the major functions of the system. In Figure 6.4 block ①, the forensic controller is modelled as an external actor, as it serves as a trigger of both the use cases of the cloud hosting system and the cloud forensics system. The forensics analyst (Figure 6.4 block ④) is similarly depicted as an actor. The forensic analyst is the consumer of the output of the final use case. A brief description is given for each of the use cases.

6.3.1.3.1 Cloud hosting system

The cloud hosting system is depicted in Figure 6.4 block ②.

Capture data. Data is captured by the forensic virtual machine. This is initiated by the forensic controller.

Generate hash. Using a hash function, the hash values are generated for the captured data.

Send hash. The generated hash values are sent to the cloud forensics system.

Transmit forensic payload. The captured data is sent to the cloud forensics system.

6.3.1.3.2 Cloud forensics system

The cloud forensic system is depicted in Figure 6.4 block ③.

Request data. Data that is captured by the cloud hosting system is requested by the cloud forensics system.

Receive forensic payload. The requested data is received and stored by the cloud forensic system.

Compare hash. The hashes of the forensic payloads are compared to verify that the files were transmitted correctly.

Request hash. The hash values for the captured data is sent to the cloud forensics system.

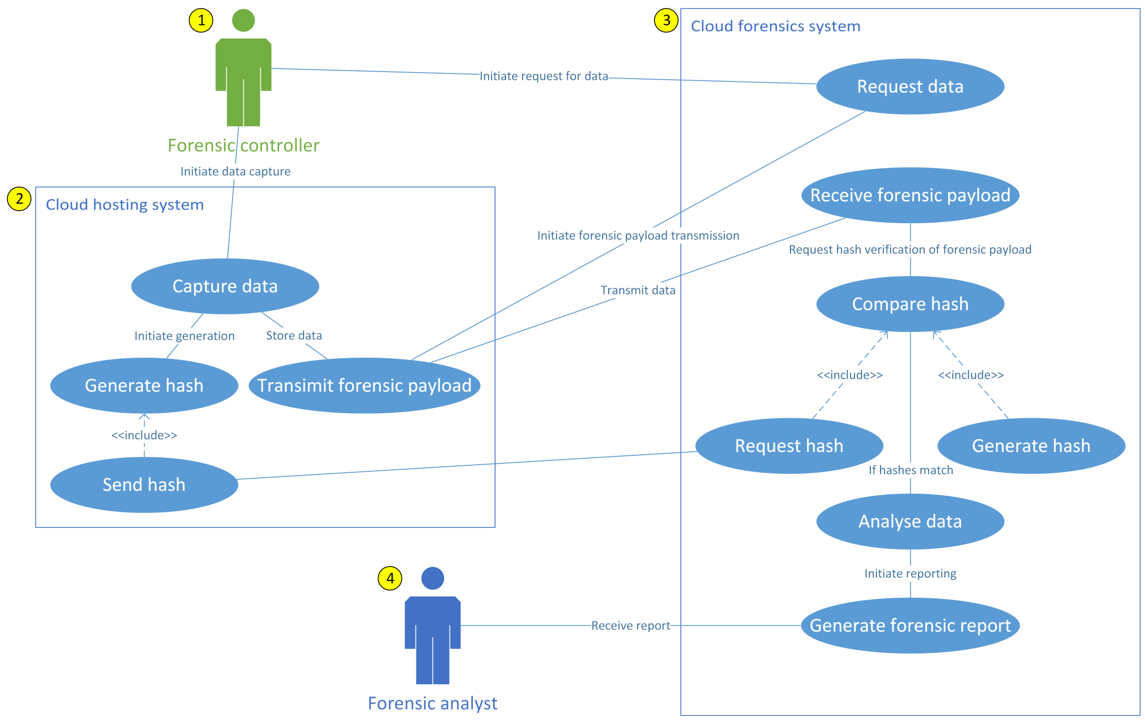


Figure 6.4: Use case view of the FRC system

Generate hash. Using a hash function, the hash values are generated for the transmitted data.

Analyse data. The data is forensically analysed for a given set of requirements.

Generate forensic report. The analysed data is compiled into a report for use by a forensic investigator.

6.3.1.4 Deployment view

The deployment view shows the physical locations of the various component of the FRC system, as depicted in Figure 6.5. The cloud hosting infrastructure contains the various cloud physical stacks. These are the physical machines on which the cloud system is built.

Each of the cloud physical stacks contains an instance of the cloud hosting system where various virtual machines can be provisioned. The virtual machines include the used provisioned guest VMs and the forensic VMs, as shown in Figure 6.1, block 1. There can be multiple physical cloud stacks within the cloud hosting

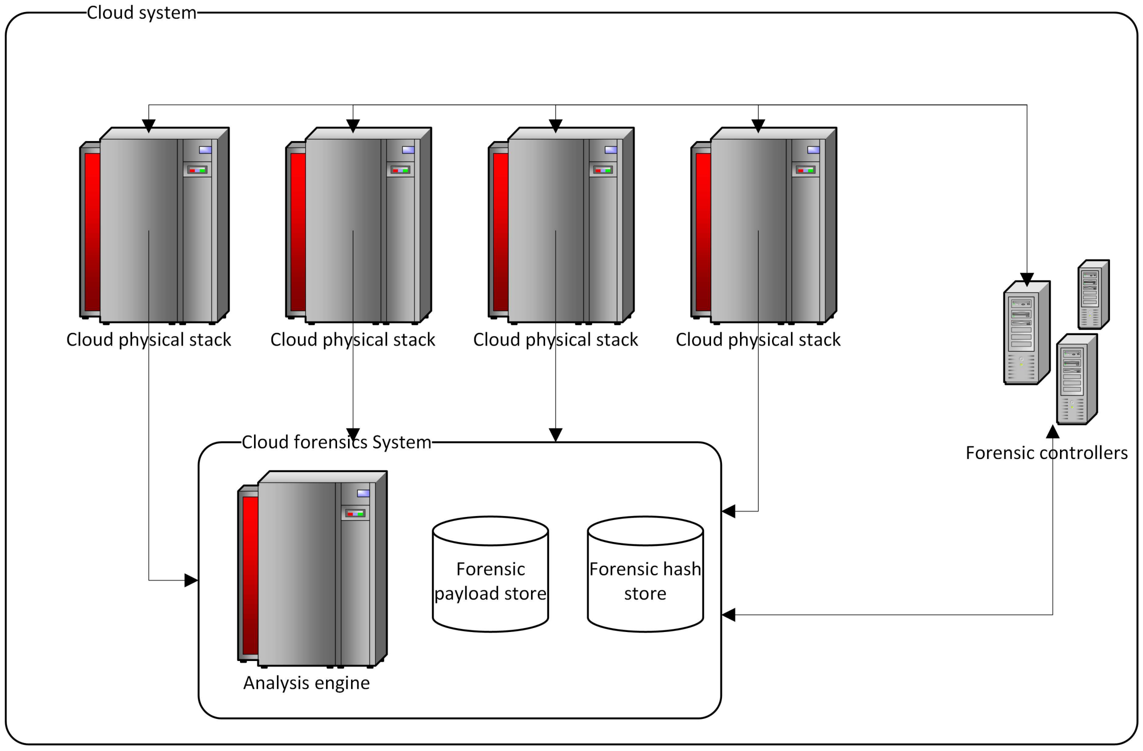


Figure 6.5: Physical deployment view

infrastructure. With all the components defined, the most applicable architectural structural pattern can be selected.

6.3.2 Architectural structural pattern and tactics

The structural architectural pattern provides a template to address challenges within a specific domain. Along with an architectural structural pattern, the relevant integration architecture and architectural tactics are also selected in order to achieve the functional requirements and quality attributes.

6.3.2.1 Structural pattern

Cloud systems follow the layered pattern, as per the NIST reference architecture [6]. However, the FRC architecture as a whole does not follow this approach. The components are distributed within the cloud environment, both logically and physically. The four main components, namely the cloud forensic system, analysis engine, forensic controller and transmission channel, would be more accurately modelled as a microkernel pattern.

The FRC is a specialised case for the service orientated architecture (SOA). The components of the FRC architecture map to SOA in Table 6.1. In the case of the FRC, the cloud forensic system and analysis engine can both be considered internal systems. The transmission channel can be considered the enterprise service bus (ESB) and the forensic controller is equivalent to the business process execution engine (BPEE). Since the FRC architecture is a specialised case of SOA, components such as the Service Registry and Service Container are omitted due to the fact that the services are fixed and do not require discovery, naming or service contracts.

Table 6.1: Mapping SOA to FRC

SOA	**FRC**
Internal system	Cloud forensic system
Internal system	Analysis engine
ESB	Transmission channel
BPEE	Forensic controller

The FRC system must be able to communicate with its components that are distributed within the cloud environment. For this to happen, mechanisms must be defined as an integration architecture to facilitate this communication.

6.3.2.2 Integration architecture

The main points of communication within the FRC system are between the following components:

- Cloud security system and forensic controller;
- Forensic controller and forensic VM;
- Forensic controller and cloud forensic system;
- Forensic VM and cloud forensic system; and

- Cloud forensic system to forensic data store.

For these components, the following services contracts are put in place.

6.3.2.2.1 Service contracts

The service contract formally defines the service and ensures that it is independent of the underlying platform or programming language. This results in the ability of SOA-based systems to function independently by handling the implementation of the service behind an interface. Table 6.2 shows the communication contracts between the FRC system components. It also shows the pre- and post-conditions, as well as the message content.

6.3.2.2.2 Integration mechanisms

As the FRC system is modelled after a simplified version of SOA, the integration mechanism is that of messaging integration. In each case of communication, the one component of the system sends the next component a message regarding the operation that must occur. Messaging is used throughout the system, as it allows for reliable communication, increased effective utilisation of system resources and possible scaling of the system.

6.3.2.2.3 Integration approach

To implement the messaging mechanism, the request-based integration approach is used. As service contracts are defined, requests can be sent to the different components to perform the services for which they are designed. In the case where payload data is transferred, the transferred resource approach is used, since the data can be of a large volume.

6.3.2.2.4 Integration pattern

In order for the messaging system to work effectively, certain message patterns must be implemented. The pattern classes and the pattern applications are shown in Table 6.3. In addition to these patterns, the basic construction of the messages passed between the forensic VM and cloud forensic system is depicted in Figure 6.6.

Table 6.2: Service contracts

Communicating components	**Pre-condition**	**Message**	**Post-condition**
Cloud security system & forensic controller	Forensic controller up and accepting messages	Begin logging for VM#	Forensic logging started for VM
Forensic controller & forensic VM	Forensic VM subscribed to forensic controller	Begin logging for VM#	Forensic logging started for VM. Acknowledgement sent to controller
Forensic controller & cloud forensic system	Cloud forensic system subscribed to forensic controller	Prepare to receive data form forensic VM#	Expecting data from forensic VM. Acknowledgement sent to controller
Forensic VM & cloud forensic system	Forensic VM is enrolled with cloud forensic system	Data sent as per process defined in 6.3	Data received. Acknowledgement sent to forensic VM
Cloud forensic system & forensic data store	Data store active setup and available	Store forensic data	Acknowledgement sent to cloud forensic system

Table 6.3: Message patterns

Message pattern class	**Pattern used**	**Components affected**
Message channels	Point-to-point	Forensic VM and cloud forensic system
	Publish-subscribe	Forensic controller and cloud forensic system
	Message bus	All components
Message construction	Event message	Forensic controller and cloud forensic system and forensic VM
Routing	Recipient list	Forensic controller and cloud forensic system
Transformation	Content enricher	Forensic VM
End-point patterns	Event driven consumer	Cloud forensic system
System management	Control bus	Forensic controller

The forensic message depicted in Figure 6.6 is the basic message that is sent, as defined by the process in section 6.3.1.2. The message contains the data hash value, VM number from which it was captured, the size of the captured data, as well as the date and time of capture.

Having defined the relevant integration architecture, the tactic of achieving the quality attributes must also be addressed.

6.3.2.3 Architectural tactics

In order for the quality attributes to be met, the relevant architectural tactics must be applied. The quality attributes that are defined for the FRC system are auditablility, security, integrability and affordability, as stated in section 6.2.1.3.

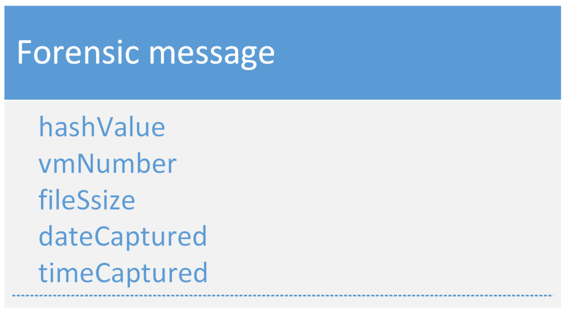


Figure 6.6: FRC data message

6.3.2.3.1 Auditablility tactics

In order for forensic evidence to be presented in a court of law, the provenance of the evidence must be clear. For a digital forensic system, this naturally also applies. With regard to the FRC system, the following tactics are to be employed to ensure the auditability of the system.

Logging of operations is key to prove that the information followed the digital forensic acquisition process as set forth in ISO 27043 [12]. With the logging of each operation performed on a piece of data, the entire chain of evidence can be established.

Logging of traffic between sub-systems is similarly required to verify that the evidence was transferred correctly and not intercepted or tampered with.

6.3.2.3.2 Security tactics

Forensic investigations can often deal with sensitive information. For this reason, the information in the FRC system must be kept secure from accidental or malicious damage. In order to achieve this security, the following tactics must be applied.

Traffic between sub-systems must be encrypted in the event that it is intercepted. Should it be intercepted, the information would prove useless to the intercepting party.

The cloud forensic system must be secure from all other systems in the cloud environment. This means that the cloud forensic system only accepts broadcast messages from the forensic controller and point-to-point connections from the forensic VMs.

6.3.2.3.3 Integrability tactics

One of the main requirements of the FRC system is that it should be a “bolt-on” system, i.e. a system that can be deployed in a cloud that already exists and is operating. Integrability is therefore important. This can be achieved by applying the following tactics.

The FRC system is software-based and will therefore not require any custom hardware. All that is required is to run the system within the cloud environment. In conjunction with this, the correct monitoring and acquisition model (see chapter 5) must be used to integrate it with the cloud system.

6.3.2.3.4 Affordability tactics

Because the FRC system is intended to be used in existing systems, the FRC system must affordable. The tactics to apply to make the FRC system affordable are the use of open-source software and libraries to implement the system, as well as avoiding the use of proprietary software that requires expensive licence fees.

Next is to make the system as simple as possible for it to achieve its requirements. Complex systems are difficult to implement and maintain. By making the FRC system as simple as possible, the cost in terms of manpower can be kept to a minimum.

In addition, to eliminate the need for custom hardware, cloud systems are usually hosted in vast data centres. To deploy new bespoke hardware in such an environment would be prohibitively costly. For this reason, the FRC system must be purely software-based.

Finally, the system must be efficient in terms of computational overhead. The FRC system uses some of the cloud resources that could have been sold as part of the cloud system, i.e. more VMs could have been run, if the FRC system was not there. Thus, the FRC system must attempt to be as efficient as possible and not waste computational power.

Having defined the full architecture, said architecture must be validated to ensure that the requirements are met.

6.4 Architectural evaluation

In order to evaluate the architecture of the FRC system, the architecture must be assessed against the following points, as given in the URDAD methodology [98]. The assessment criteria are:

- Each functional requirement must be addressed by the business process;
- Functional requirements must be grouped into responsibility domains that do not overlap;
- Service providers must be represented by service contracts and independent of technology; and
- The structure of the exchanged objects must be defined.

An architectural evaluation is done in order to ensure that all functional requirements are adequately addressed by the architectural design. With the criteria established, the FRC system architecture can be evaluated in terms of these criteria.

6.4.1 Functional requirement validation

The functional requirements of the FRC system, as defined in section 6.2.2, are addressed by the following:

6.4.1.1 Data capture

Data is captured and a hash value for the captured data is immediately generated. The message to the cloud forensic system is created with the hash value, VM from which the data was captured, as well as the time and date of capture. All this meta data is put into a message for the cloud forensics system and sent. This is all done as per the process discussed in section 6.3.1.2.

6.4.1.2 Data storage

Data payloads are stored in the forensic VM until the hash value can be validated by the cloud forensic system. When the hash value validation is complete, the data is encrypted and transferred to the cloud forensic system. Here it is validated using the generated hash values and stored in the forensic payload store for later use. This process is discussed in section 6.3.1.2.

6.4.1.3 Data attribution

Captured data is attributed with the following meta data: the hash value of the data, the VM from which the data was captured, as well as the time and date of capture. These attributes ensure that the data can be verified as forensically sound should the need for an investigation arise. Figure 6.6 depicts the message that is sent to the cloud forensic system with the data attributes.

6.4.1.4 Data examination

The payload data is transferred from the forensic VM to the cloud forensic system, which contains the payload data store. Should a forensic analyst require the data for analysis, the data can be retrieved from the forensic payload store.

The FRC system caters for all the above-mentioned requirements. Thus, in terms of functional requirements, the FRC system can be deemed fit for purpose.

6.4.2 Grouping of functional requirements

In section 6.3.1.3 and Figure 6.4, the use cases for the FRC system are defined. Each of these use cases have only one responsibility and none of the responsibilities are duplicated. For this reason, the FRC system meets the criteria for grouping of functional requirements with no overlapping responsibilities.

6.4.3 Representation of service providers

As defined in section 6.3.2.2, the integration architecture represents each service provider with a technology independent service contract. The summary of these contracts can be found in Table 6.2.

6.4.4 Structure of exchanged objects

Figure 6.6 depicts the construction of the messages sent between the forensic VM and the cloud forensic system. This structure is clearly defined and can be implemented in the FRC system.

As all the criteria are met, the FRC architecture can be seen as valid in terms of addressing the relevant requirements. This architecture can be implemented.

6.5 Conclusion

From the defined architecture in this chapter, it is possible to construct a physical implementation of the FRC system. This implementation can then be tested against the functional and non-functional requirements to determine if the FRC system meets its goals within the given constraints. In the next chapter, the FRC system is implemented and experiments are run to determine whether it does indeed meet the functional and non-functional requirements.

Chapter 7

Implementation of the FRC system

7.1 Introduction

In order to demonstrate the concept that the Forensic Ready Cloud (FRC) architecture does facilitate forensic readiness in a cloud computing system, it is necessary to implement a proof of concept (PoC) prototype. Such a PoC touches on all the major components. It should be noted that a PoC is not meant to be a fully featured system. It is the goal of the FRC system PoC to indicate that the idea of proactive forensics is feasible in a cloud environment. This is done by implementing the system with applicable technologies.

A key part of the implementation is linking the architectural designs to the different technological components. These components must satisfy both the functional and non-functional requirements, as described by the architecture. In addition, the concrete implementation of architecture tactics must satisfy the defined quality attributes.

With a completed implementation of the FRC system, experiments are run to ascertain whether the FRC system meets the functional requirements. The experiments are designed to test various aspects of the FRC system, including performance, computational overhead and utility of captured data.

In this chapter the implementation of the FRC system is examined. Firstly, the hardware configuration is given of how the FRC is deployed. Following this, the implementation of both the base cloud operating system and the various FRC software components are discussed in the software configuration section. Next, experiments are run on the FRC system to determine its suitability to proactive forensics. The results are given for each experiment and discussed in Chapter 8. Finally, the chapter is concluded.

7.2 Hardware setup

The hardware used to implement the FRC is selected to minimise variables in hardware configuration. For this reason, all the hardware used is homogeneous in terms of specifications. The hardware used is five PCs with identical configurations. Table 7.1 lists the hardware components used to implement the FRC system.

The hardware is networked using a standard Ethernet switch that connects each PC to the network switch via network cabling. The network is then configured via a DHCP server to assign IP addresses in the defined range. The range used is 192.168.0.3 through 192.168.0.253. This IP range accommodates the physical network between the hardware hosts and the virtual machines running on the cloud.

The hardware configuration was kept as simple as possible in order not to introduce unnecessary complexity into the system. Such complexity might distort experimental results. All peripherals, such as monitors, mice and keyboards, are of a generic variety that should have no impact on the system.

The hardware used for the implementation of the FRC system can be considered as standard hardware that would be found in a budget-level personal computer. Since access to actual cloud hardware, e.g. large servers with multiple CPUs and HDDs, arranged in large server stacks, is not available to the author, the selected hardware serves as a baseline analogue.

With the hardware configuration defined, it is necessary to define the software, both off the shelf and custom developed, to implement the FRC system. In the next section this software configuration is discussed.

Table 7.1: Hardware used for implementation

Hardware component	**Specification**
CPU	Intel i5 3.2 GHz quad core
RAM	8GB DDR3
HDD	500GB SATA
Network interface	1000BASE-T/100BASE-TX Ethernet
Network switch	D-link 100BASE-TX unmanaged
Network cabling	CAT5 UTP Ethernet

Table 7.2: Software used for implementation

Software component	**Specification**	**Version**
Base operating system	Centos	6.9
Cloud operating system	Apache CloudStack	4.9
Hypervisor	KVM	17
FRC components	Java EE	8
FRC database	MySQL	5.7.20
Benchmarking tools	Sysbench	0.5
Forensic tools	TSK Autopsy	4.5.0

7.3 Software setup

The basic software used in the implementation of the FRC system, the cloud hosting system and forensic tools used, are listed in Table 7.2. These software packages form the underpinnings of the implementation in order to run the FRC system. The components are the base operating system, cloud operating system, hypervisor, virtual machines, implementation language (FRC components), database (FRC database), benchmarking tools and forensic tools. The base operating system is the operating system running on each node on which the cloud operating system can be installed. Each software component is discussed in terms of its usage and implementation.

7.3.1 Operating system

The base operating system for all the nodes in the cloud hosting system is the Centos Linux distribution. Centos is used as the operating system as it is a prerequisite for the Apache CloudStack cloud operating system [119]. A standard minimalist installation is required for the cloud to be set up. Centos is also a free Linux distribution that aids in fulfilling of the affordability quality attribute.

7.3.2 Cloud operating system

Apache CloudStack is free, open-source software designed to provide public and private Infrastructure-as-a-service (IaaS) clouds [119]. The cloud hosting system is installed as per the basic installation and configuration guidelines supplied by the CloudStack user manual [120].

In broad terms, the cloud installation requires a management server, a secondary storage server and computational nodes. The management server is the core of the cloud system, as it orchestrates all the operations of the cloud. It is also the location of the management admin user interface. Figure 7.1 depicts the CloudStack dashboard after all the different components are installed and configured. From this point, the cloud is ready for operation.

The secondary storage (Figure 7.1 ①) is the location where VM templates are stored. These templates can be used to rapidly provision VMs in the cloud.

Finally, the cloud consists of computational nodes (Figure 7.1 ②). These nodes contain the VMs which use the computing resources from the node to execute their functions. Each node is seen as a computational unit with its own primary storage. The primary storage is where the VMs assigned to the node are stored.

The cloud is divided logically into zones ③, pods ④, clusters ⑤ and nodes ②. Zones encompass the entire cloud. As the cloud can be distributed over multiple data centres, zones can be very large. Pods are usually associated with a single data centre or part of a data centre. Within a pod are clusters. Clusters are a collection of nodes that operate together. Under default configuration, VMs cannot

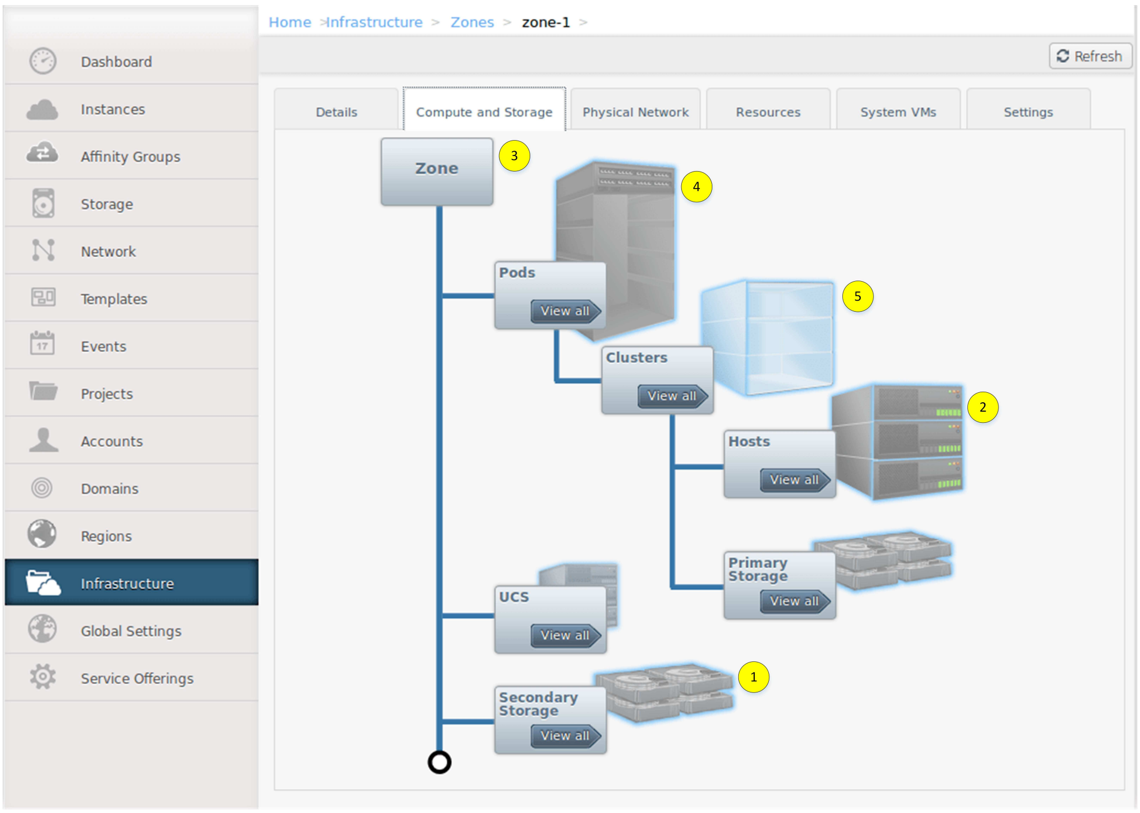


Figure 7.1: Apache CloudStack dashboard [2]

scale beyond their assigned cluster. Finally, nodes are the base building block for the cloud system.

As with the base operating system, the affordability quality requirement is fulfilled by using free, open-source software.

7.3.3 Hypervisor

Kernel Virtual Machine (KVM) is the hypervisor used within the FRC system [121]. KVM is a type 2 hypervisor that manages and optimises the interaction of the VMs with the underlying physical hardware. It manages calls for the CPU, memory, hard disk and network.

7.3.4 Virtual machines

The hypervisor provides the base for running the forensic virtual machine. The hypervisor contained in the forensic virtual machine in turn provides the base for

Table 7.3: VM specifications

Component	**Specification**	
	Forensic VM	Guest VM
CPU	3.2GHz 2 cores	3.2GHz 1 core
HDD	30GB	15GB
RAM	4GB	2GB
Networking	100MB/s Bridged VLAN	100MB/s Bridged VLAN

running the guest virtual machine. Table 7.3 shows the specifications for both the forensic and guest VMs. Since the guest VM is hosted by the forensic VM, the specifications of the guest VM must be less than that of the forensic VM. In addition, the forensic VM must have sufficient resources available to run the forensic components of the FRC system which is dedicated to each forensic VM.

7.3.5 Database

The database used for the FRC system is MySQL [122]. MySQL is a relational database which is used to store the meta data of the captured files by the forensic VM. On the cloud forensic system side, the database is used to store both the meta data and received payload data.

7.3.6 Implementation language

The implementation language used for the FRC system is Java Enterprise Edition 8 (Java EE8) [123]. Java is a general purpose object-orientated language with extensive libraries and support. All operations of the FRC system are coded in Java, as it is the most widely used business implementation language, making it easy to support. It is also possible to run Java on practically any hardware due to the use of the Java Runtime Environment (JRE) and Java Virtual Machine (JVM).

Of particular interest is the Java Messaging Service (JMS). The JMS allows for loosely coupled distributed communication. Software components can communicate by sending messages. JMS offers both publish-subscribe and point-to-point communication. Both these patterns are used in the FRC system.

The publish-subscribe pattern allows the sender of messages, or publisher, to send messages to receivers of messages, or subscribers, without the need to know who the publisher or subscriber is. The published messages are categorised into message classes and any subscriber can receive messages that are of interest to him or her. In the FRC system, the forensic VM automatically subscribes to the message class defined by the forensic controller. The point-to-point model, in contrast, sends messages directly from one sender to one receiver. For this to happen, both the sender and receiver must be known to each other.

7.3.7 Benchmarking tool

The benchmarking tool used is Sysbench. Sysbench is a common Linux-based benchmarking tool that is common to all Linux distributions. This allows benchmarks to be run on many different configurations and is able to compare the results directly. Sysbench supports benchmark tests for CPU, file input and output, and MySQL. All these factors are important in the FRC system, making Sysbench an ideal tool.

7.3.8 Forensic tools

The Sleuth Kit (TSK) is a free, open-source digital forensic investigation tool [124]. TSK is run as a command line tool used to analyse disk images. Autopsy is the graphical user interface for TSK. Because TSK is a set of command line tools, predefined analysis scripts can be written. These scripts can then be executed when new data must be analysed. TSK also supports plug-ins, which include plug-ins for analysing individual files and virtual machines.

Having selected all the different tools for the implementation of the FRC system, it is possible to describe the actual implementation.

7.4 FRC components implementation

The various architectural components of the FRC system are described in Chapter 4 section 6.3.1. These components are the forensic virtual machine, forensic controller, cloud forensics system, forensic hash store and the transmission channel. The cloud security system is included in this list for the sake of completeness, although it is not a component of the FRC system. It is also important to show the physical deployment of the various components, as it gives the context of where the different components are located and how they interact.

7.4.1 Physical deployment

Figure 7.2 depicts the physical deployment of the cloud environment and the deployment of the FRC system. The cloud environment ① contains the management server ③, secondary storage ④ and cloud nodes ⑤. Each cloud node contains forensic VMs that, in turn, contain the guest VMs ⑥. Each node also contains primary storage ⑦, where the VMs and their data are stored.

The FRC system ② is separate from the cloud environment. The FRC system contains the forensic controller ⑧ and the cloud forensics system ⑨. The cloud forensics system contains the analysis engine ⑩, the forensic payload store ⑪ and the forensic hash store ⑫.

While the FRC system is separate from the cloud environment, there is a point of overlap at the point of the forensic VM. Because data must be captured from the cloud environment, this overlap is necessary to enable to FRC system to capture said data. The dashed line in figure 7.2 ⑥ depicts this overlap of the cloud environment and the FRC system.

Given the deployment of the various components, each can be described in more detail.

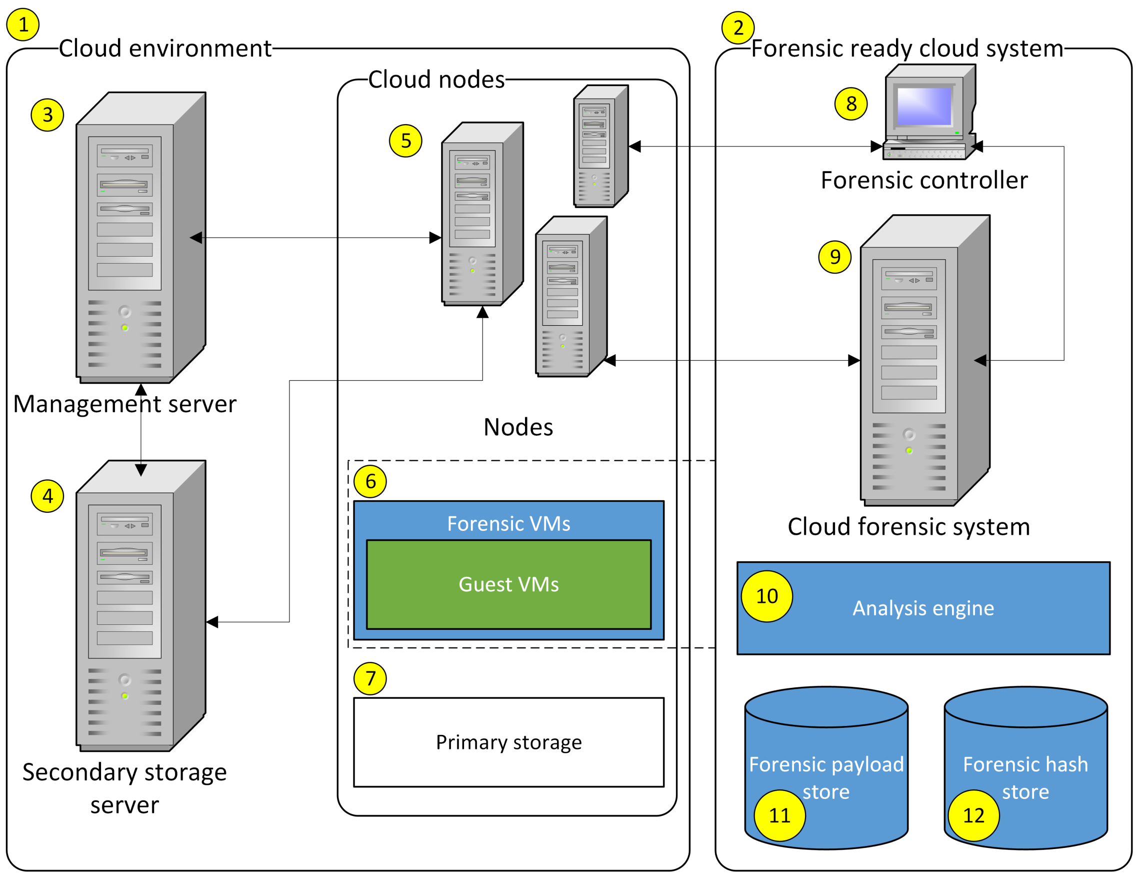


Figure 7.2: Physical deployment of the FRC system

7.4.2 Forensic virtual machine

The forensic virtual machine (Figure 7.2, block $\textcircled{6}$) is implemented as a VM template on the CloudStack system. The forensic VM template is stored in the secondary storage of the CloudStack system, ready to be provisioned should it be required. The template is set up that a guest virtual machine is already set up whenever the template is provisioned. This has the result that whenever the user provisions a guest VM, it is actually the forensic VM that is provisioned with the guest VM running inside it.

For the implementation of the FRC system, the forensic VM template and guest VM are configured with Linux Centos 6.9. Each different operating system would have a template that would create a forensic VM and nested guest VM. It should be noted that the guest forensic VM and guest VM need not have the same operating system.

The connection of the forensic VM and the guest VM is achieved by pre-configuring a network file system (NFS) share. For the implementation of the FRC system, this NFS share is fixed; however, for a full commercial system, it would be advisable to make this customisable for the user to select which data is to be forensically logged.

In order for the forensic VM to be able to connect with the cloud forensic system, it must receive the enrolment information from the forensic controller. Using a publish-subscribe model, the forensic VM is pre-configured to be the subscriber to the messages that the forensic controller publishes. When the forensic VM starts up for the first time, it receives the published message from the forensic controller and replies with its name, size and network information. The forensic controller notifies the cloud forensic system and sends the connection information of the cloud forensic system to the forensic VM. The forensic VM then connects and enrols with the cloud forensic system and sets up a point-to-point connection. This is all done with the JMS libraries.

7.4.3 Forensic controller

Because the cloud security system is not in the scope of this research, the forensic controller (Figure 7.2, block ⑧) has limited functionality. In this implementation, the forensic controller is triggered manually to set active logging in motion. It is also responsible for the enrolment of forensic VMs, as described in section 7.4.2.

The forensic controller is implemented in Java, using JMS libraries. The publisher-subscriber model is used, where the forensic controller acts as the publisher and the forensic VM as the subscriber. In the implementation of the FRC, the forensic controller acts as the publisher and the forensic VM as the subscriber.

7.4.4 Cloud forensics system

The cloud forensics system (Figure 7.2, block ⑨) is implemented as programs written in the Java programming language for the communication and hashing functions, a MySQL database for the hash and payload store, and TSK as the analysis engine. Each of these components is discussed in the sections below.

7.4.4.1 Forensic hash and payload store

For this implementation, the forensic hash and payload store (Figure 7.2, block (11) and (12)) is a MySQL database. The hash value and meta data are received and stored as per the process described in Chapter 4 section 6.3.1.2. MySQL is selected due to the fact that it is a free relational database that is easy to implement and commonly used.

Figure 7.3 depicts the database schema used in the FRC system. The VM table forms the core of the schema. This table contains the names and IP addresses of both the forensic and guests VMs. For each VM record, there can be one or more payloads associated with it. The different payloads are stored in the Payload table. For each payload, there can only payload hash and meta data record stored in the Payload_Hash table. Additional meta data records that are stored are the file name and size, time of capture and time the file was received by the cloud forensic system, and finally, the hash values of the specific payload. The hash values are generated with the MD5 and SHA256 algorithms.

As per the process defined in section 6.3.1.2, the hash generated by the forensic VM must and sent to the cloud forensic system must also be stored. The received hash values and meta data are stored as Received_Hash table. As with the meta data generated from the payload, the fields stored in the record are file name and size, time of capture and the hash values generated by the forensic VM.

In all the tables, the primary key is auto-generated in order to ensure uniqueness. Using the information in the database, data captured by the forensic VM can be attributed accurately for use in an investigation.

7.4.4.2 Analysis engine

The analysis engine (Figure 7.2, block (10)) for the FRC system implementation is The Sleuth Kit (TSK). TSK is installed as part of the cloud forensic system to analyse captured disk images and files. In addition to the main TSK installation,

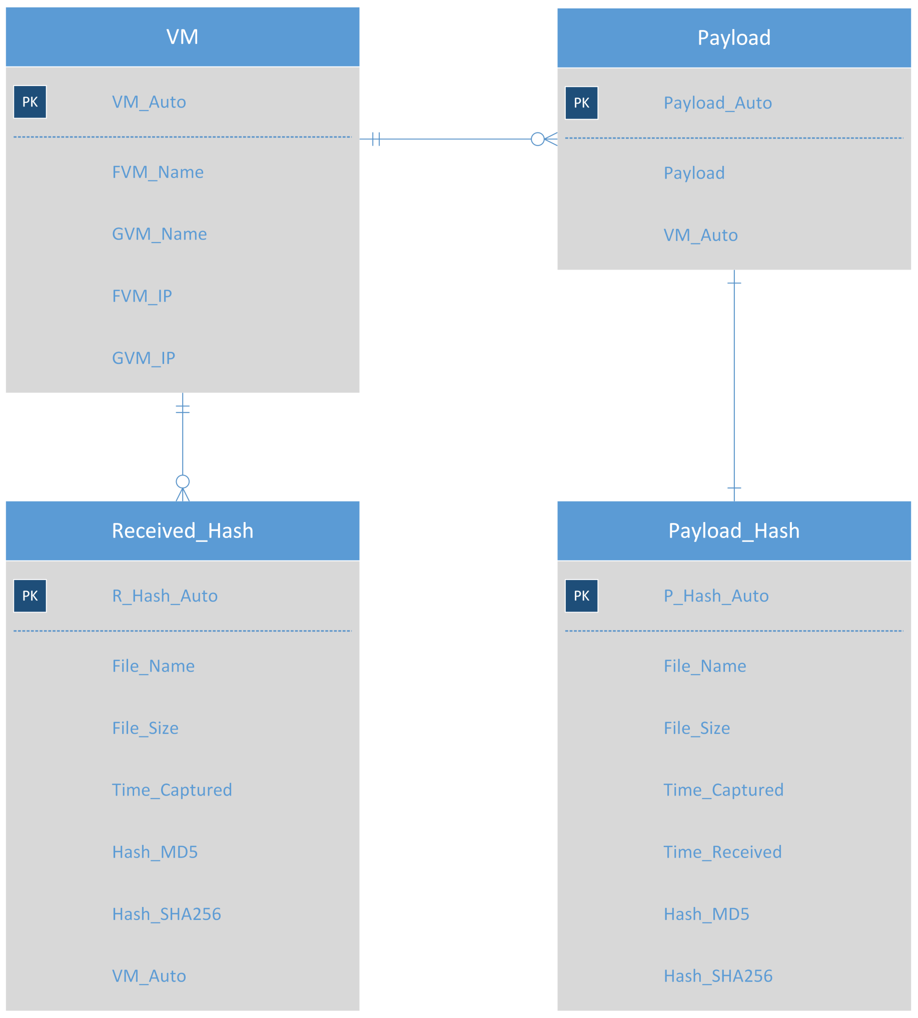


Figure 7.3: Forensic hash and payload store ERD

the plug-ins for analysing individual files and the Autopsy user interface are also installed.

When data is received from the forensic VM, and the payload is processed, the payload is piped to the TSK system using the TSK command line interface. The commands used for the command line interface are embedded in the cloud forensic system so that the system can act as a controller for TSK. In addition to the command line interface, the Autopsy user interface is also installed. This is done to allow a forensic investigator to get a more user-friendly interface when operating the cloud forensic system.

7.4.5 Transmission channel

The transmission channel is used to transfer information from the forensic VM to the cloud forensic system. In this scenario, the transmission channel is a standard Local Area Network (LAN) connection 100BASE-TX Ethernet. In a full implementation, this networking infrastructure would be much more performance-driven, as large volumes of data would have to be transferred between the various system components.

An ideal implementation of the transmission channel would be a fibre link from the individual server stacks to the FRC system. Such a fibre link would greatly improve the transmission bandwidth and speed. In addition to these performance gains, security would also be improved. While it is possible to perform a wiretap on fibre, this is a difficult proposition [125]. Any basic method that results in the fibre being cut would result in instant detection.

More advanced methods would be less effective as they rely on tactics that tap off light from the fibre. To achieve this would still require physical access to the cloud infrastructure. Also, the tapped light signal must still be decoded. As each manufacturer uses different methods of encoding in terms of phase, amplitude and polarisation, this remains a challenging option [125].

Regardless of the transmission channel implementation, all data that is transferred from the forensic VM to the cloud forensic system is encrypted. The encryption algorithm used in the PoC implementation is the AES-128 algorithm.

7.4.6 Cloud security system

The cloud security system can be the cloud firewall, DMZ area or intrusion detection system. The main function of the cloud security system in relation to the FRC system, is to provide input to the forensic controller. The DMZ or demilitarised zone of a network is the part of a network that is exposed to external and untrusted systems, e.g. the Internet. The DMZ forms a secure buffer against intrusion from these networks and the internal network of an organisation.

Having defined the software and implementation of the various components of the FRC system, it is now possible to run experiments on the FRC system.

7.5 Experiments

In order to ascertain whether the requirements set for the FRC system are met, the FRC system must be tested. The method of testing the FRC system is to run experiments against the system that indicate whether or not the goals set for the FRC system are met. In each case, the experiments are designed to test a certain facet of the system. Each experiment is divided into the experimental setup and the obtained results. The experiments are the acquisition of specific data from a guest VM, the full capture of a guest VM image, the performance of the system while data hashing is in progress, and the time in which the system can encrypt and decrypt files of various sizes.

For each experiment, some background is given regarding that specific experiment. From this background, a hypothesis is formed and the experiment is set up. The results of the experiment are given and discussed. The conclusions drawn from the experiments are discussed in Chapter 8.

7.5.1 Acquisition of data

One of the primary tasks in a forensic investigation is to acquire data from a target system. This test is designed to determine if the forensic VM system can acquire data from the guest VM and transfer said data to the cloud forensic system, while maintaining forensic stability.

7.5.1.1 Hypothesis

In order for the FRC system to meet its functional requirements regarding the acquisition and transmission of specified data, the following statement must be true.

The FRC system captures data and ensures the validity via the process of hashing. Both the meta data and actual data payloads are transmitted from the forensic VM to the cloud forensic system where the integrity is verified via hash checking. The hashed data that is generated, is meta data to the actual captured data.

It is expected that the system will capture the relevant data files, apply the hash algorithms and transfer the data from the forensic VM to the cloud forensic system. The data will then be verified by comparing the generated hash values.

7.5.1.2 Setup

For this test a sample file is created with the following attributes: file name: test.txt, file size: 128kB, contents: This file was filled with content by generating the MD5 and SHA256 hash of the string “Forensic test file”; the resulting hash values were:

```
MD5: 53e91c7a8a21a358f9831ba9d19e119a
SHA256: 046c9e7eeae441a81a523cdf17605eda
        4bc36d0bf8e22227bd5ac041935247b3
```

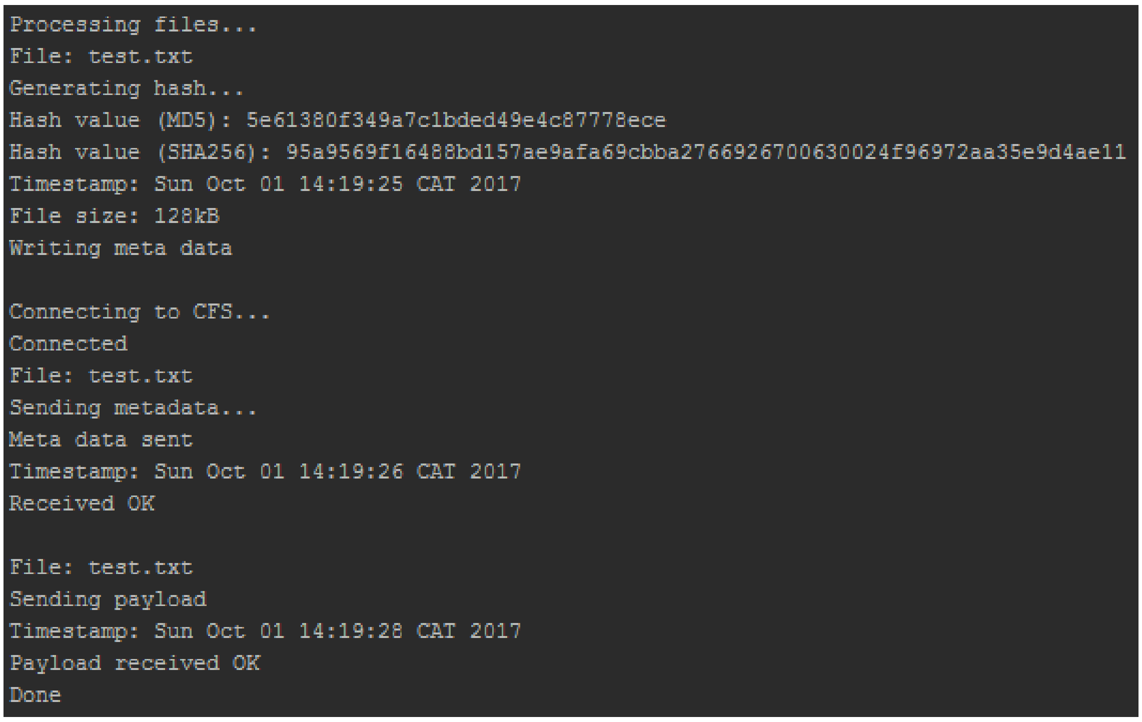
The sample file is placed in the monitored directory of the guest VM. The console output of the forensic VM and cloud forensic system is monitored to see the results.

7.5.1.3 Results

Figure 7.4 depicts the console output from the forensic VM and Figure 7.5 depicts the console output view from the cloud forensic system. These figures show how the file was captured by the forensic VM, after which it was attributed with the relevant meta data including the hash values, file size and time stamp. Following this, the meta data was transmitted to the cloud forensic service and stored in the forensic hash store. After this process was completed, the payload data was transferred to the cloud forensic system.

The cloud forensic system then validates the payload data by generating the hash values for the payload. These generated values are then compared. In this case the values match and the payload is written to the forensic payload store for later analysis.

From these results it can be seen that the file was captured successfully, attributed, hashed, transmitted, the transmission validated and the file stored for later examination.

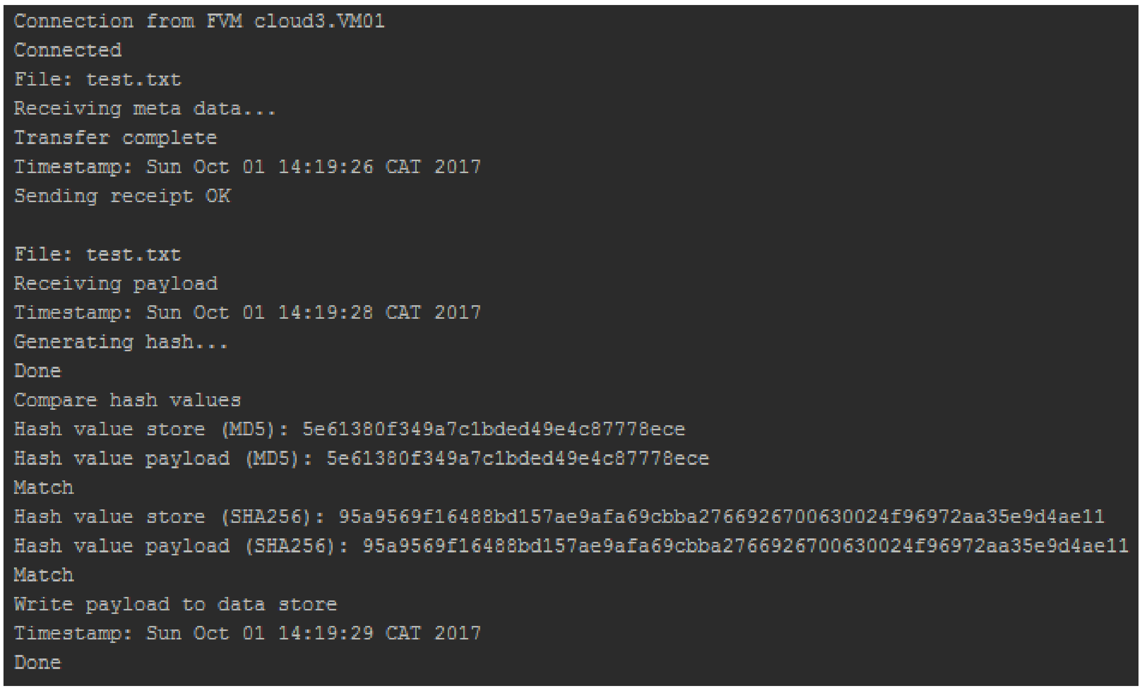


```
Processing files...
File: test.txt
Generating hash...
Hash value (MD5): 5e61380f349a7c1bded49e4c87778ece
Hash value (SHA256): 95a9569f16488bd157ae9afa69cbba2766926700630024f96972aa35e9d4ae11
Timestamp: Sun Oct 01 14:19:25 CAT 2017
File size: 128kB
Writing meta data

Connecting to CFS...
Connected
File: test.txt
Sending metadata...
Meta data sent
Timestamp: Sun Oct 01 14:19:26 CAT 2017
Received OK

File: test.txt
Sending payload
Timestamp: Sun Oct 01 14:19:28 CAT 2017
Payload received OK
Done
```

Figure 7.4: Forensic VM console view



```
Connection from FVM cloud3.VM01
Connected
File: test.txt
Receiving meta data...
Transfer complete
Timestamp: Sun Oct 01 14:19:26 CAT 2017
Sending receipt OK

File: test.txt
Receiving payload
Timestamp: Sun Oct 01 14:19:28 CAT 2017
Generating hash...
Done
Compare hash values
Hash value store (MD5): 5e61380f349a7c1bded49e4c87778ece
Hash value payload (MD5): 5e61380f349a7c1bded49e4c87778ece
Match
Hash value store (SHA256): 95a9569f16488bd157ae9afa69cbba2766926700630024f96972aa35e9d4ae11
Hash value payload (SHA256): 95a9569f16488bd157ae9afa69cbba2766926700630024f96972aa35e9d4ae11
Match
Write payload to data store
Timestamp: Sun Oct 01 14:19:29 CAT 2017
Done
```

Figure 7.5: Cloud forensic system console view

7.5.2 Full VM capture

In some scenarios it might be required to capture an entire virtual machine from a cloud system. This could be the case when a suspicious VM can be identified and isolated.

7.5.2.1 Hypothesis

In order for the FRC system to meet its functional requirements regarding the acquisition and transmission of a full guest VM, the following statement must be true.

The FRC system captures a full guest VM and transmits the captured guest VM from the forensic VM to the cloud forensic system. The captured VM is then analysed by the analysis engine of the cloud forensics system to find possible digital evidence.

The expected result of the acquisition of a full guest VM, is that the VM is captured successfully and the drive image transferred to the analysis engine of the cloud forensics system.

7.5.2.2 Setup

In this scenario, the entire guest VM is to be captured, transferred and analysed. For this test, a guest VM is created and installed with a base operating system. The guest VM is started and two digital photographic images are copied to the VM. These images are to act as the test medium to determine if the data was captured correctly. The images used are Figure 7.6 (Lena) and Figure 7.7 (Nuke).

The MD5 hash values for these images are:

- Lena: a56f716b3c97dbc45e32375a56a00a12
- Nuke: cdf39cb9eb9b1c49121af3fd4d3223a7

Four copies of each image are used as a test medium. The first is the unaltered image with an unaltered file name. The location for these files are:

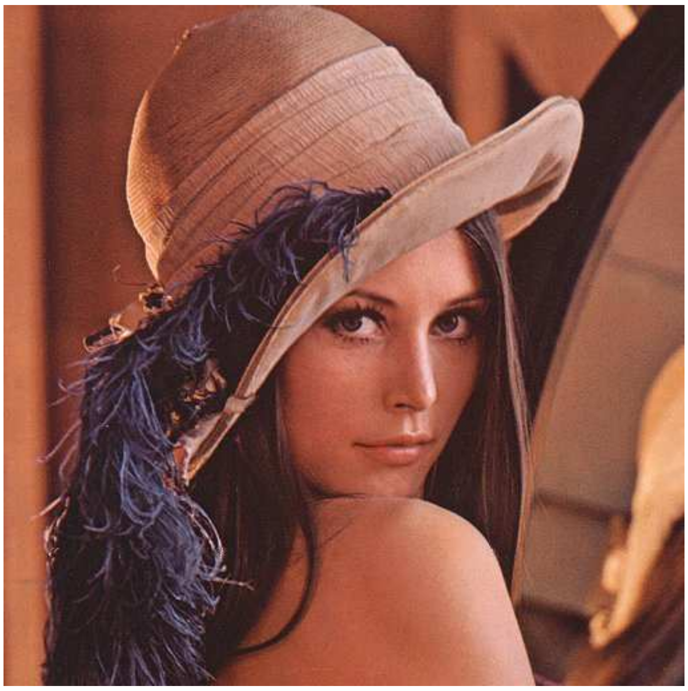


Figure 7.6: Test image 1: Lena

```
$ /home/TestDir/test1.jpg
$ /home/TestDir/test2.jpg
```

Next, the file names are changed to IMG_3920.jpg and IMG_3921.jpg for test1 and test2 respectively. The location for these files is:

```
$ /home/tmp/IMG_3920.jpg
$ /home/tmp/IMG_3921.jpg
```

After that, the file extension is changed from ".jpg" to ".pdf" for both files. The location for these files is:

```
$ /home/media/test1.pdf
$ /home/media/test2.pdf
```

Finally, both the file names and the file headers are changed. The headers of the files are changed to that of a PDF file. The samples below show the change of the file headers of the test1 file from JPEG to PDF.

```
=FF=D8=FF=E0 =10JFIF =01=01 --> %PDF-1.3
```

Figure 7.7: Test image 2: Nuke

The location for these files is:

```
$ /home/mnt/secondary/test1.jpg
$ /home/mnt/secondary/test2.jpg
```

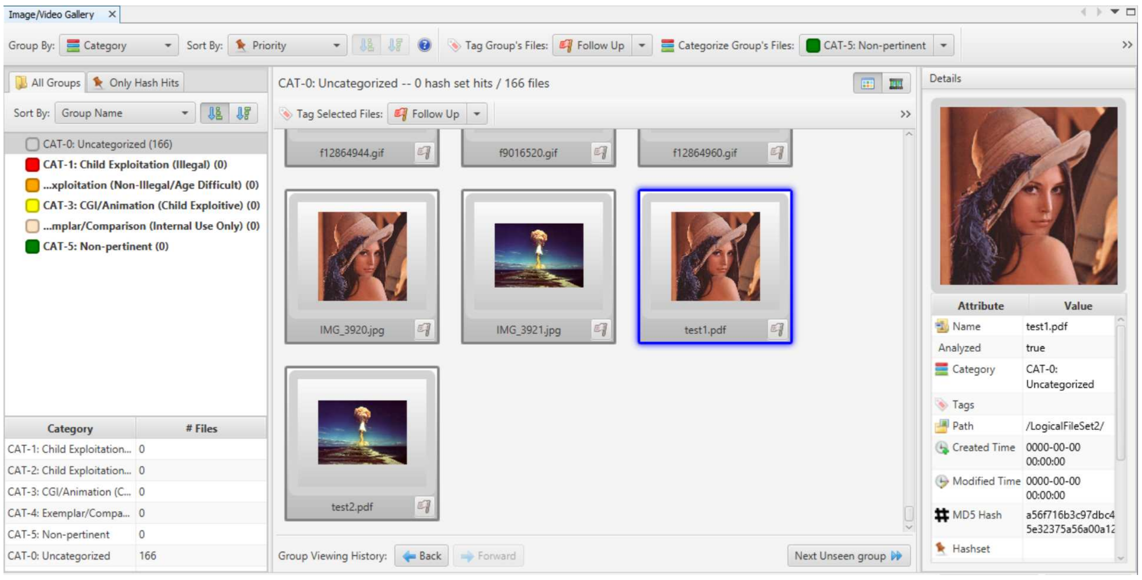
In this case it would not be necessary for the forensic VM to capture data from the guest VM. The entire forensic VM can be captured by setting the monitored directory of the forensic VM to the CloudStack primary storage directory. In this case:

```
$ /usr/cloudstack/storage/primary/
```

From here the entire VM image could be acquired in one step.

7.5.2.3 Results

The VM was copied from the forensic VM to the cloud forensic system using the dd tool [126]. dd is a command line tool that copies files at the bit level and is



CAT-0: Uncategorized -- 0 hash set hits / 166 files

- ⬜ CAT-0: Uncategorized (166)
- 🟥 **CAT-1: Child Exploitation (Illegal) (0)**
- 🟧 **...xploitation (Non-Illegal/Age Difficult) (0)**
- 🟨 **CAT-3: CGI/Animation (Child Exploitive) (0)**
- ⬜ **...mplar/Comparison (Internal Use Only) (0)**
- 🟩 **CAT-5: Non-pertinent (0)**

Category	# Files
CAT-1: Child Exploitation...	0
CAT-2: Child Exploitation...	0
CAT-3: CGI/Animation (C...	0
CAT-4: Exemplar/Compa...	0
CAT-5: Non-pertinent	0
CAT-0: Uncategorized	166

Attribute	Value
Name	test1.pdf
Analyzed	true
Category	CAT-0: Uncategorized
Tags	
Path	/LogicalFileSet2/
Created Time	0000-00-00 00:00:00
Modified Time	0000-00-00 00:00:00
MD5 Hash	a56f716b3c97dbc4 5e32375a56a00a12
Hashset	

Figure 7.8: Screen capture from Autopsy interface

commonly used in digital forensic investigations. It generates raw image files of disks that can be read by other programs. In this case, the image file generated by the dd tool is imported into TSK. TSK runs a standard analysis on the captured image and generates a summary of what was found.

Figure 7.8 depicts the output for an image search of the captured image. It can be seen that the test photographs were indeed found on the image.

For the case where the photographs are copied to the drive unaltered, the photographs were recovered and the MD5 hash values generated. The resulting hash values were:

- test1.jpg (Lena): a56f716b3c97dbc45e32375a56a00a12
- test2.jpg (Nuke): cdf39cb9eb9b1c49121af3fd4d3223a7

These hash values are identical to that of the sample files.

For the next case, only the file names were altered. The names were changed from test1.jpg to IMG_3920.jpg and test2.jpg to IMG_3921.jpg. These images were again found and, as in the aforementioned case, the hash values matched for both files.

In this case, the file extension was altered. The extension was changed from .jpg to .pdf for both files. Again, both files were successfully found and their hash values matched successfully.

In the final case, the file header was changed from that of a JPEG file to a PDF file, as described in the experiment setup. In this case the files were again found. However, the files could not be rendered in the image viewer. The hash values for both files also did not match that of the original files. This was to be expected, as the content of both files were changed. This would invariably change the MD5 hash value for the files.

7.5.3 CPU performance

It is a given fact that the operation of the FRC system will have some impact on the performance of the cloud hosting system. The FRC system, just like any other piece of software, requires machine cycles to operate. However, it is necessary to determine how large an impact the operation of the FRC system has on the cloud hosting system. The pricing for using public cloud systems is often determined on the amount of resources used by the VM if hosted in the cloud. Should the FRC be too resource-intensive, it would not be a viable option due to costs. Thus, the following experiment is set up to determine the impact of the FRC system on the cloud hosting system.

7.5.3.1 Hypothesis

In order for the FRC system to meet its functional requirements regarding stability of captured digital evidence, the following statement must be true.

Captured digital evidence must be verified as a true copy of the data by applying a hashing algorithm to the data. The hashing process must not put undue load on the cloud system in terms of performance.

Hashing is an iterative process and will likely have an adverse effect on performance when applied to large files. As the data within the target file is used and reduced to a single string of unique characters, it stands to reason that large files would require more iterations to complete this process. As a result, larger files should be more resource-intensive than smaller files.

7.5.3.2 Setup

For this experiment, the Sysbench benchmarking utility is used to test the performance of a single node while the cloud hosting system and FRC system are running. The tests are run to determine the performance of the CPU. To get a base line, the node is benchmarked when the installation of the base operating system is completed. No other software is installed on the host, apart from the base operating system and the Sysbench tool. The CPU is also confined to single core operation, as per the setup of the VM template.

The next benchmark is run when the cloud hosting system has been installed and the host is running as a node forming part of the cloud hosting system. For this scenario, a forensic VM is allocated to the node by the CloudStack management console. A guest VM is also running within the guest VM to simulate the normal configuration of the FRC system. While the VMs are running, no other code or software is executed other than the standard operating system processes.

Finally, the system is benchmarked while the FRC system is running. This is done while the FRC system is running the hashing operations in the forensic VM. In order to test the hashing of the FRC system, sample files of 1kB, 100kB, 1MB, 100MB, 1GB and 10GB are created. This file must be hashed and transmitted to the cloud forensic system during the benchmark test. The MD5 hashing algorithm is used for the test. The Sysbench test is run with the following command:

```
\$ sysbench --test=cpu --cpu-max-prime=20000 run
```

The combination of these tests should give a good indication of how much resources the FRC system requires to execute. Each test was repeated ten times and the average of all the runs was taken.

7.5.3.3 Results

Figure 7.9 depicts the number of operations per second of a single node. From this figure it can be seen that for the benchmark test, the running VM and the small files of 1kB and 10kB, there is very little variation in performance. These is a slight drop for a 1MB file, but this could be that some other process was also active. A noticeable drop in performance could be seen with large files of 100MB, 1GB and 10GB. The effect of the 10GB file was severe and seemed to consume all available system resources.

Since hashing algorithms depend on repeating the same operation a number of times, it is consistent with the test results that the larger a file, the more intensive the resource use would be. Another factor is that the Sysbench test usually runs for about 10 seconds until completion. In the case of the large files, hashing operations are started before the test is run. This gives priority to the hashing operation and might further degrade performance.

This test is only done on a single node, so it is likely that should there be more CPU resources, these results would improve. This can be accomplished by running multiple CPU cores, or spanning the VM over multiple nodes.

7.5.4 Encryption performance

This experiment isolates the encryption portion of the process, as described in Chapter 4 section 6.3.1.2. Encryption is usually a performance-intensive process [127]. As stated, all performance overhead in a cloud system comes at a cost, either to the cloud provider or the cloud consumer. For this reason, the encryption and decryption components are likely to be the most resource-intensive components of the FRC system. This experiment aims to benchmark the speed at which the forensic VM and cloud forensic system can encrypt and decrypt files.

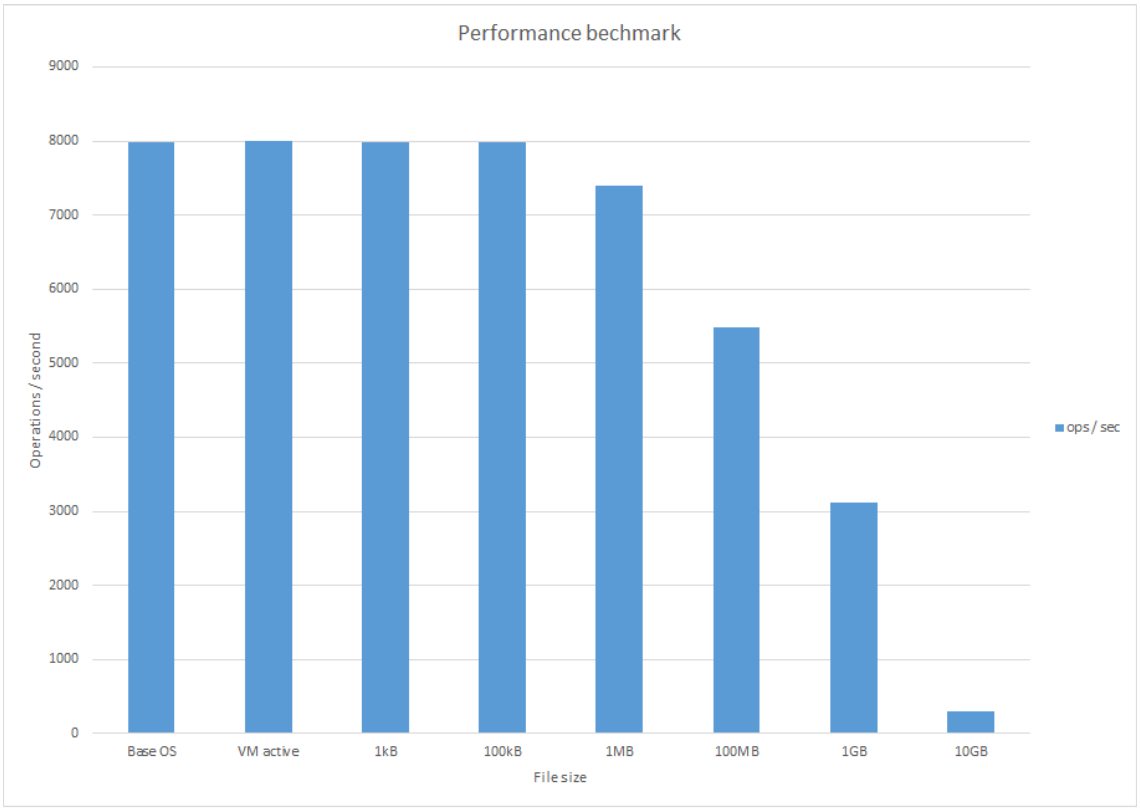


Figure 7.9: Performance while hashing

7.5.4.1 Hypothesis

In order for the FRC system to meet its functional requirements regarding the maintaining of stability of forensic data during transmission, the following statement must be true.

Data that is transmitted from the forensic VM to the cloud forensic system is encrypted in order to ensure that it is not intercepted or changed during transmission. The encryption process must not put undue load on the cloud system.

Since the encryption process is iterative, it stands to reason that the larger the encrypted file, the longer it should take to encrypt and thus would require more resources. It is expected that small files would encrypt quickly and, as the file size increases, so too will the time required to perform the encryption operation.

7.5.4.2 Setup

In order to implement this experiment, the Rijndael Advanced Encryption Standard (AES) algorithm was used as implemented in the Java cryptography libraries [128, 129]. In this implementation, a 128bit cipher and key are used, as it is the least complex version of AES and should thus yield the best performance.

In order to determine the time taken for the algorithm to execute, the system time was taken exactly before execution of the algorithm began and the end time exactly after the execution of the algorithm ended. The times are then subtracted and the execution time is determined. The time is measured in milliseconds (ms). The code snippet below gives the exact instructions, including the console output.

```
double startTime = System.currentTimeMillis();
crypto.run();
double endTime = System.currentTimeMillis();
double duration = endTime - startTime;
System.out.println("Execution time: " + duration + "ms");
```

As with the acquisition of data test described in section 7.5.1, files are created on which the encryption could be tested. These files range from 1kB to 10GB in size. The contents are again the MD5 and SHA256 hash values for the string “Forensic test file”. The contents are replicated until the desired file size is achieved. The file sizes used are 1kB, 100kB, 1MB, 100MB, 1GB and 10GB.

The times for encryption and decryption are tested separately in order to get the best case for each test run. For each file size, the encryption and decryption algorithm is run 10 times and the average time is taken.

7.5.4.3 Results

Figure 7.10 depicts the results of the encryption decryption test. The time axis for this in Figure 7.10 is logarithmic in order to make the results clearly visible.

From Fgure 7.10, it can be seen that for file sizes of 1kB to 1MB, the encryption and decryption times are more or less constant. The average time for both encrypting and decrypting files was 450ms. At 100MB, the average encryption time was

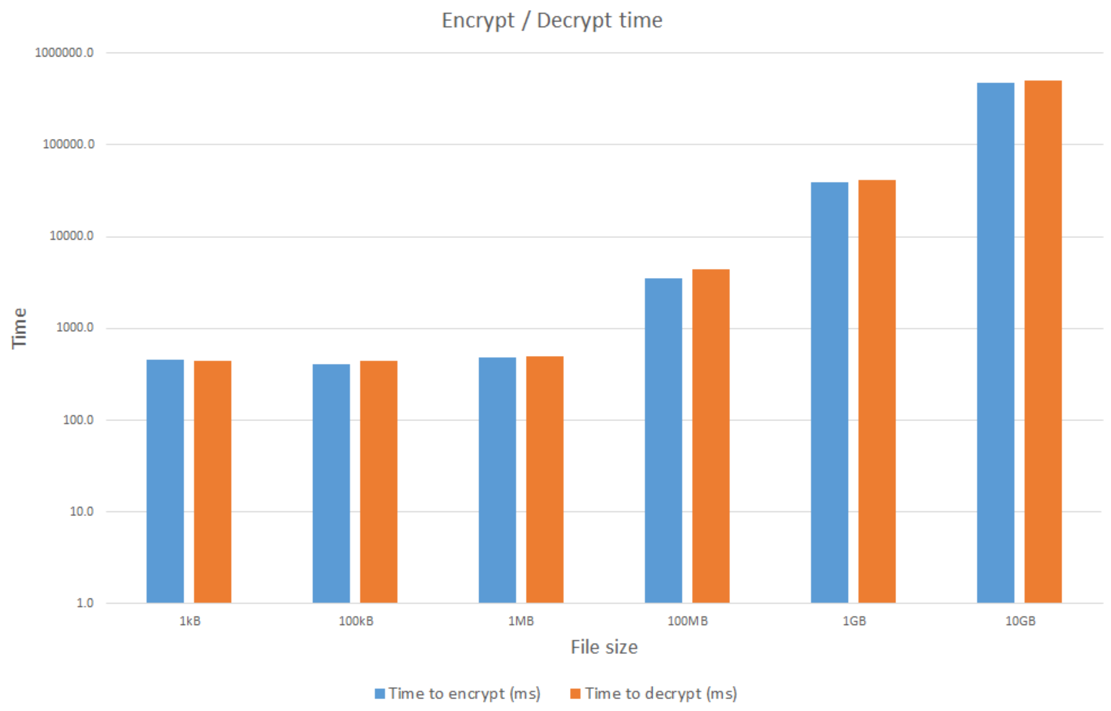


Figure 7.10: Encryption and decryption times

3493.4ms. However, at 1GB file size, the encryption and decryption times increased drastically.

For the 1GB file, the average encryption time was 39.12 seconds (s) and the decryption time was 41.86s. It should also be noted that, with the initial implementation of the encryption component, the heap space for the Java virtual machine ran out when attempting to encrypt or decrypt files. The component had to be rewritten to allow for sequential blocks to be encrypted and decrypted as a stream. This likely added to the time needed to execute the encryption and decryption functions.

For the 10GB file, the average time again rose substantially. The average time for encryption was 7.83 minutes or 7 minutes and 50 seconds. The average decryption time was 8.37 minutes or 8 minutes and 22 seconds.

The implementation of the FRC system allowed experiments to be performed to determine whether the system meets the relevant requirements. These results must be evaluated to determine their validity.

7.6 Conclusion

This chapter details the proof of concept implementation of the FRC system. In this chapter, the hardware and software configuration used to implement the FRC system is given. The implementation of each of the different components of the FRC system is then examined. Finally, experiments are performed using the implemented components. This is done to obtain information regarding the implemented components with regard to functionality and performance of the system.

Having performed the above-mentioned experiments, it is now possible to critically evaluate the FRC system. This is done in the next chapter.

Chapter 8

Critical evaluation

8.1 Introduction

When looking at the implementation of the FRC system, as described in Chapter 7, it is necessary to evaluate said implementation to determine whether it meets the goals set for the FRC system. The goals for the FRC system are set in the software architecture and must be met for the system implementation to be considered successful.

The goals set for the FRC system, as per section 6.2.1.1, are that data must be captured from a cloud VM, for the purposes of digital forensics. The captured data must be kept in a forensically sound state and must be attributed with the relevant meta data. Finally, the data must be transferred to an isolated location for automated analysis. From these goals, the relevant functional requirements and quality attributes are derived and can be evaluated.

This evaluation consists of an evaluation of the implementation of the functional requirements, the adherence to the set quality attributes, an analysis of the experimental results, possible improvements that can be made to the FRC system and a comparison to related work.

8.2 Implementation of functional requirements

The function requirements for the FRC system, as defined in Chapter 6 section 6.2.2, are the following:

1. Data must be captured from a virtual machine in a cloud environment for the purposes of digital forensics.
2. The captured data must be kept in a state that is forensically sound, should the need for a forensic investigation arise.
3. The captured data must be attributed with meta data relevant to the forensic investigation.
4. The captured data must be transferred to an isolated location where an automated preliminary forensic investigation can be conducted.

Examining these requirements, it can be seen that they have been met, as shown by the experiments conducted in sections 7.5.1 and 7.5.2. These experiments show that data can be captured from a VM that is deployed in a cloud environment. In the experiment conducted in section 7.5.1, data was successfully captured from a cloud VM. In the experiment conducted in section 7.5.2, the entire virtual disk was captured.

When examining requirement 2, it can be seen that this requirement has also been met. This is shown firstly in section 7.5.1, where data that is captured is hashed and the hash values stored in both the forensic VM and the cloud forensic system. The payload data is stored in a database that can be retrieved as required. This data integrity can be verified as forensically sound by comparing the hash that was generated when the file was captured by the forensic VM, the hash that was received from the forensic VM, and the hash that was generated when the payload was received by the cloud forensic system. If all these hashes match, it can be assumed that the data is forensically sound. In the case of capturing the entire VM, as shown in section 7.5.2, the image is directly imported into the analysis engine, which keeps the captured data forensically sound in its internal database.

For requirement 3, data is attributed as it is captured. The experiment in section 7.5.1 shows that every piece of captured data is attributed with meta data. This meta data accompanies the captured data to the cloud forensic system, where it is stored for later use. In the case of capturing the full VM, as shown in section 7.5.2, the image is directly imported into the analysis engine, which analyses the data and automatically attributes it.

Finally for requirement 4, the data is transferred to the cloud forensic system where it is securely stored. The cloud forensic system is completely separate from the cloud hosting system. In terms of the preliminary analysis, the analysis engine can perform such analyses, as shown in the experiment described in section 7.5.2.

From this analysis, it can be concluded that the functional requirements defined for the FRC system have indeed been met by the provided implementation.

In addition to the functional requirements, the quality attributes are also important attributes of the FRC implementation.

8.3 Implementation of quality attributes

The quality attributes for the FRC system are defined in section 6.2.1.3. Meeting these attributes are important, as their implementation can have far-reaching effects on the FRC system. These effects can mean that, despite the system meeting its functional requirements and because of a quality attribute not being implemented correctly, the system might still fail to achieve the overall goal for which it was built. An example of and effect like this can be, in the case of auditability, that the audit trail could be broken. This results in the chain of evidence being broken, rendering the captured digital evidence inadmissible in a court of law. Similar problems can be expected with all the defined quality attributes.

The defined quality attributes for the FRC system, as shown in section 6.2.1.3, are:

- Auditability;
- Security;

- Integrability; and
- Affordability.

In terms of auditability, the FRC system caters for this in multiple ways. When using the forensic VM, each hash generated is written to the hash store contained in the forensic VM. In addition to the hash being stored, the transaction log of the hash being written is also stored as an audit trail. All records written to the forensic VM hash store are time-stamped, as are the transaction logs.

From the perspective of the cloud forensic system, the records written in the forensic hash store and payload store are also time-stamped. The time stamps relate to both when the file was captured and when the payload was received. Again, transaction logs are kept for all database actions. The analysis engine also has internal mechanisms for auditability. These internal mechanisms include: enforcing strict read-only access to all captured data, in order to maintain the integrity of the data. Next, to time-stamp all the captured data to ensure the chain of evidence. Finally, to build a tree structure to show from where the data was captured, in order to ensure the provenance of the data.

The security quality attribute is achieved by maintaining the CIA triad. In the case of confidentiality, the FRC implementation allows for each forensic VM to contain a single guest VM. This has the result that no cross-contamination of information can occur within the forensic VM. Data transferred from the forensic VM to the cloud forensic system is encrypted. This ensures that, should data be intercepted, it would be useless without the relevant decryption key.

In the case of integrity, data integrity is ensured by using hashing, as described in section 6.3.1.2.

Finally, availability is ensured by the method of implementation of the FRC system. The FRC system uses the concept of a “bolt-on” system. Installing the FRC system does not require any interruption of the cloud hosting system. Guest VMs that are already running in the cloud hosting system can be migrated to a forensic VM at an opportune time. By maintaining the CIA triad, the security quality attribute is achieved by the FRC implementation.

Security can be greatly influenced by external factors with regard to the physical security of the cloud infrastructure. However, this is outside the scope of this research.

Integrability is achieved by the “bolt-on” construction of the FRC system. The FRC system can be deployed in an already operational cloud environment. Each of its components can be installed as needed. For example, an already existing guest VM can be migrated to a forensic VM via a simple clone operation. The switch-over between the original guest VM and guest VM running in a forensic VM can then be done at an opportune time. The cloud forensic system is not dependent on any of the cloud hosting infrastructure. It must be installed separately in order to maintain security. By removing the need for complex integration, the integrability of the system is improved. Finally, the forensic controller needs only to interface with the cloud security system.

Finally, for the quality attribute of affordability, the entire implementation of the FRC system was done using only free open-source tools and software. The only cost involved was time and effort to design and implement the FRC system. Related to this is that the total cost of ownership can be kept low. This is due to the fact that no third party software licences are required. In addition to this, all hardware used is generic PCs. No custom hardware is needed to run the FRC system.

From this analysis, it can be concluded that the quality attributes defined for the FRC system have been met by the provided implementation.

Although the quality attributes have been met by the FRC system, the experimental results can shed some light on additional factors not covered by the formal requirements. These experimental results are discussed in the next section.

8.4 Experimental results

The experiments conducted in Chapter 7 section 7.5, are designed to test the various functional aspects of the FRC system. These functional experiments included capturing and attributing data, and capturing and analysing a full VM. However, in addition to the functional experiments, additional experiments were also conducted.

These experiments tested the performance of the FRC system in terms of CPU performance and performance of encrypting and decrypting files.

The results of the experiments regarding the functional requirements of the system are discussed in section 8.2 and will therefore not be repeated in this section.

The performance experiments can be found in section 7.5.3 and section 7.5.4 respectively. It should be noted that the network performance was not tested. The reason for this is that the available hardware for the implementation was of the ordinary commercial variety and not that which would be found in a cloud data centre. No useful information would be gained from testing a basic network.

The two most computationally expensive operations of the forensic VM runs, are hashing and encryption. Both hashing and encryption rely on operations that are performed repeatedly on a piece of data. In the case of hashing, this is to reduce the size of the data to a unique string associated with the data, whereas in the case of encryption, it is to scramble the data using a key so that it is unintelligible to anyone who does not have the decryption key. Regardless of the specific operation of hashing and encryption, both are CPU intensive.

In the case of the hashing experiment, it can be seen that for small files (1kB to 1MB), the hashing has little impact on performance. This is consistent with the operation of the hashing operation, since less cycles are needed to reduce the file to a unique string. From the 100MB mark, there is a noticeable decline in the performance. The operations per second of the CPU drop from the benchmark of around 8000 ops/sec to approximately 5400 ops/sec. The performance drops as the files get larger, consistent with the number of operations needed to perform the hash operation. For 10GB sized files, the operations per second went down to approximately 300 ops/sec.

From this it can be concluded that the hashing process would in all likelihood have the most impact on the performance of the forensic VM. The process is, however, a vital component in the forensic process.

In addition to the performance overhead of the hashing operations, encryption is another resource-intensive process. In the encryption and decryption experiment, a number of files of different sizes were encrypted and decrypted. The time taken to complete these operations was measured.

It was found that the time taken for small files, in the size range of 1kB to 1MB, was approximately 450ms to encrypt and decrypt. However, for larger files, this time increased drastically. For a file of size 100MB, the time went up to 3.5 seconds. As the file sizes increased, so too did the time. Figure 7.10 shows that the time growth is exponential for every order of magnitude that the file size increases.

Another issue was that the encryption module on the forensic VM ran out of memory when encrypting files larger than 1GB. This necessitated a rewrite of the encryption component to enable data blocks to be streamed in and out of memory. The streaming added additional overhead to the already resource-intensive process.

It can thus be concluded that the size of the file that is captured, will have a dramatic effect on the performance of the forensic VM. As it is required that the captured file be transmitted while being encrypted, in order to maintain the CIA triad, a more efficient method of encryption must be found. It should be noted that it is not the encryption itself that is the issue, but rather computational overhead created by running the encryption process.

While the premise of the FRC system is that of a "bolt-on" solution, it is of interest how other solutions handle large volumes of data. The most common method of handling large amounts of data is to reduce its size using data compression [130]. Depending how a compression technique is implemented, be it lossy or lossless, would determine if it could be used for a digital forensic investigation. Data to which lossy compression is applied, would immediately be disqualified, as lossy compression changes the data. Lossless compression might be a better option, if the data can remain in a forensically stable state.

The use of big data tools or platforms could also be applicable. Examples of such technologies are IBM Watson [131], SAP Data Hub [132], Teradata [133] and Mongo DB [134]. While these applications are excellent at data analytics, they are not geared toward digital forensics. Also, the cost of these technologies can be extremely high.

Finally, a common solution for dealing with large amounts of data is to put it in the cloud [135]. This, however, brings the problem back full circle, as this dissertation attempts to solve the problem of cloud forensics.

Having analysed the experimental results, it is possible to suggest improvements to the FRC system to address the identified issues.

8.5 Possible improvements

From the analysis of the FRC system, certain improvements can be identified and possible solution proposed. The main issue that emerged regarding the implementation of the FRC system is that of performance, specifically the hashing, encryption and management of large files.

In the case of hashing large files, there are a number of tactics that can be employed to make the process more efficient. The first, and most obvious tactic, would be to simply increase the resources available to the forensic VM. This will naturally have an effect on the affordability of the system as the running cost will increase. Next would be to remove the hashing from the CPU and transfer it to a GPU. GPU pipelines are much more efficient at repetitive mathematical operations, due to the large thread blocks and large number of threads in each thread block. In addition to this, the use of a parallel hashing algorithm can spread the load to multiple threads. An example would be the SHA-3 algorithms.

Closely related to the performance issue encountered while hashing, is the performance issue of encryption and decryption. It is again possible to increase the available resources. As before, this tactic will have a negative effect on the affordability of the system. Another possible option is to transmit the data from the forensic VM to the cloud forensic system using a stream cipher instead of a block cipher. In the implementation tested, the AES-128 algorithm is used, which is a block cipher. This requires that the entire portion of captured data must first be encrypted, only after which it can be sent. Using a stream cipher would eliminate this problem. The data can be consciously encrypted and sent after the hash value has been generated. The hash values can still be encrypted with a block cipher to ensure that they cannot be modified while in transit from the forensic VM to the cloud forensic system.

Another issue with large files is that of storage within the cloud forensic system. The implementation has large files added to a relational database as BLOBs. This is not ideal, as it is very inefficient to store large objects in a relational database. A possible solution would be to use a NoSQL database [136], such as MongoDB [134], to store large blocks of data. An index reference can be kept in the main SQL payload database for quick queries.

Having evaluated the FRC system implementation, it is possible to compare the FRC system to related solutions.

8.6 Comparison with related work

The architectures and models discussed in section 3.9 can be compared to the results gained form the FRC implementation. The related work discussed is that of De Marco et al. [86], Alenezi et al. [87] and, Kebande and Venter [88].

In the case of the reference architecture (CFRS) proposed by De Marco, there are many similarities to the FRC architecture. The goals of both are essentially the same in that both attempt to minimise the cost and time taken to perform a digital forensic investigation. The key difference between the CFRS and FRC systems are the proposed methods of data acquisition. The CFRS system attempts to inject tools into the VMs running on cloud infrastructure to perform forensic functions. The FRC system, in contrast, makes use of nested VMs as previously shown.

While the De Marco proposed that the CFRS system can be configured using XML and the Open Virtualisation Format (OVF) [137], the forensic soundness of each configuration could be questioned and would have to be proven. This could result in the failure of the main requirement of the system, in that time and money would have to be spent to prove each configuration forensically sound. The FRC system has no such problem since raw data is captured, attributed and analysed in a consistent manner. Also, it is stated that additional infrastructure might be required for the implementation of the CFRS architecture in terms of networking hardware. This is not a requirement for an implementation of the FRC architecture as it uses the cloud communications hardware as is. Since a detailed implementation of the CFRS system is not available, it is difficult to do a direct comparison in terms of performance to the FRC system.

The framework proposed by Alenezi relates to the technical, legal and organisational factors of forensic readiness. The FRC system relates mostly to the technological factors in the framework. The technical factors are cloud infrastructure, cloud architecture, forensic technologies and cloud security. In the implementation of the FRC system, the existing cloud infrastructure is utilised to achieve forensic readiness. This is done without impact to the existing cloud architecture due to the "bolt-on" design of the FRC system. The forensic tools take the form of the Cloud forensic system, as discussed in section 6.3.1. Finally, cloud security is handled by the cloud security system also discussed in section 6.3.1.

The legal factor relevant to this dissertation is the factor of a regulatory nature. The regulatory factors relate to aspects of admissibility of digital evidence and chain of custody. The FRC system is specifically setup to maintain the forensic soundness of the captured data and the ability to prove the origin of said data. In addition to this the chain of custody is kept as short as possible to maintain the integrity of the captured data. With these factors taken into account, it can be seen that the FRC system satisfies all the technical factors and some of the legal factors of the proposed framework. The organisational factors are outside the scope of this dissertation.

The model proposed by Kebande and Venter utilises a modified botnet to install an agent on a cloud VM to facilitate the acquisition of digital evidence. The model, named the cloud forensic readiness (CFR) model, has many similarities with the Operating system embedded forensic monitor model proposed in section 5.3.1. The CFR model is however superior in that the agent behaves like a node of a botnet which is continuously installed and deleted. This mitigates the problem of OS embedded model where the forensic monitor can be tampered with or disabled. A major similarity between the CFR model and the FRC system, is that the CFR model also does not require modification of the cloud infrastructure, as this can be costly and is thus undesirable. In addition to this, both the CFR model and the FRC architecture implement proactive data acquisition in order to make the investigative process more streamlined.

A major limitation of the CFR system is that it is limited to the SaaS delivery model. In contrast, the implemented FRC model can work with any of the cloud service delivery models as described in section 2.3.

Having analysed the implementation of the FRC system and implementing the goals set, as well as how it compares to other related models and architectures, it is possible to summarise and conclude this dissertation.

8.7 Conclusion

From the analysis done in this chapter, it can be seen that the FRC system meets the relevant goals. The FRC system meets the functional requirements as it operates as defined. In terms of the set quality attributes, the FRC system meets these attributes by applying the relevant tactics. Experiments identified possible weaknesses in the FRC system and these weaknesses are addressed.

The next, and final chapter, summarises the research done and the publications derived from it. Some avenues of possible future research are mentioned and the dissertation is concluded.

Chapter 9

Conclusion

9.1 Introduction

Cloud forensics remains a very challenging problem in the field of digital forensics. This research is aimed at enabling proactive forensics in a cloud computing environment. Proactive forensics aims to aid a forensic analyst by cutting down on the time and complexity of a forensic investigation. This dissertation proposed models for forensic monitoring and an architecture in which these models could be integrated. The architecture was validated, implemented and tested.

9.2 Dissertation summary

In this dissertation, the problems faced by cloud forensics were examined. From this examination, the following research question was formulated: Is it possible to achieve forensic readiness, in a cloud environment, so that a digital forensic investigation can be conducted with minimal or no disruption to the operation of said cloud environment?

A literature survey was undertaken regarding the fields of study related to the problem. These fields are cloud computing, digital forensics and software architecture. Models for forensic monitoring were created using the NIST cloud reference architecture.

To implement these models, an architecture was created to enable proactive forensics in a cloud system. This architecture was implemented and tests were conducted to determine whether it did in fact enable proactive forensics. It was found that the architecture did indeed enable proactive forensics.

The results of this research were critically analysed and areas for improvements were identified. The dissertation was then concluded.

9.3 Research contribution

This research contributed to the field of cloud forensics by proposing multiple models that would enable forensic monitoring in a cloud system. From there, an architecture was created to enable the forensic monitoring models to be implemented in a cloud environment. One of the models was selected and the architecture was implemented as a proof of concept. After conducting a series of tests, the areas for improvement were identified for further research.

In terms of creating models for data acquisition in a cloud system, the NIST cloud reference architecture was used as a base. The core of the cloud hosting component was then analysed to determine where forensic monitoring could occur. A model was defined for each point in the core hosting component where monitoring could be viable. These models were elaborated upon, compared and contrasted.

The monitoring models form the core component of the larger architecture, that not only captures data from a cloud system, but does it in a manner that ensures the forensic soundness of digital evidence. The architecture also addresses some of the major issues associated with cloud forensics. The addressed issues are the scale of cloud systems, isolation of relevant data, integrity of captured data and the availability of the cloud system. In addition to these issues, the CIA triad is also maintained to ensure the confidentiality of information that is captured within the cloud environment.

The architecture was validated in terms of its functional and quality attributes to ensure that it is relevant to the problem of digital forensics of cloud computing systems. The architecture was implemented as a proof of concept. This was done to show that it will indeed achieve the desired goals set the architectural specification. The goals were met as stated thus indicating that the architecture is fit for purpose. The architecture is also platform- and technology-independent. This results in that the implementation shown in this dissertation could be one of many different interpretations of the FRC architecture.

The next section contains the publications that were derived from this research.

9.4 Derived publications

The following publications were derived from the research done for this dissertation.

- D. J. Ras and H. S. Venter, "Models for the Forensic Monitoring of Cloud Virtual Machines," in *13th European Conference on Cyber Warfare and Security ECCWS 2014*, A. Liaropoulos and G. Tsihrintzis, Eds. Reading: Academic Conferences and Publishing International Limited, 2014, pp. 290–299 [18]
- D. J. Ras and H. S. Venter, "Proactive digital forensics in the cloud using virtual machines," in *2015 International Conference on Computing, Communication and Security (ICCCS)*. IEEE, Dec 2015, pp. 1–6 [138]
- ——, "Architecture for the proactive acquisition and analysis of forensic information in cloud systems," *Suid-Afrikaanse Tydskrif vir Natuurwetenskap en Tegnologie*, vol. 35, no. 1, Feb 2016 [139]

Having shown the research contribution and publications derived from this dissertation, more avenues of potential research were uncovered. This future work is discussed in the next section.

9.5 Future work

Much work is still to be done in the field of cloud forensics. In order to improve the FRC system, the performance issue must be addressed. It is currently unknown how the system will perform in a larger cloud environment and this could be tested. Another approach to the performance issue would be to introduce forensic monitoring on the hardware level. This would increase the performance significantly, as the computational overhead would be removed from the cloud system. The trade-off would be that data segregation might become a significant issue. As it is of critical importance to be certain of the provenance of all digital evidence, a hardware level solution might be very difficult to implement. Data would have to be attributed from its point of origin, through the hardware monitoring solution, to the point where it is stored, in a forensically sound manner. It would also have to be ensured that data is not in anyway altered by the hardware monitoring solution.

More research must also be done in the field of proactive forensics. It remains one of the only methods to cut down on the time when conducting a digital forensic investigation. With this, a scientific comparison should be made to prove that proactive forensics is viable due to the time saving as opposed to the traditional form of postmortem forensics.

Other avenues of research can be found by looking at the areas which this dissertation did not address. As described in Chapter 1 section 1.5, the limitations that are of note are the networking in the implementation of the FRC system, the storage of captured data and the scaling of the FRC system.

When examining the networking setup, it would be of great interest to see how the FRC system performs when using high performance networking hardware, that is usually found in cloud systems. As the networking setup of commercial cloud systems is geared towards large volumes of data, the efficiency of the FRC system should be greatly increased. It would also be of interest to determine how much additional network traffic is created by the FRC system.

Storage would be another area of research of interest. Storage can refer to both to the storage of captured data and the storage of digital evidence. Depending on the implementation and configuration, the storage requirements of the FRC system could be significant. This would be especially significant when capturing entire guest VMs in large numbers, as each captured guest VM would require the same amount of space as the original guest VM. This could lead, in a worst case scenario, to the storage requirements of the cloud having to be doubled.

Finally, the effective scaling of the FRC architecture can be determined. For the FRC architecture to be usable in an industrial cloud deployment, it would have to effectively scale in such an environment. The FRC architecture is designed with this in mind. Tactics, such as decoupling of components and messaging as a means of communication between components, should be adequate to allow the architecture to scale. This assertion should, however, be tested further to determine its validity.

9.6 Final conclusion

Cloud forensics remains, and in all likelihood will remain, a challenging field of research. As technology evolves, so too will crime related to such technology. As a result, law enforcement must keep pace with this evolution. While many strategies can be devised to safeguard systems and information, only time will prove their success.

Bibliography

[1] R. B. Bohn, J. Messina, F. Liu, J. Tong, and J. Mao, "NIST cloud computing reference architecture," in *Proceedings - 2011 IEEE World Congress on Services, SERVICES 2011*, 2011, pp. 594–596.

[2] Citrix, "Cloudstack dashboard." [Online]. Available: https://www.citrix.com/blogs/wp-content/uploads/2013/04/cloudplatform-dashboard.png

[3] R. Rojas, *Encyclopedia of computers and computer history.* Chicago: Fitzroy Dearborn, 2001.

[4] IBM, "The birth of the IBM PC." [Online]. Available: https://www-03.ibm.com/ibm/history/exhibits/pc25/pc25_birth.html

[5] Intel, "Intel Introduces The Pentium® 4 Processor." [Online]. Available: https://www.intel.com/pressroom/archive/releases/2000/dp112000.htm

[6] NIST US Department of Commerce, "NIST Cloud Forensic Science Challenges," NIST, Tech. Rep., 2014.

[7] P. Mell and T. Grance, "The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology (2011)," NIST, Gaithersburg, Tech. Rep., 2012.

[8] Y. Jadeja and K. Modi, "Cloud computing - Concepts, architecture and challenges," in *2012 International Conference on Computing, Electronics and Electrical Technologies, ICCEET 2012*, 2012, pp. 877–880.

[9] K. Ruan, J. Carthy, T. Kechadi, and M. Crosbie, "Cloud Forensics: An overview," in *Advances in digital forensics VII.* Springer, 2011, pp. 35–46.

[10] S. Ballou *et al.*, "Electronic Crime Scene Investigation: A Guide for First Responders," U.S. Department of Justice, Washington, Tech. Rep., 2001.

[11] D. Birk, "Technical Challenges of Forensic Investigations in Cloud Computing Environments," in *Workshop on cryptography and security in clouds*, 2011, pp. 1–6.

[12] ISO 27043, "Incident investigation principles and processes," *Information technology - Security techniques*, vol. 2015, pp. 1–27, 2015.

[13] F. Zhang, J. Chen, H. Chen, and B. Zang, "CloudVisor," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles - SOSP '11*. New York, New York, USA: ACM Press, 2011, p. 203.

[14] B. Hayes, "Cloud computing," *Communications of the ACM*, vol. 51, no. 7, p. 9, Jul 2008.

[15] F. Cohen, *Digital Forensic Evidence Examination*, 3rd ed. Fred Cohen & Associates, 2009.

[16] J. Tan, "Forensic readiness," *Cambridge, MA:@ Stake*, no. October, pp. 1–23, 2001.

[17] S. Alharbi, J. Weber-Jahnke, and I. Traore, "The Proactive and Reactive Digital Forensics Investigation Process: A Systematic Literature Review," *International Journal of Security and Its Applications*, vol. 5, no. 4, pp. 87–100, 2011.

[18] D. J. Ras and H. S. Venter, "Models for the Forensic Monitoring of Cloud Virtual Machines," in *13th European Conference on Cyber Warfare and Security ECCWS 2014*, A. Liaropoulos and G. Tsihrintzis, Eds. Reading: Academic Conferences and Publishing International Limited, 2014, pp. 290–299.

[19] F. Solms and L. Cleophas, "A Systematic Method for Software Architecture Design," *ACM Transactions on Software Engineering and Methodology*, no. August, pp. 1–35, 2014.

[20] N. Antonopoulos, G. Exarchakos, M. Li, and A. Liotta, Eds., *Handbook of Research on P2P and Grid Systems for Service-Oriented Computing.* IGI Global, 2010.

[21] K. Jamsa, *Cloud Computing.* Jones & Bartlett Learning, 2012.

[22] M. Kavis, *Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS).* John Wiley & Sons, 2014.

[23] B. Siddhisena, L. Warusawithana, and M. Mendis, "Next generation multi-tenant virtualization cloud computing platform," in *Advanced Communication Technology (ICACT), 2011 13th International Conference on*, vol. 54, no. 2. MIT Press, 2011, pp. 405–410.

[24] B. Sosinsky, *Cloud Computing Bible.* John Wiley & Sons, 2011.

[25] R. Sheldon, "The Community Cloud," 2014. [Online]. Available: https://www.red-gate.com/simple-talk/cloud/platform-as-a-service/the-community-cloud/

[26] J. Adams, "Private Cloud Adoption Is Alive And Well," 2016. [Online]. Available: https://go.forrester.com/blogs/16-10-18-private_cloud_adoption_is_alive_and_well/

[27] M. L. Badger, T. Grance, R. Patt-Corner, and J. Voas, "Cloud computing synopsis and recommendations," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep., 2012.

[28] V. Kundra, "Federal cloud computing strategy," p. 43, 2011. [Online]. Available: https://obamawhitehouse.archives.gov/sites/default/files/omb/assets/egov{_}docs/federal-cloud-computing-strategy.pdf

[29] US Department of Commerce NIST, "Security and Privacy Controls for Federal Information Systems and Organizations," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep., Apr 2013.

[30] C. Pettey and R. van der Meulen, "Gartner Says Cloud Consumers Need Brokerages to Unlock the Potential of Cloud Services," 2009. [Online]. Available: http://www.gartner.com/newsroom/id/1064712

[31] E. Manoel, C. Carlane, L. Ferreira, S. Hill, D. Leitko, and P. Zutenis, *Linux Clustering with CSM and GPFS.* IBM Redbooks, 2002.

[32] M. Christodorescu, R. Sailer, and D. Schales, "Cloud security is not (just) virtualization security: a short paper," in *Proceedings of the 2009 ACM workshop on Cloud computing security*, New York, 2009, pp. 97–102.

[33] T. Lillard, *Digital Forensics for Network, Internet, and Cloud Computing: A Forensic Evidence Guide for Moving Targets and Data*, 1st ed. Syngress, 2010.

[34] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Elastic management of cluster-based services in the cloud," in *Proceedings of the 1st workshop on Automated control for datacenters and clouds - ACDC '09.* New York, New York, USA: ACM Press, 2009, p. 19.

[35] K. Hess and A. Newman, *Practical Virtualization Solutions: Virtualization from the Trenches.* Prentice Hall, 2009.

[36] D. Barrett, *Virtualization and Forensics A Digital Forensic Investigator's Guide to Virtual Environments.* Syngress, 2010.

[37] H. A. Lagar-Cavilla, J. a. Whitney, R. Bryant, P. Patchin, M. Brudno, E. de Lara, S. M. Rumble, M. Satyanarayanan, and A. Scannell, "SnowFlock," *ACM Transactions on Computer Systems*, vol. 29, no. 1, pp. 1–45, Feb 2011.

[38] P. Dash, "Hypervisor," in *Getting Started with Oracle VM VirtualBox.* Packt Publishing, 2013, ch. 1.

[39] O. Karakok, A. Afrose, H. Fayed, D. Klebanov, and N. Shamsee, "Server Virtualization," in *CCNA Data Center DCICT 640-916 Official Cert Guide.* Cisco Press, 2015, ch. 17.

[40] B. P. Tholeti, "Hypervisors, virtualization, and the cloud," 2011. [Online]. Available: https://www.ibm.com/developerworks/cloud/library/cl-hypervisorcompare/

[41] R. Boddington, *Practical Digital Forensics.* Packt Publishing, 2016.

[42] K. Inman and N. Rudin, *Principles and Practice of Criminalistics: The Profession of Forensic Science.* CRC Press, 2000.

[43] L. Thomson, "A handbook of computer security," *The British Accounting Review*, vol. 20, no. 1, pp. 90–91, Apr 2009.

[44] M. E. Kabay, "A Brief History of Computer Crime: An Introduction for Students," *Security*, pp. 1–51, 2008. [Online]. Available: http://www.mekabay.com/overviews/history.pdf

[45] IACIS, "IACIS History," 2017. [Online]. Available: https://www.iacis.com/about-2/history/

[46] M. Pollitt, "A history of digital forensics," in *Advances in Digital Forensics VI: Sixth IFIP WG 11.9 International Conference on Digital Forensics, Hong Kong, China, January 4-6, 2010, Revised Selected Papers*, K.-P. Chow and S. Shenoi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 3–15.

[47] C. M. Whitcomb, "An historical perspective of digital evidence: A forensic scientists view," *International Journal of Digital Evidence*, vol. 1, no. 1, pp. 7–15, 2002.

[48] K. Hafner and J. Markoff, *Cyberpunk: outlaws and hackers on the computer frontier, revised.* Simon and Schuster, 1995.

[49] FBI, "Innocent Images," 2006. [Online]. Available: https://archives.fbi.gov/archives/news/stories/2006/february/innocent_images022406

[50] R. Downing, "G-8 initiatives in high tech crime," in *Asia-Pacific Conference on Cybercrime and Information Security*, 2002.

[51] International Organization on Computer Evidence, "G8 Proposed Principles for the Procedures Relating to Digital Evidence," in *IOCE 2000*, Ottawa, 2000.

[52] M. G. Noblett, "Report of the federal bureau of investigation on development of forensic tools and examinations for data recovery from computer evidence," in *Proceedings of the 11th INTERPOL Forensic Sciense Symposium*, 1995.

[53] Scientific Working Group on Digital Evidence, "Digital evidence: Standards and principles," *Forensic Science Communications*, vol. 2, no. 2, 2000.

[54] American Society of Crime Laboratory Directors Laboratory Accreditation Board, "ASCLD Policy Library," Garner, 2017. [Online]. Available: http://www.ascld.org/resource-library/ascld-policy-library/

[55] DFRWS, "The Digital Forensic Research Workshop," 2017. [Online]. Available: https://www.dfrws.org/

[56] IFIP 11.9, "The International Federation for Information Processing (IFIP) Working Group 11.9 on Digital Forensics," 2017. [Online]. Available: http://www.cis.utulsa.edu/ifip119/

[57] SADFE, "International Conference on Systematic Approaches to Digital Forensics Engineering," 2016. [Online]. Available: http://sadfe.org/

[58] B. Carrier, "Defining digital forensic examination and analysis tools using abstraction layers," *International Journal of digital evidence*, vol. 1, no. 4, pp. 1–12, 2003.

[59] M. Reith, C. Carr, and G. Gunsch, "An examination of digital forensic models," *International Journal of Digital Evidence*, vol. 1, no. 3, pp. 1–12, 2002.

[60] G. Palmer, "A road map for digital forensic research: Report from the first digital forensic research workshop (dfrws)," *Utica, New York*, 2001.

[61] E. Casey, *Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet*, 2nd ed. Academic Press, 2004.

[62] E. Casey, Ed., *Handbook of Digital Forensics and Investigations*, 1st ed. Academic Press, 2010.

[63] R. McKemmish, "When is digital evidence forensically sound?" *IFIP International Federation for Information Processing*, vol. 285, pp. 3–15, 2008.

[64] J. Sammons, *The Basics of Digital Forensics: The Primer for Getting Started in Digital Forensics*, 1st ed. Syngress Publishing, 2012.

[65] G. G. Richard and V. Roussev, "Next-generation digital forensics," *Communications of the ACM*, vol. 49, no. 2, p. 76, Feb 2006.

[66] F. Adelstein, "Diagnosing your system without killing it first," *Communications of the ACM*, vol. 49, no. 2, pp. 63–66, Feb 2006.

[67] A. Orebaugh, "Proactive Forensics," *Journal of Digital Forensic Practice*, vol. 1, no. 1, pp. 37–41, Mar 2006.

[68] G. Pangalos, C. Ilioudis, and I. Pagkalos, "The Importance of Corporate Forensic Readiness in the Information Security Framework," *2010 19th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, pp. 12–16, 2010.

[69] C. Taylor, B. Endicott-Popovsky, and D. A. Frincke, "Specifying digital forensics: A forensics policy approach," *Digital Investigation*, vol. 4, no. SUPPL., pp. 101–104, 2007.

[70] R. Rowlingson, "A ten step process for forensic readiness," *International Journal of Digital Evidence*, vol. 2, no. 3, pp. 1–28, 2004.

[71] A. Yasinsac and Y. Manzano, "Policies to Enhance Computer and Network Forensics," *Proceedings of the 2001 IEEE*, pp. 5–6, 2001.

[72] J. Wolfe-Wilson and H. Wolfe, "Management strategies for implementing forensic security measures," *Information Security Technical Report*, vol. 8,

no. 2, pp. 55–64, Jun 2003. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1363412703002073

[73] W. Jansen and R. P. Ayers, "Guidelines on PDA forensics," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep., 2004.

[74] M. G. Noblett, F. Church, M. M. Pollitt, and L. A. Presley, "Recovering and Examining Computer Forensic Evidence," *Forensic Science Communications*, vol. 2, no. 4, p. 8, 2000.

[75] M. A. Caloyannides, N. Memon, and W. Venema, "Digital Forensics," *IEEE Security & Privacy Magazine*, vol. 7, no. 2, pp. 16–17, Mar 2009.

[76] M. S. Olivier, "On metadata context in Database Forensics," *Digital Investigation*, vol. 5, no. 3-4, pp. 115–123, Mar 2009.

[77] W. Kruse and J. Heiser, *Computer Forensics: Incident Response Essentials*, 1st ed. Addison Wesley, 2002.

[78] M. a. M. Guimaraes, R. Austin, and H. Said, "Database forensics," *2010 Information Security Curriculum Development Conference on - InfoSecCD '10*, p. 62, 2010.

[79] S. Datt, *Learning Network Forensics.* Packt Publishing, 2016.

[80] S. Fiorillo, "Theory and practice of flash memory mobile forensics," *7 th Australian Digital Forensics Conference*, 2009.

[81] H. Mahalik, R. Tamma, and S. Bommisetty, *Practical Mobile Forensics*, 2nd ed. Packt Publishing, 2016.

[82] O. Afonin, *Mobile Forensics Advanced Investigative Strategies.* Packt Publishing, 2016.

[83] M. Taylor, J. Haggerty, D. Gresty, and R. Hegarty, "Digital evidence in cloud computing systems," *Computer Law & Security Review*, vol. 26, no. 3, pp. 304–308, 2010.

[84] H. Jahankhani and A. Hosseinian-Far, "Challenges of cloud forensics," in *Enterprise Security*. Springer, 2017, pp. 1–18.

[85] K. Dahbur and B. Mohammad, "The anti-forensics challenge," in *Proceedings of the 2011 International Conference on Intelligent Semantic Web-Services and Applications - ISWSA '11*, 2011, pp. 1–7.

[86] L. De Marco, F. Ferrucci, and T. Kechadi, "Reference Architecture for a Cloud Forensic Readiness System," pp. 1–9, 2014.

[87] A. Alenezi, R. K. Hussein, R. J. Walters, and G. B. Wills, "A Framework for Cloud Forensic Readiness in Organizations," in *2017 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, no. April. IEEE, apr 2017, pp. 199–204.

[88] V. R. Kebande and H. Venter, "On digital forensic readiness in the cloud using a distributed agent-based solution: issues and challenges," *Australian Journal of Forensic Sciences*, vol. 50, no. 2, pp. 209–238, mar 2018.

[89] C. Grobler, C. Louwrens, and S. von Solms, "A Framework to Guide the Implementation of Proactive Digital Forensics in Organisations," in *2010 International Conference on Availability, Reliability and Security*. IEEE, feb 2010, pp. 677–682.

[90] M. Elyas, S. B. Maynard, A. Ahmad, and A. Lonie, "Towards A Systemic Framework for Digital Forensic Readiness," *Journal of Computer Information Systems*, vol. 54, no. 3, pp. 97–105, mar 2014.

[91] M. Elyas, A. Ahmad, S. B. Maynard, and A. Lonie, "Digital forensic readiness: Expert perspectives on a theoretical framework," *Computers & Security*, vol. 52, pp. 70–89, jul 2015.

[92] A. Valjarevic and H. Venter, "Implementation guidelines for a harmonised digital forensic investigation readiness process model," in *2013 Information Security for South Africa*. IEEE, aug 2013, pp. 1–9.

[93] P. M. Trenwith and H. S. Venter, "Digital forensic readiness in the cloud," in *Information Security for South Africa*, 2013, pp. 1–5.

[94] G. Sibiya, T. Fogwill, H. S. Venter, and S. Ngobeni, "Digital forensic readiness in a cloud environment," in *2013 Africon*. IEEE, sep 2013, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/document/6757831/

[95] C. Perrin, "The CIA Triad," 2008. [Online]. Available: http://www.techrepublic.com/blog/it-security/the-cia-triad/

[96] C. Easttom, *Computer Security Fundamentals*, 3rd ed. Pearson Certification, 2016.

[97] C. P. Pfleeger and S. L. Pfleeger, *Security in Computing*, 4th ed. Boston: Prentice Hall, 2009.

[98] F. Solms and D. Loubser, "URDAD as a semi-formal approach to analysis and design," *Innovations in Systems and Software Engineering*, vol. 6, no. 1, pp. 155–162, 2010.

[99] ISO/IEC/IEEE:42010, "Systems and software engineering – Architecture description," *ISO:42010*, vol. 2011, 2011.

[100] M. W. Maier, D. Emery, and R. Hilliard, "Software architecture: Introducing IEEE standard 1471," *Computer*, vol. 34, no. 4, pp. 107–109, 2001.

[101] F. Solms, "What is software architecture?" *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference on - SAICSIT '12*, pp. 363–373, 2012.

[102] R. N. Taylor, N. Medvidovic, and E. Dashofy, *Software Architecture: Foundations, Theory, and Practice*, 1st ed. Wiley, 2009.

[103] I. Gorton, "Essential software architecture," *Essential Software Architecture*, pp. 1–283, 2006.

[104] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Addison-Wesley Professional, 2012.

[105] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, and J. Stafford, *Documenting Software Architectures: Views and Beyond*, 2nd ed. Addison-Wesley Professional, 2010.

[106] G. Senthilvel, O. M. A. Khan, and H. A. Qureshi, *Enterprise Application Architecture with .NET Core.* Packt Publishing, 2017.

[107] G. Bochmann and C. Sunshine, "Formal Methods in Communication Protocol Design," *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 624–631, Apr 1980.

[108] G. Hohpe and B. Woolf, *Enterprise Integration Patterns.* Boston: Pearson Education, 2011.

[109] P. Kruchten, R. Capilla, and J. C. Dueñas, "The role of a decisions view in software architecture practice," *IEEE Software*, vol. 26, no. 2, pp. 36–42, 2009.

[110] P. Kruchten, "Architectural Blueprints The 4 + 1 View Model of Software Architecture," *IEEE software*, vol. 12, no. November, pp. 42–50, 1995.

[111] M. Fowler and K. Scott, *UML Distilled Second Edition A Brief Guide to the Standard Object Modeling Language*, 2nd ed. Addison Wesley, 1999.

[112] ISO 19501:2005, "Unified Modeling Language (UML)," *Information technology - Open Distributed Processing*, vol. 2005, p. 432, 2005.

[113] J.-F. Monin, *Understanding Formal Methods.* London: Springer London, 2003.

[114] K. Beck *et al.*, "Manifesto for Agile Software Development," 2001. [Online]. Available: http://agilemanifesto.org/

[115] R. C. Martin, *Agile Software Development, Principles, Patterns, and Practices*, 1st ed. Pearson, 2002.

[116] R. France, S. Ghosh, T. Dinh-Trong, and A. Solberg, "Model-Driven Development Using UML 2.0: Promises and Pitfalls," *Computer*, vol. 39, no. 2, pp. 59–66, Feb 2006.

[117] W. Richter, C. Isci, B. Gilbert, J. Harkes, V. Bala, and M. Satyanarayanan, "Agentless Cloud-Wide Streaming of Guest File System Updates," *2014 IEEE International Conference on Cloud Engineering*, pp. 7–16, 2014.

[118] A. Mazrekaj and I. Shabani, "Pricing Schemes in Cloud Computing : An Overview," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 7, no. 2, pp. 80–86, 2016.

[119] Apache Software Foundation, "Apache Cloudstack," 2017. [Online]. Available: https://cloudstack.apache.org

[120] ——, "CloudStack Installation Documentation," 2017. [Online]. Available: http://docs.cloudstack.apache.org/projects/cloudstack-installation/en/4.9/

[121] KVM, "Main page — kvm,," 2016, [Online; accessed 1-November-2017]. [Online]. Available: https://www.linux-kvm.org/index.php?title=Main_Page&oldid=173792

[122] Oracle, "MySQL," 2017. [Online]. Available: https://www.mysql.com/

[123] ——, "Java EE 8," 2017. [Online]. Available: http://www.oracle.com/technetwork/java/javaee/overview/index.html

[124] B. Carrier, "The Sleuth Kit," 2017. [Online]. Available: http://www.sleuthkit.org/sleuthkit/

[125] M. Zyczkowski, M. Szustakowski, W. Ciurapiński, P. Markowski, M. Karol, and M. Kowalski, "Optical fiber sensors as the primary element in the protection of critical infrastructure especially in optoelectronic transmission lines," *WIT Transactions on the Built Environment*, vol. 134, pp. 273–283, 2013.

[126] The Open Group, "dd," 2008. [Online]. Available: http://pubs.opengroup.org/onlinepubs/9699919799/utilities/dd.html

[127] A. Nadeem and M. Javed, "A Performance Comparison of Data Encryption Algorithms," *2005 International Conference on Information and Communication Technologies*, no. April 2015, pp. 84–89, 2005.

[128] S. Heron, "Advanced Encryption Standard (AES)," *Network Security*, vol. 2001, no. 12, pp. 8–12, Dec 2001.

[129] Oracle, "Java Cryptography Architecture (JCA) Reference Guide," 2017. [Online]. Available: https://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html

[130] D. Salomon, *A Concise Introduction to Data Compression.* London: Springer London, 2008.

[131] IBM, "IBM Watson," 2018. [Online]. Available: https://www.ibm.com/watson/

[132] SAP, "SAP Data Hub," 2018. [Online]. Available: https://www.sap.com/products/data-hub.html

[133] Teradata, "Teradata," 2018. [Online]. Available: https://www.teradata.com/Products/Software/Database

[134] D. Merriman, E. Horowitz, and K. Ryan, "MongoDB," 2017. [Online]. Available: https://www.mongodb.com

[135] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F. C. M. Lau, "Moving big data to the cloud," in *2013 Proceedings IEEE INFOCOM.* IEEE, apr 2013, pp. 405–409.

[136] S. Edlich, "NoSQL," 2017. [Online]. Available: http://nosql-database.org/

[137] OVF, "Open Virtualization Format." [Online]. Available: https://www.dmtf.org/standards/ovf

[138] D. J. Ras and H. S. Venter, "Proactive digital forensics in the cloud using virtual machines," in *2015 International Conference on Computing, Communication and Security (ICCCS).* IEEE, Dec 2015, pp. 1–6.

[139] ——, “Architecture for the proactive acquisition and analysis of forensic information in cloud systems,” *Suid-Afrikaanse Tydskrif vir Natuurwetenskap en Tegnologie*, vol. 35, no. 1, Feb 2016.

Appendix A

Abbreviations

ABS	Agent Based Solution
ADD	Attribute driven design
ADL	Architectural description language
AES	Advanced encryption standard
ASCLD-LAB	The American Society of Crime Laboratory Directors – Laboratory Accreditation Board
BLOB	Binary large object
BPEE	Business process execution engine
CF	CompactFlash
CFR	Cloud Forensic Readiness
CFRS	Cloud Forensic Readiness System
CFS	Cloud forensics system
CIA	Confidentiality, Integrity, Availability
CPU	Central processing unit
DBMS	Database management system
DFRWS	The Digital Forensic Research Workshop
DHCP	Dynamic host configuration protocol
DMZ	Demilitarised zone
ERD	Entity relationship diagram
ESB	Enterprise service bus
FBI	Federal Bureau of Investigation

FRC	Forensic ready cloud
GB	Gigabyte
GHz	Gigahertz
GPU	Graphical processing unit
HDD	Hard disk drive
IaaS	Infrastructure-as-a-Service
IACIS	International Association of Computer Investigative Specialists
IDE	Integrated development environment
IDS	Intrusion detection system
IO	Input and output
IOCE	International Organization on Computer Evidence
JMS	Java message service
JRE	Java run-time environment
JVM	Java virtual machine
kB	Kilobyte
LAN	Local area network
MB	Megabyte
MDD	Model driven design
MHz	Megahertz
NFC	Near-field communication
NFS	Network file system
NIST	National Institute of Standards and Technology
OS	Operating system
PaaS	Platform-as-a-Service
PC	Personal computer
PoC	Proof of Concept
RAM	Random access memory
SaaS	Software-as-a-Service
SADFE	International Conference on Systematic Approaches to Digital Forensics Engineering

SD card	Secure Digital card
SDK	Software development kit
SIM	Subscriber
SLA	Service level agreement
SOA	Service orientated architecture
SQL	Structured query language
SVN	Apache subversion
SWGDE	Subcommittee and Scientific Working Group on Digital Evidence
TSK	The Sleuth Kit
UML	Unified modelling language
URDAD	Use-Case, Responsibility Driven Analysis and Design
VM	Virtual machine