

Client-Side Encryption and Key Management: Enforcing Data Confidentiality in the Cloud

A dissertation submitted to the University of KwaZulu-Natal

College of Agriculture, Engineering and Science

School of Mathematics, Statistics

And

Computer Science (MSCS)

In the Discipline of

Computer Science

Towards the fulfilment of the Degree

Master of Science

In

Computer Science

University of KwaZulu-Natal

Durban

South Africa

Compiled and Submitted by Napo Nathnael Mosola (216075642)

Under the supervision of Professor J.M Blackledge

And co-supervision of Mr. M.T Dlamini

Acknowledgements

I express my sincere gratitude to my Supervisors Professor Jonathan Blackledge and Mr. Moses Dlamini. It is with their constant support, mentorship, guidance and encouragement that I am able to complete my research under their supervision. I would like to thank Mr. Moses for encouraging me to step up into this research area. During our daily meetings, I learned a lot from him than I could have grasped in a lecture! He helped me gain confidence in my research and gave me fruitful ideas to enable me to produce this dissertation. His critical reviews and support have really made my journey in this research an interesting one, though at times a demanding one. Thank you for making sure this dissertation is of the best quality. I am forever thankful to you, for being an inspirational figure and for taking good care of my academic and personal life. I would also like to thank Prof. Blackledge for his valuable ideas, comments and suggestions.

I would also like to thank the entire IT technical team at the UKZN, Westville campus. Their harmony in working together with me was second to none. Their patience and useful technical insights always benefited me. Their dedication in helping me with technical issues is highly appreciated. Mr. Soren Greenwood, special thanks go to you! Always ready to “protect and serve” the students. Thank you for your help everyone. It was indeed a wonderful experience.

I would like to thank my friend, who is very dear to me, Seeiso Jonas Koali. You have witnessed all my efforts and gave me the will to further my studies. You helped me with my applications and always had those words of encouragement. Thank you bro!



Dedication

To my parents, Daniel Mosola and Elizabeth Mosola, who through their hard work, dedication, determination and sacrifices have done all in their power to provide my sister and I the best support ever. You have given me the courage to aim high. I will forever love you!

Academic integrity declaration

I, Napo Nathnael Mosola, student number: 216075642 declare that:

1. The research presented in this dissertation is my own original work.
2. Where material written by other authors has been used, either from printed or digital sources, this has been duly acknowledged and referenced.
3. This dissertation has not been submitted in part or whole, for any degree or examination at any other institution.
4. I have not allowed any other person to copy my work with the intention of passing it as his/her own work.

Signature:		Date: YYYY/M/D	2016/12/05
Witness (Full name in print): SPREN GARY GREENWOOD			
Signature:		Date: YYYY/M/D	2016/12/05

As the candidate's supervisors, we have reviewed and approved this dissertation for submission.

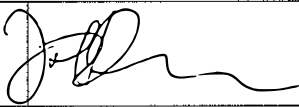

Main Supervisor (Full name in print): Professor Jonathan Blackledge			
Signature:		Date: YYYY/M/D	2016/12/
Co-Supervisor (Full name in print): MOSES T. BLAMINI			
Signature:		Date: YYYY/M/D	2016/12/05

Table of Contents

Table of figures	vii
List of tables.....	viii
Abstract.....	1
Chapter 1.....	3
1 Introduction.....	3
1.1 Overview of the cloud computing paradigm	3
1.2 Problem Statement.....	5
1.3 Research questions.....	8
1.3.1.1 Main research question.....	8
1.3.1.2 Subsidiary research questions.....	8
1.4 Research objectives.....	9
1.5 Research outputs.....	9
1.6 Dissertation scope.....	10
1.7 Dissertation outline.....	10
1.8 Conclusion.....	12
Chapter 2.....	14
2.1 Introduction.....	14
2.2 Cloud computing definition and Background.....	14
2.2.1 Cloud computing definition.....	14
2.2.2 Virtualization.....	15
2.2.3 Cloud computing architecture.....	15
2.2.3.1 Cloud computing services.....	16
2.2.3.1.1 Infrastructure as a Service.....	16
2.2.3.1.2 Platform as a Service.....	17
2.2.3.1.3 Software as a service.....	17
2.2.3 Cloud computing deployment models.....	18
2.2.3.1 Public clouds.....	18

2.2.3.2	Private clouds	20
2.2.3.3	Community clouds.....	21
2.2.3.4	Hybrid clouds	22
2.2.3.5	Cloud computing model essential characteristics.....	23
2.3	Cloud computing security challenges	24
2.3.1	Encryption and key Management	25
2.3.2	Data leakages	26
2.3.3	Partial deletions.....	27
2.3.4	Data co-residency.....	27
2.3.5	Loss of physical control	28
2.3.6	Identification and authentication.....	28
2.3.7	Insider threats	28
2.4	Cryptographic schemes.....	29
2.4.1	Symmetric key cryptography	29
2.4.2	Asymmetric key cryptography.....	31
2.5	Conclusion	33
Chapter 3 – Literature Review		36
3.1	Introduction.....	36
3.2	Related work	36
3.2.1	Encryption.....	36
3.2.2	Key management	41
3.2.2.1.	Key management solutions	42
3.2.3	Secret sharing.....	44
3.2.3.1.	Secret sharing methods	44
3.3	Conclusion	45
Chapter 4 – Proposed Model.....		48
4.1	Introduction.....	48
4.1.1	System functional requirements.....	48
4.2	Proposed model - CloudCrypt	49

4.2.1 Key generation	50
4.2.2. Encryption	50
4.2.3 The Counter Propagation Neural Network (CPNN)	52
4.2.4 Key learning	54
4.2.4.1 Unsupervised learning algorithm	54
4.2.4.2 Supervised learning	55
4.2.5 Decryption	56
4.3. Data sharing	57
4.3.1 Secret sharing scheme	58
4.3.2 Key share self-destruction	60
4.4 Conclusion	60
Chapter 5	63
5.1. Introduction	63
5.2. Implementation	63
5.3. Experimental results	66
5.3.1. Key generation	67
5.1.1. Encryption	70
5.1.2. Key learning	74
5.1.3. Secret-Sharing	75
5.2. Discussion of results	77
5.3. Conclusion	78
Chapter 6 – Conclusion and future work	81
6.1. Introduction	81
6.2. Conclusions	81
6.3. Future work	83
References	84

Table of figures

Figure 1.1: Monthly cyber-attacks [26]	7
--	---

Figure 1.2: Motivation behind cyber-attacks [26]	7
Figure 2.1: Cloud computing model architecture [29].....	15
Figure 2.2: Companies offering private clouds [33]	19
Figure 2.3: On-premises private cloud model [48]	20
Figure 2.4: Community cloud deployments [48]	21
Figure 2.5: Companies offering hybrid cloud deployment [52]	22
Figure 2.6: Symmetric key cryptosystem model [60].....	30
Figure 2.7: Asymmetric key cryptosystem model [60]	32
Figure 4.1: CloudCrypt – An Innovative Crypto Scheme [2].....	49
Figure 4.2: Encryption algorithm.....	51
Figure 4.3: Structure of a CPNN.....	52
Figure 4.4: Unsupervised Learning.....	55
Figure 4.5: Decryption algorithm	57
Figure 4.6: CloudCrypt's secret sharing scheme.....	59
Figure 5.1: System's specifications	63
Figure 5.2: OpenNebula Sunstone Server login	64
Figure 5.3: Data centers	65
Figure 5.4: Host 1 monitoring in openNebula	65
Figure 5.5: VMs on the UKZN cluster	66
Figure 5.6: Generation of fitness functions.....	66
Figure 5.7: Random noise	67
Figure 5.8: Generation of floating point numbers	68
Figure 5.9: Key generation	69
Figure 5.10: Encryption and decryption times.....	71
Figure 5.11: CPU usage	72
Figure 5.12: Memory consumption.....	73
Figure 5.13: Training the neural network for key learning.....	74
Figure 5.14: File size vs key shares	76
Figure 5.15: Key length vs number of key shares	76

List of tables

Table 5.1: Comparison of cryptosystems.....	70
Table 5.2: Number of shares vs File size.....	75

Abstract

Cloud computing brings flexible, scalable and cost effective services. This is a computing paradigm whose services are driven by the concept of virtualization and multi-tenancy. These concepts bring various attractive benefits to the cloud. Among the benefits is reduction in capital costs, pay-per-use model, enormous storage capacity etc. However, there are overwhelming concerns over data confidentiality on the cloud. These concerns arise from various attacks that are directed towards compromising data confidentiality in virtual machines (VMs). The attacks may include inter-VM and VM sprawls. Moreover, weaknesses or lack of data encryption make such attacks to thrive. Hence, this dissertation presents a novel client-side cryptosystem derived from evolutionary computing concepts. The proposed solution makes use of chaotic random noise to generate a fitness function. The fitness function is used to generate strong symmetric keys. The strength of the encryption key is derived from the chaotic and randomness properties of the input noise. Such properties increase the strength of the key without necessarily increasing its length. However, having the strongest key does not guarantee confidentiality if the key management system is flawed. For example, encryption has little value if key management processes are not vigorously enforced. Hence, one of the challenges of cloud-based encryption is key management. Therefore, this dissertation also makes an attempt to address the prevalent key management problem. It uses a counter propagation neural network (CPNN) to perform key provision and revocation.

Neural networks are used to design ciphers. Using both supervised and unsupervised machine learning processes, the solution incorporates a CPNN to learn a crypto key. Using this technique there is no need for users to store or retain a key which could be compromised. Furthermore, in a multi-tenant and distributed environment such as the cloud, data can be shared among multiple cloud users or even systems. Based on Shamir's secret sharing algorithm, this research proposes a secret sharing scheme to ensure a seamless and convenient sharing environment. The proposed solution is implemented on a live openNebula cloud infrastructure to demonstrate and illustrate its practicability.

Keywords — Cloud computing, Security, Encryption, inter-VM attacks, Virtualization, VM sprawl, Neural Network, Shamir Secret Sharing

Chapter 1

1 Introduction

This chapter introduces the dissertation and gives some background overview on cloud computing. It goes further to discuss the dissertation's fundamental aspects i.e. the research problem statement, ultimate research questions, objectives, methodology, contributions, scope and structure thereof. The next section highlights an overview of cloud computing from a security perspective.

1.1 Overview of the cloud computing paradigm

Technological trends have made cloud computing popular. Cloud computing has become the hub of most services offered via the Internet. For example, software as a service (SaaS), infrastructure as a service (IaaS), platform as a service (PaaS) are all services offered by cloud computing. Each of these cloud services is offered as a utility using a pay-per-use model. This model is used by major companies such as Amazon, Google, and Microsoft etc., to provision their cloud services. The emergence of the cloud computing paradigm is gaining an expanding interest. More technologies are converging towards provisioning cloud-based services. Cloud computing is also a technology enabler. It provides cost effective architectures that support easy integration of multiple and heterogeneous technologies. For example, the Internet of Things (IoT). This computing paradigm has a huge impact in today's use of the Internet.

As new technological trends are introduced, there is an explosive growth of digital content that is moving to the cloud. This content includes files, photos, videos etc. The emergence and use of various digital devices such as smart phones, tablets, etc., is responsible for generating large amounts of this digital content. A study by [1] predicts a data growth in the digital universe by a factor of 300 by the year 2020. All this astronomical amounts of digital data will be stored in the cloud. This proliferation calls for efficient, effective, and secure cloud services. With the emergence of the IoT, cloud-bound data is increasing at an exponential rate daily. Today, data floods the cloud in Exabytes [2]. This widespread use of cloud storage services emanates mostly from organizations and governmental agencies as they seek more robust and resilient cloud services [3][4]. The increase in the usage of cloud computing services is due to its benefits, namely: flexibility, reduced costs, elasticity, scalability etc., [5].

Cloud computing is founded on a virtual environment. Thus, it is heavily reliant on the concept of virtualization to provision VMs. Virtualization enables abstract infrastructure and resources to be available to clients as isolated VMs [11]. The security threats described herein apply to virtualized environments inherent in the cloud computing space. Virtualization enables multi-tenancy. Multi-tenancy is another unique feature of cloud computing. Essentially, it allows cloud service providers (CSPs) to efficiently manage resource utilization through partitioning of virtualized infrastructure that is shared among different customers. This concept makes it possible to have VMs from directly conflicting tenants co-residing on the same physical infrastructure. This opens up gaping vulnerabilities which might be exploited through inter-VM and VM sprawl attacks [2][9][10]. This could easily result in data leakage threats. Such vulnerabilities might also allow attackers to compromise the confidentiality of cloud-hosted data.

Furthermore, inter-VM and VM sprawl attacks can be launched by a malicious guest VM that targets co-residing VMs on the shared infrastructure. These attacks usually “*hijack*” the hypervisor. Hence, they are referred to as “*hyper-jacking*” attacks [9]. Once the hypervisor has been hijacked, it is easy for the attack surface to increase and escalate to other VMs co-residing on the same infrastructure [58]. That is if virtualization-aware security is not implemented [11]. Virtualization-aware security helps in implementing safe-guards against attacks launched on virtualized resources. An example of such attacks could be when an attacker compromises one guest VM. Such attacks may have cascading effects on other co-residing VMs.

The attacks mentioned above might go unnoticed by cloud users as they occur in a virtualized environment. Thus, co-location of VMs brings vulnerabilities and increases the risks of VM-to-VM attacks due to multi-tenancy [12][13]. In general, attacks emanating from multi-tenancy are due to workloads of multiple tenants residing concurrently on the same physical machine. Practically, client's data may be co-residing with workloads belonging to direct competitors or adversaries, separated only by logical means through access control mechanisms. A flaw in the access control method used could compromise the confidentiality of the hosted data [14].

The benefits of cloud computing maybe tempting, however, not a lot of customers are rushing to transfer their data into the cloud. This can be attributed to security concerns. Most of the security concerns are centred on deliberate or accidental disclosure of cloud-hosted data. These concerns revolve mostly around

confidential data leaks. Currently, this security concern can be addressed by policies and legislative measures binding data owners and service providers to have service level agreements (SLAs) [6]. However, this is not sufficient. There is a need to complement contractual SLAs with cryptographic methods to guarantee the confidentiality of cloud-hosted data. To attest to the fore-going statement, major companies such as Amazon, Google and Microsoft, are constantly releasing new features and updates to their services to meet legislative compliance requirements. For example, Amazon Web Services (AWS) has released a security patch called security 2 on their web services in order to strengthen its security [7].

Data confidentiality is indeed a major concern on the cloud. This is supported by [8] who asserts that data confidentiality is a major hindrance for the wide adoption and acceptance of cloud computing services. More often, cloud-hosted data can be accessed remotely using any Internet-enabled device, from anywhere and at any given time. Consequently, cloud-hosted data provides low hanging fruits for cybercriminals [2]. To this end, it is clear that a variety of problems exist on the cloud. The next section discusses the problem statement of this dissertation.

1.2 Problem Statement

Many cloud service providers offer cloud storage as a service. This is a new business model for delivering virtualized storage to cloud users. However, security is a top priority when it comes to choosing a cloud storage service. For example, CSPs such as Dropbox offer storage services on demand with modest security and confidentiality mechanisms. Dropbox uses Advanced Encryption Standard (AES-256), Secure Socket Layer (SSL) or Transport Layer Security (TLS) to preserve the confidentiality of cloud-hosted data [21]. SpiderOak is another CSP with better confidentiality guarantees that encrypts conversations and file sharing capabilities [22]. Amazon Cloud Storage (ACS) also offers cloud storage services. However, ACS's systems are relatively weak in terms of security safe-guards [24].

In general, cloud storage services are relatively cheap. As mentioned earlier, the inexpensive costs and scalable nature of the cloud attracts organizations and individuals to upload their confidential data in large data centres. Although there are service level agreements steered by legal policies binding CSPs to offer a safe environment for cloud-hosted data, there have been cases of security breaches that resulted in confidential data leaks.

Recently, Standard bank experienced a major breach from an Auto Teller Machine (ATM) cash-out scheme in Japan. This cash-out scheme reportedly resulted in \$19 million United States (US) dollars

being stolen from the South Africa's bank, in less than three hours [15][16]. This proves how sophisticated cybercriminals have become. Dropbox experienced a data breach where user credentials were stolen and used to reveal their cloud-hosted data [17]. In this attack, a cybercriminal could upload and link arbitrary files to a victim's Dropbox account using compromised user credentials. Despite the tight security measures that are applied to protect data on the cloud, attacks continue to breach systems at an increasing rate.

Moreover, CSPs may be bound by legislative measures to provide "back-doors" to third parties such as governments and law enforcement agencies to have access to customers' data. Dropbox recently stated that employees can see metadata only as the data itself is encrypted. However, when legally required to, some employees have access to view customers' data in plaintext form [23]. In addition to this, Amazon's Cloud Drive (ACD) does not offer any encryption at all. Under its terms of use, it is stated that the CSP can do whatever it likes with the user's data when required by law [24]. This might compromise confidentiality of cloud hosted data. Hence, there is a need to provide an extra layer of encryption before synchronizing data with CSPs, as described in [25].

Large organizations, in the financial sector and government agencies are the main target and have fallen victim to cyber-attacks. In addition to the above two cases of Standard bank and DropBox, below are two other cyber-attacks that are a telling testimony:

- Sony data breach [18];
- Target (TGT) Corporation data breach [19].

The major barrier to widespread adoption of cloud computing is security. Most organizations and individuals are hesitant to utilize the recent technologies which make cloud computing and the Internet of things a reality due to security concerns; data confidentiality in particular. The use of technology has not only brought efficient and convenient capabilities but has also opened doors for cybercrimes to thrive. Cybercrimes are reportedly on the rise almost every year. This proves that there is indeed a big problem in the cyberspace to curb such criminal activities. The following statistics compare cyber-attacks as well as their respective motivations in 2014 and 2015. It can be seen (in figure 1.1 and 1.2 below) that cybercrimes are the number one reason why cyber-attacks thrive year after year. The attacks shown in figure 1.1 are not world-wide but only the main ones reported and analysed to have sparked a lot of interest in the field of cyber-security. These attacks have been reported in various countries such as the

United States of America (U.S.A), China, Japan, Russia etc. Hence they are national attacks within the vicinity of their origin.

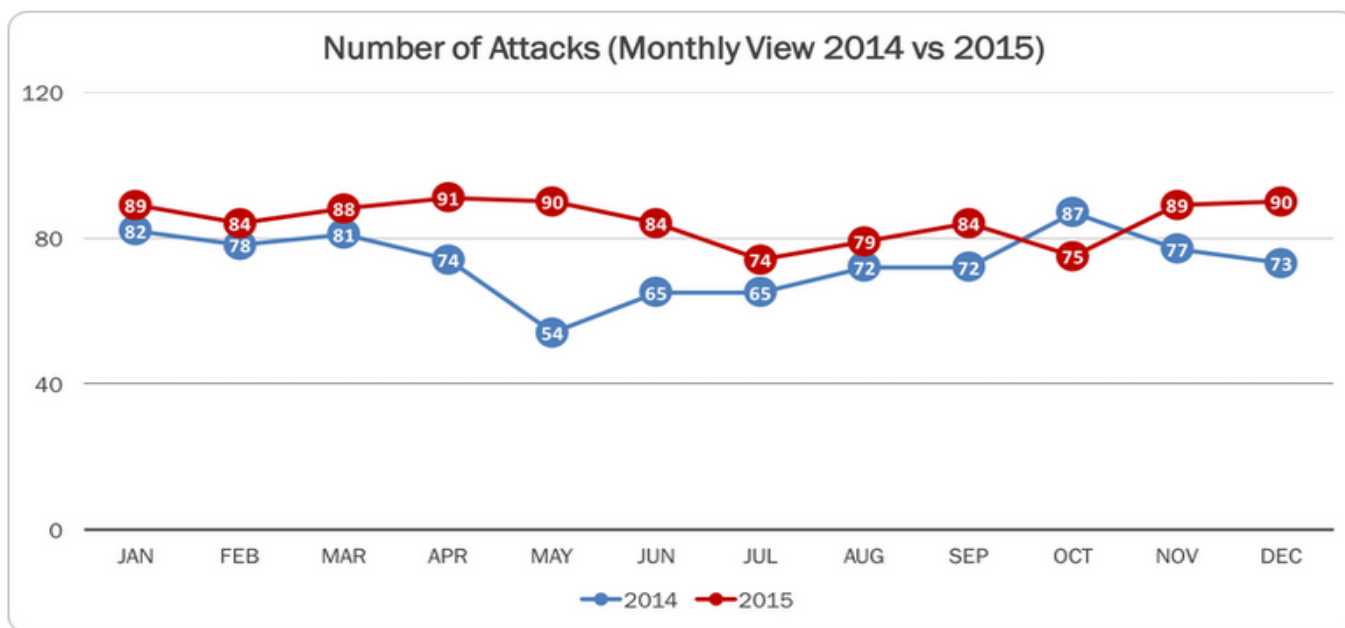


Figure 1.1: Monthly cyber-attacks [26]

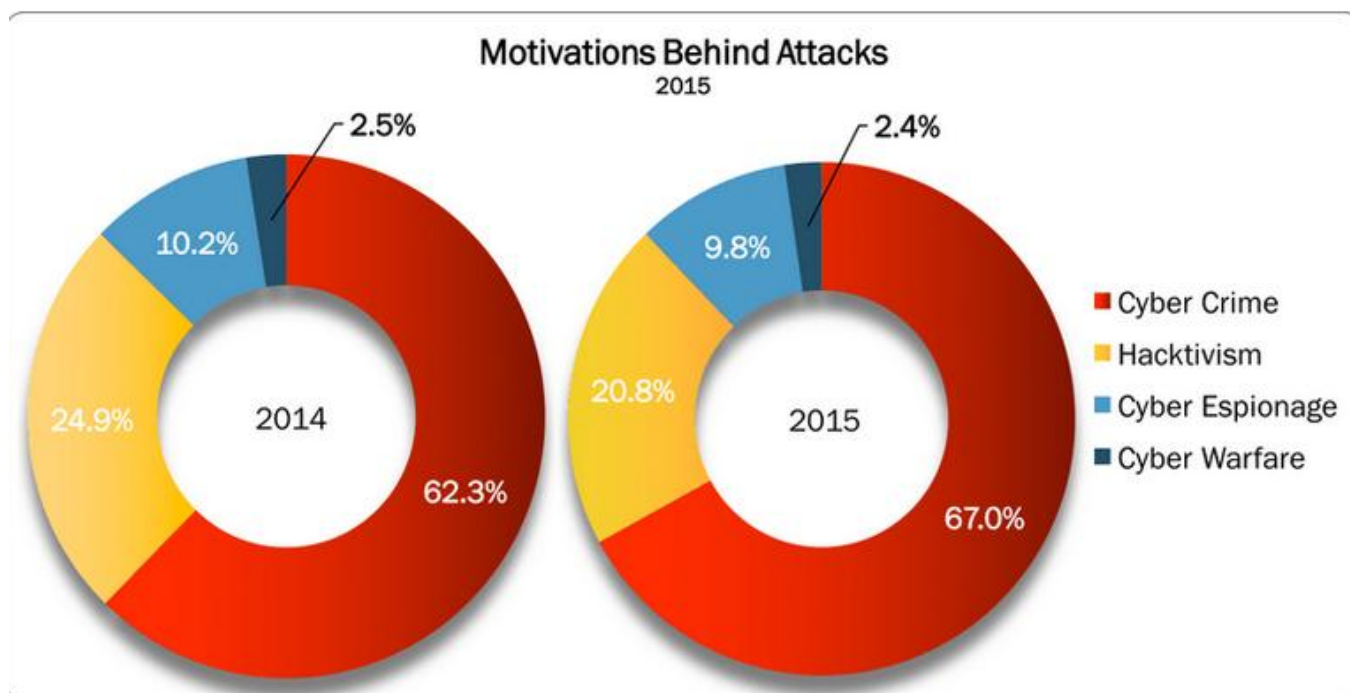


Figure 1.2: Motivation behind cyber-attacks [26]

Figures 1.1 and 1.2 above indicate that there is a big problem, related to cloud security. For example, figure 1.1 compares the number of cyber-attacks in 2014 and 2015. The figure shows that there is a

general increase in the number of cyber-attacks. That is, besides a few instances, the figure suggests that in years to come, if no measures are taken, there might be more attacks. This idea is reinforced by figure 1.2 which depicts the motivations behind cyber-attacks. It also indicates that cybercrimes hold the largest percentage share among other motivations. This becomes a motivation for this research which aims to curb the exponential rise of cybercrimes.

This research seeks to address some of the problems surrounding cybercrimes as stated earlier. The focus is placed on data confidentiality. To achieve the objectives of this research, a few questions must be answered. The next section discusses the research questions that this dissertation aims to answer.

1.3 Research questions

Based on the problem statement in the previous section, this dissertation primarily attempts to answer the following research question.

1.3.1. Main research question

- How can confidentiality of cloud-hosted data be ensured through the use of a novel cryptographic method?

To find an answer to the dissertation's main research question, three subsidiary questions are formulated to address the problem statement thoroughly. The subsidiary research questions are as follows:

1.3.1.1 Subsidiary research questions

- How can the strength of the encryption or decryption keys be improved without necessarily increasing the key length?
- How can key management, distribution and revocation in a cloud computing environment be conducted to ensure data confidentiality?
- How can the implementation of the proposed cryptographic system be practically demonstrated on a live public cloud computing infrastructure?

In a nutshell, the questions above seek to find answers to help design and develop a strong cryptographic solution using novel encryption algorithms which ensure confidentiality of cloud-hosted data. The three subsidiary questions above are the focal point of the body of this dissertation. The main research question

is re-visited in the last chapter of this dissertation. The hypothesized answer to this question is a successful implementation of a cryptographic system which may help to mitigate cyber-security threats and prevent insider threats existing in public clouds. Security is not only a combat mechanism to keep out the “bad guys” but it is equally used also on the malicious “good guys” who can equally be an adversary. The next section discusses the objectives of this study.

1.4 Research objectives

The aim of the research is to address the existing security concerns in the cloud. The focus is placed on data confidentiality. The goal is to determine innovative ways to provide strong data confidentiality guarantees for public cloud computing. The main objective is to design and develop a cryptographic scheme which preserves data confidentiality in the cloud. Given the complexity of the problem domain, the objectives are decomposed into more concise sub-objectives. This is done in order to create milestones that are meant to achieve the main objective. The following is a list of research objectives that must be fulfilled to reach the main objective of this dissertation:

- Design, model and implement of a novel cryptographic system that improves the strength of encryption keys, without necessarily increasing the key length;
- Design, model and implement a key management and revocation solution to perform key distribution and management;
- Extend and use the openNebula cloud to demonstrate the practicality of the proposed cryptographic system to show how it can provide data confidentiality guarantees.

The objectives listed above constitute the milestones expected from this research. Each objective is mapped onto a component of the proposed cryptosystem (presented in chapter four).

To evaluate the validity and relevance of the research conducted, exploratory research work is carried out. This research produced some publications to validate its contribution in the current body of knowledge within the field of Cryptography. The next section discusses some research outputs resulting from this research work.

1.5 Research outputs

The following is a list of published and presented research papers emanating from this research. The publications form part of the contributions of this study and are an evaluation of the dissertation’s topic.

- Mosola N.N., Dlamini M.T., Eloff J.H.P., Eloff M.M. (2016). *Evolutionary Neural Crypto-System for Cloud-bound Data*. Southern Africa Telecommunications Networks and Applications Conference (SATNAC), George, South Africa.
- Mosola N.N., Dlamini M.T., Blackledge J.M. (2016). *Cloud Multi-Party Secret Sharing Crypto-System*. University of KwaZulu-Natal postgraduate research day, Howard College, Durban, South Africa.

These published papers are included in the DVD accompanying this dissertation. The next section discusses the scope of the dissertation.

1.6 Dissertation scope

Cloud computing security is currently one of the challenges facing the wide adoption of this computing paradigm. Providing security requires a holistic approach, using various security mechanisms. This dissertation focuses on providing confidentiality guarantees for cloud-hosted data. Cryptographic methods are used to provide security. Encryption is one of the methods used in cryptography. Data confidentiality is offered using different encryption schemes. Hence, the scope of this research is solely based on providing confidentiality guarantees in the cloud. The proposed solution is only applied on openNebula cloud infrastructure. The next section outlines the dissertation structure.

1.7 Dissertation outline

This dissertation is divided into six chapters. The following diagram shows the organization of the dissertation.

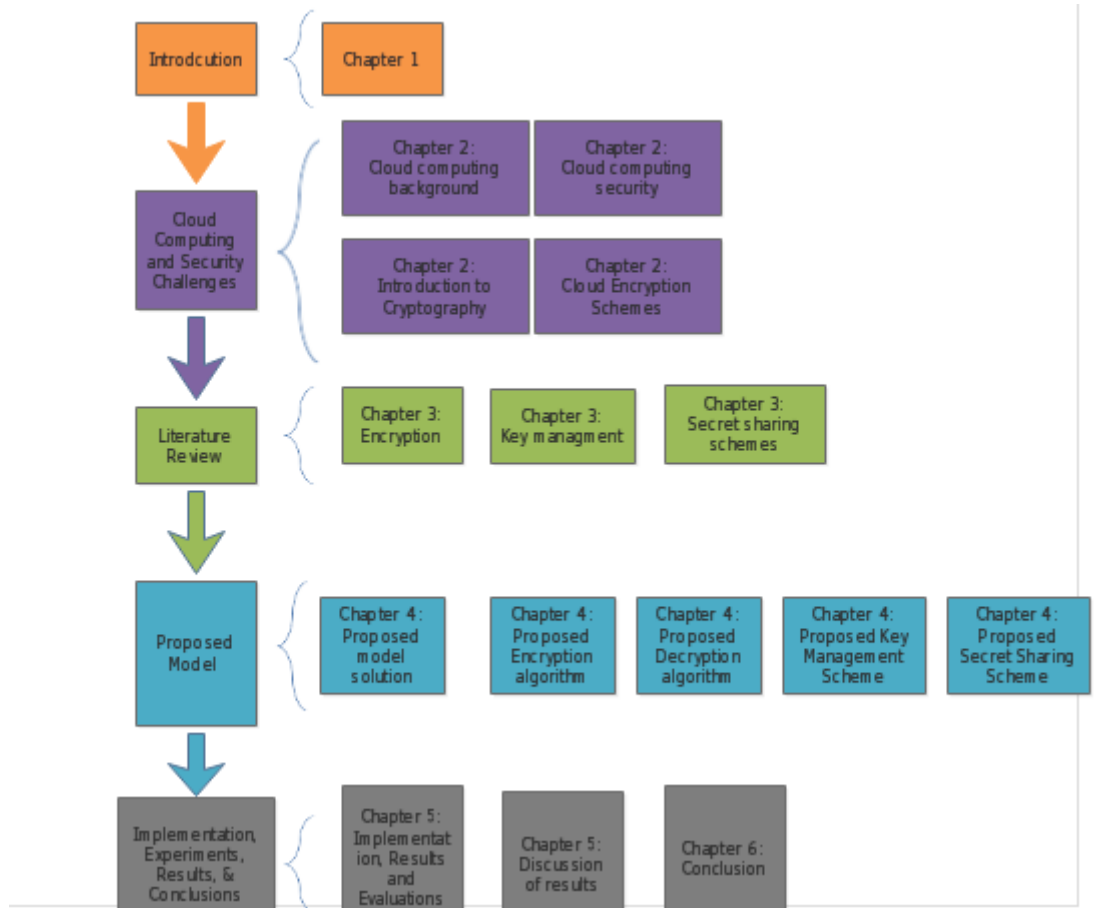


Figure 1.3: Dissertation structure

Chapter 1 – This chapter introduces the dissertation and forms the foundation of the entire study. It considers cloud computing from a security perspective.

Chapter 2 – This chapter provides a background on cloud computing concepts. Furthermore, it discusses cloud computing security with a specific focus on data confidentiality. Also in the discussions are various cryptographic schemes that can be applied on the cloud in the quest to provide data confidentiality.

Chapter 3 - This chapter discusses existing literature on encryption, key management and revocation on the cloud, and secret sharing schemes.

Chapter 4 - This chapter forms the gist of the dissertation and answers the research questions listed earlier. It first outlines the system’s requirements and presents the proposed cryptographic model which forms the solution to the research problem of this study. This chapter seeks to address the objectives presented in section 1.4.

Chapter 5 – The discussions entailed in this chapter seek to undertake analysis and discussion of the results from various experiments conducted during this research.

Chapter 6 – This chapter concludes the dissertation and provides recommendations for future work. The recommendations made are aligned to the global recommendations for cloud computing security listed in [28].

The next section concludes this chapter.

1.8 Conclusion

To this end, the discussions entailed herein show that cloud computing is plagued with a lot of security concerns. It is therefore crucial for cloud users to be aware of such security concerns. In particular, those concerns that target the confidentiality of their cloud-hosted data. Quite often, encryption is used to protect data confidentiality. However, existing encryption techniques seem to be vulnerable. The rising number of cyber-attacks attests to this point. To protect cloud-hosted data, new and innovative methods are required or an improvement of existing ones. Cybercriminals are being equipped with new techniques to gain unauthorised access to the cloud. Conventional information security concepts cannot directly fit into the cloud computing environment. This is due to the multi-tenant and distributive nature of the cloud. Hence, an entirely new, or a hybrid of existing security mechanisms must be devised. It is for these reasons that we embarked on this study. The proposed solution seeks to provide confidentiality guarantees to cloud users and keep the cloud computing environment secure.

Therefore, the next chapter gives a detailed background of cloud computing and its security challenges. It also discusses various cryptographic schemes applied in cloud computing in the quest to defeat the challenges thereof.

This page is intentionally left blank.

Chapter 2

Background: Cloud Computing and Security Challenges

2.1 Introduction

The previous chapter introduced the study and illustrated the actual research problem, objectives and outlined the structure of the work. This chapter sets the scene by discussing some of the concepts around cloud computing and the security challenges thereof. The focus of the discussions is on cloud computing services, deployment models, essential characteristics, security challenges and cryptography. The next section delves into the background of cloud computing and defines this new paradigm.

2.2 Cloud computing definition and Background

Several researchers have already defined cloud computing. Hence, there are various definitions of cloud computing that exist today. The next sub-section defines cloud computing and briefly discusses virtualization and the architecture.

2.2.1 Cloud computing definition

According to [29], cloud computing is a model which enables ubiquitous, on-demand access to a shared pool of configurable resources, in the most convenient manner. These resources may include networks, servers, applications and storage services. This computing model abstracts hardware, platform, infrastructure and software as services. [29] further states that a cloud computing model consists of five essential characteristics (i.e. broad-network access, resource pooling, measured services, scalability and rapid elasticity, on-demand self-service), four deployment models (i.e. public, private, community and hybrid clouds) and three service models (i.e. infrastructure as a service, software as a service and platform as a service). Cloud computing can also be described as a descendant of grid computing [30]. Grid computing is defined as an infrastructure providing dependable, consistent, pervasive and inexpensive access to computational resources, using both hardware and software [30]. Based on the above definitions and for the purposes of this study, cloud computing is a model in which computational resources are conveniently shared and services are seamlessly provisioned through the Internet. Cloud computing relies on the concept of virtualization. The next sub-section discusses virtualization and the cloud computing architecture.

2.2.2 Virtualization

Virtualization technology empowers cloud computing in all aspects. In its implementation, a physical host machine is equipped with an application called the hypervisor. The hypervisor creates virtual machines which are hosted on the physical machine. The virtual machines are capable of simulating physical computers in every aspect. The virtual machines can run any software, including operating systems and application software [31]. At hardware level, physical devices such as hard disks, processors, and network devices are managed in data centers. The data centers are responsible for storage and processing requirements. Using this concept, physical resources are dynamically allocated to cloud users. These resources include computational power, storage and networks. These resources are delivered to cloud users as services. Thus, virtualization is a critical element of cloud computing and its implementations. Above all, virtualization enables the cloud to portray its essential characteristics.

2.2.3 Cloud computing architecture

Thus, according to [30] virtualization is the key difference between grid and cloud computing. The following diagram is adopted from [29]. It depicts the cloud computing architecture. The architecture comprises of deployment models, service models and essential characteristics.

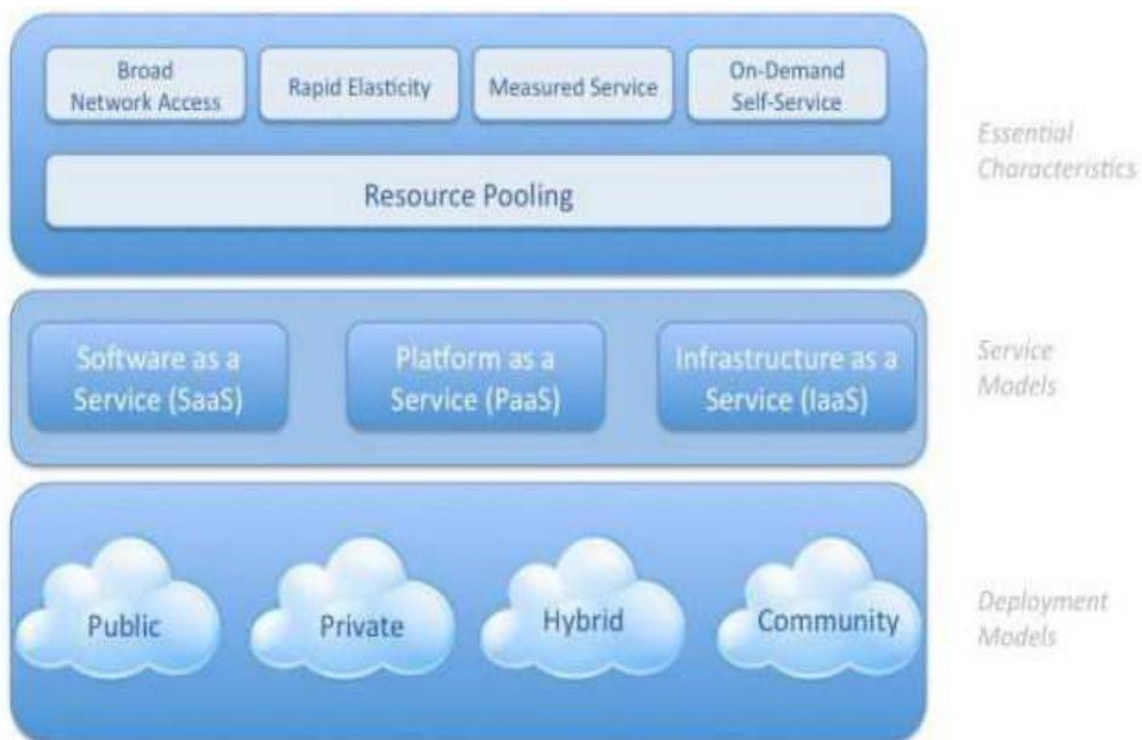


Figure 2.1: Cloud computing model architecture [29]

To grasp the details on cloud computing and its adoption, the next sections discuss the details on different types of services, deployment models and essential characteristics of cloud computing as depicted in figure 2.1 above. It further explains how cloud users can make use of the cloud to exploit the on-demand services, ubiquitous network access, rapid elasticity and location independence.

2.2.3.1 Cloud computing services

Cloud computing can be delivered using three distinct service models. These are: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). These service delivery models provide infrastructure resources, application platform and software as services to the cloud users. IaaS is the foundation of the cloud services. PaaS is built on top of IaaS and SaaS in turn is built on top of PaaS in the services stack [29]. The next sub-section discusses each of the service delivery models in detail.

2.2.3.1.1 Infrastructure as a Service

IaaS is the most basic yet important service model. This is where virtualization is widely used to provide virtual machines, load balances, networking etc. This service model is responsible for providing cloud users with fundamental computing resources which include the following capabilities; processing, storage, networking etc. This service model grants users with the infrastructure to run application and system software in an on-demand manner. Using this service, the cloud user has control over the storage provided, software, deployed applications and selected networking components. IaaS is the most extensible service delivery model compared to the rest [29]. It provides very few application-like features. For example, resources are aggregated, managed and delivered as storage capacities. Examples of this service include RackSpace [33], Amazon Simple Storage Services (S3) [34], Amazon Elastic Compute Cloud (EC2) [33][35] etc.

As an example, for computational purposes, Amazon EC2 offers resize-able computational capacity in the cloud. It allows minimal computation times depending on the speed of the central processing unit (CPU) available for a job request [36]. IaaS is a service which enables cost savings. Using IaaS, there is no need to procure and manage new infrastructure locally. Instead of spending and creating data centers locally, IaaS offers the infrastructure on the cloud. An example of a service provider offering IaaS is Amazon Web Services (AWS). Using AWS, a virtual server is created and offers services on a pay-per-use model. IaaS also abstracts physical hardware components, allowing cloud users to utilize IaaS without getting into the complexities behind managing the infrastructure [29]. Cloud users run any system and

application software which meets their demand. However, they cannot manage or control the underlying infrastructure. Next in the discussions is PaaS.

2.2.3.1.2 Platform as a Service

This is a service delivery model which provides a toolkit with an integrated development environment (IDE) for developers. Developers can conveniently develop, deploy and administer their application's software with relative ease using programming languages and tools supplied by the CSP. The goal is to allow developers to create their own software applications and deploy them on top of the provided development platforms. It enables programming environments to have access to additional application packages [36][37]. Developers can also execute customizable applications which target a specific platform with the tools provided by the CSP. PaaS offers developers a service which provides a complete software development life-cycle (SDLC) and management. This includes initial planning, development, deployment, testing and maintenance. Application developers, testers and administrators do need to worry about computing resources (e.g. processing, storage, networking etc.) that are needed to operate the developed applications. CSPs are then responsible for isolating customer's applications and workspaces from one another [36]. Generally, PaaS is the same as any traditional computing platform as software applications are developed and run on PaaS clouds.

Furthermore, the PaaS model includes programming languages, operating systems, databases, web services etc. For example, Google App Engine [37] and Microsoft Azure platform [38] offer PaaS. These platforms allow developers to design, develop, execute and deploy services through an application interface. PaaS also enables collaboration such that multiple users can work on the same project simultaneously, thus increasing productivity. An example is GitHub [39]. The next subsection discusses SaaS.

2.2.3.1.3 Software as a service

This is a software service delivery model which provides software applications over the cloud. Using SaaS, CSPs offer software applications as a service. This service model is solely responsible for providing cloud users with the capabilities to utilize CSP's software applications. For example, email, Microsoft office, archiCAD drawing tools etc. CSPs ensure that the software applications they provide over the cloud are updated and well maintained. SaaS can be integrated to generate new services delivered via flexible networks [41]. For example, instead of buying and installing software on stand-alone systems, cloud consumers use the provided software applications on a pay-per-use basis. Examples

of SaaS delivery models include GoogleDocs [42], Salesforce [40][43], Dropbox [44], Office 365 [45], GMail etc.

The software applications may be accessible remotely and from any client device, through a program interface such as a web browser. Using this service, cloud users do not control or manage the underlying cloud infrastructure [29]. SaaS provides scalability and shifts the administration from users to CSPs. The CSP, who is either a third-party entity or a cloud vendor, is responsible for managing the entire service from the application level down to the lowest granularity in the infrastructure. The CSP is also responsible for formulating application-tier security policies which form part of the SLA. The SLA binds the CSP to offer security guarantees for the clients using SaaS. The clients must ensure that the SLA meets their confidentiality requirements before using such services.

However, it is not necessary for a CSP to physically own the infrastructure running the software. The main advantage of SaaS is that it is easily available and accessible. SaaS is normally accessible on demand via the Internet, in a remote manner. Hence, this service is inexpensive and location independent. Providers of SaaS include Google Apps, Facebook, and Salesforce [43]. The next section discusses cloud computing deployment models.

2.2.3 Cloud computing deployment models

As depicted in figure 2.1, the cloud computing model has four deployment models, namely: public clouds, private clouds, hybrid clouds and community clouds. The next sub-sections discuss each of these in detail.

2.2.3.1 Public clouds

This type of a cloud deployment model is open for use by the general public. Public clouds exist on the premises of the cloud service provider. Public clouds depict the basic concept of the pay-per-use model offered by the cloud computing paradigm [41]. Ideally, using public clouds, organizations may use cloud services offered by other organizations. This creates a cloud of clouds.

Furthermore, public cloud deployments help companies to leverage a shared pool of resources and services. Hence, there is no need to have the cloud set up within a client organization's premises. This aspect consequently reduces the costs of building and maintaining the public cloud infrastructure from the client's perspective. For example, the New York Times archive project consists of 100 Amazon EC2

instances and 5.5 Terabytes (TB) of S3 storage. This storage is a digital library for digital content. All this comes at a fraction of what it would cost to archive digital content using traditional archiving methods on a local server hosted on-premises [41]. Public clouds are ideal for deployment, development, and enterprise applications management. They come at affordable costs. However, there are serious security challenges related to public cloud deployments. These security challenges are discussed later in this chapter.

Moreover, one advantage of public clouds is concurrency control [29]. In a public cloud infrastructure, a cloud service provider provisions various services to multiple unrelated cloud users who access services running concurrently and independently. These users may be individual tenants or other cloud vendors. Cloud users do not need to worry about the maintenance or the security of the underlying infrastructure. It is the responsibility of the cloud vendor to provide adequate security guarantees and maintain the protection of the data in the public cloud. It is important to note that in a public cloud environment, all cloud users share the same configured infrastructure as it is managed by the cloud service provider. The figure below is adopted from [33]. It highlights various companies which are currently deploying public clouds.



Figure 2.2: Companies offering private clouds [33]

The next section discusses private cloud deployments.

2.2.3.2 Private clouds

Private clouds are also known as internal clouds. This deployment model is ideal for exclusive use such as storage, computations by a single entity. This deployment model is set up on a private network and is owned by a single organization. Private clouds offer maximum control over performance and reliability. This is because the infrastructure is usually hosted on the organization's premises. However, private clouds may also be hosted off-premises. Using this type of cloud deployment, the security configurations may be managed and controlled within the perimeters of the organizations owning the cloud.

Legitimate access may be granted to clients from outside the security perimeters of the organization. However, access must be monitored by a boundary controller, such as a firewall to ensure some level of security is provided. The figure below is adopted from [48]. It depicts an on-premises private cloud deployment model.

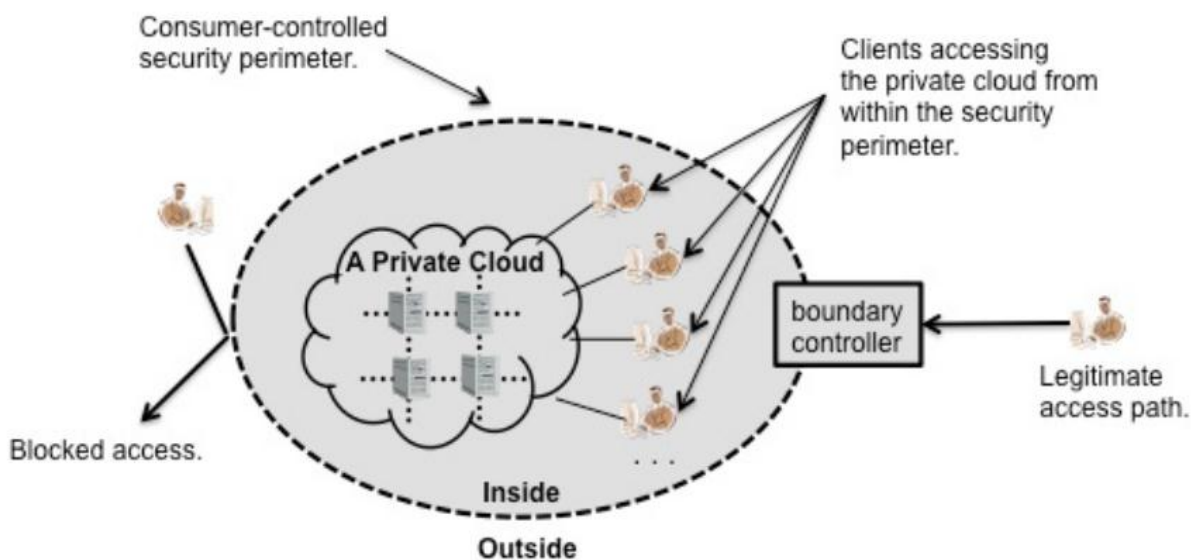


Figure 2.3: On-premises private cloud model [48]

An organization may purchase or lease the required infrastructure to set up private clouds off-premises. The CSP may lease the infrastructure to the cloud consumer (e.g. an organization). However, this service model does not benefit from low costs and reduced maintenance as public clouds do. This is because the environment is completely locked internally and not shared [28]. Access from outside the perimeter can

only be granted to users on legitimate paths. The next sub-section discusses community cloud deployment models.

2.2.3.3 Community clouds

Community cloud deployment models are provisioned to be used by a distinct group of users grouped in a community. These users may be from several organizations which have shared concerns. For example, policies, security requirements, compliance considerations etc. [29]. The figure below is adopted from [48]. It illustrates the scenario of setting up community clouds.

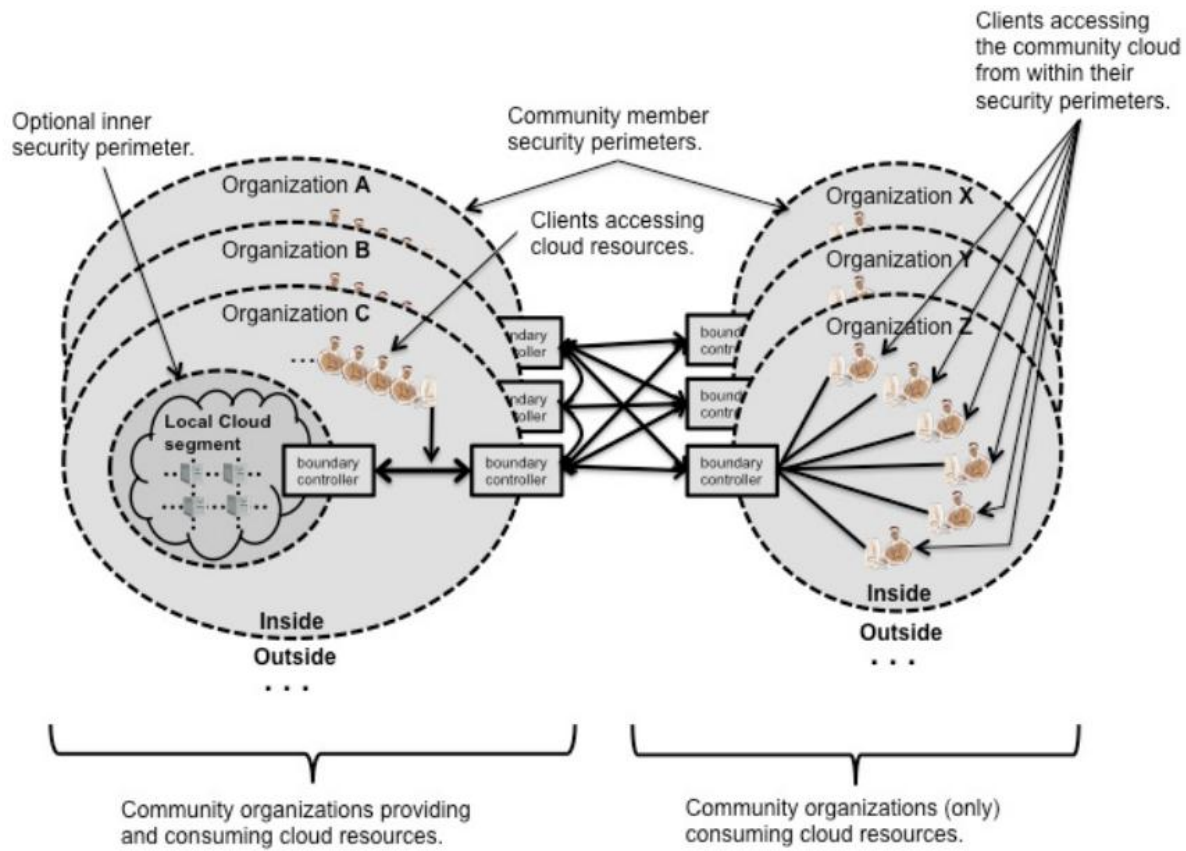


Figure 2.4: Community cloud deployments [48]

In figure 2.4 above, organizations on the left are rendering services to organizations on the right. A security perimeter separates the inner networks from the outside networks, just like in the private cloud scenario. The communication between the organizations providing services and the ones receiving services is offered through the boundary controller of each organization.

Using a community cloud deployment model, the underlying infrastructure may be owned, managed and operated within a community by a single organization or co-operated by multiple organizations which form part of the community. This is done just like in private deployments. Third parties can also operate this cloud deployment on behalf of a community of organizations or a combination of entities in an off-premise, outsourced scenario. The infrastructure may be hosted on or off premises [46][48]. The next subsection discusses hybrid cloud deployments.

2.2.3.4 Hybrid clouds

This cloud deployment model is a combination of two or more cloud deployment models. Hybrid clouds are now being widely used by CSPs to offer cloud services. The figure below is adopted from [52]. It depicts companies running hybrid clouds. These include Amazon web services, rackpace hosting, GoGrid etc.



Figure 2.25: Companies offering hybrid cloud deployment [52]

Hybrid cloud deployments are normally a composition of private, public or community clouds [49][52]. This is realized when a private cloud vendor merges its services with a public or community cloud vendor

or vice versa. Despite the integration of two or more cloud deployments; each deployment model remains as a unique entities. The integration represents a loose coupling as different cloud deployments can be added or removed with relative ease. This means that a hybrid cloud can change over time with constituent cloud deployment models being integrated or detached from the existing ones. This makes hybrid clouds flexible and highly scalable. Due to the loose coupling, this type of cloud deployment exhibits advantages of the constituent cloud deployment models, either private, public or community clouds. For example, if it has a public cloud deployment, it achieves maximum reduction of costs through outsourcing as a feature of public clouds. Hybrid clouds also achieve the highest degree of control over cloud hosted data as a feature of private clouds.

Despite their advantages, organizations are reluctant to place customer data in hybrid clouds. This is because hybrid cloud deployments inherit the same weaknesses as the constituting cloud deployment models, especially if public clouds form part of the hybrid. Having a hybrid cloud deployment allows organizations to host their highly confidential corporate data in a private cloud while using the public portion of the hybrid cloud for less confidential data [50][51]. In a hybrid deployment, an organization may leverage services from third-parties in full or partial manner. Leveraging services from third-party entities increases flexibility. This is because in a hybrid cloud, organizations may provide on-demand, externally managed scalability [51].

Hybrid cloud deployments may consist of public and private cloud attributes which complement one another. One major advantage of hybrid clouds is the ability to effectively meet various organizational requirements. However, it should be made clear that hybrid cloud environments inherit the disadvantages of cloud deployment models they are made of. For example, a hybrid cloud deployment could result in confidential data leaks to unauthorized users if the bigger part of it comes from a public cloud deployment.

The next section focuses on the various characteristics of the cloud computing paradigm.

2.2.3.5 Cloud computing model essential characteristics

The following are the essential characteristics of cloud computing.

- **On-demand self-service:** Cloud users are provided with computing resource capabilities such as server time and network storage when they need them. This is done in an autonomous manner with no human interaction from a cloud service provider that provisions the services.

- **Broad network access:** The cloud consists of multitudes of interconnected networks. Access to these networks is controlled through networking standards and protocols that promote access to various heterogeneous thin or thick client platforms. Examples of such platforms are mobile phones, tablets, laptops etc. Without the broad network access, cloud computing has limited capabilities or worse still it cannot be realized.
- **Resource pooling:** Cloud service providers offer a pool of resources to multiple cloud users through a multi-tenant environment. Using a multi-tenant model, cloud users are dynamically assigned and reassigned as per their demands to resources that have been put together by the CSP. This aspect of cloud computing is location independent. Generally, cloud consumers are without a clear knowledge of the exact location of the resources they are accessing. However, locations can be specified at a higher abstraction level such as country, state or data center.
- **Rapid elasticity:** Commensurate with demand, cloud computing capabilities can be elastically provisioned, manually or automatically. This is used to scale resources upwards or downwards depending on the level of demand. To cloud consumers, cloud computing capabilities appear to be unlimited and can be requested in any quantity or queried at any time.
- **Measured service:** Cloud-based systems control and optimize resource utilization by having monitoring and metering capabilities. This is done in an autonomous manner. However, this can be done at a level of abstraction that is appropriate to the type of a service being rendered. For example, storage, processing, memory etc. Resource utilization can be controlled or monitored, using a resource monitor.

Now that a general discussion has been laid down on cloud computing, the next section shifts its focus to the security challenges of cloud computing. The focus is placed on cloud storage and data confidentiality.

2.3 Cloud computing security challenges

While cloud computing brings flexible, scalable and economical services, there are security concerns that challenge this new computing model. At the basic level, cloud users need to be aware of security concerns involving customizations, control delegation, and provider dependency when considering cloud facilities and services.

There is an exponential increase of data being uploaded on the cloud, mostly for storage or processing. With this increase comes a potential for confidential data to be leaked, either intentionally or unintentionally. Given that cloud computing provides outsourced services, it must be noted that such services evade the physical, logical and personal control over cloud-hosted data. Once the data is transferred and uploaded to the cloud, CSPs assume responsibility to keep it secure. Although CSPs do not guarantee such security nor take accountability should there be a data breach or leakage. For example, Amazon's web service level agreement states that the CSP will strive to keep the data secure, but cannot guarantee success [35]. This poses an intrinsic threat to users utilizing cloud services.

Thus, as one of the risks, for example, data leakages compromise the confidentiality of cloud-hosted data. When there are weak or no confidentiality guarantees, the consequences of data breaches might have a cascading effect on the entire cloud computing model. This is because cloud service models are inter-related and interdependent. Due to these relations and dependencies, any attack launched on any of the service layers can also compromise the other unrelated layers. Each of the cloud services and deployment models are associated with their own security concerns that are unique to each of them. For example, confidential data leakage threats are a main concern in IaaS and not so much so on PaaS and SaaS.

Numerous security standards have recently surfaced in a bid to protect the confidentiality of cloud-hosted data. However, these standards are not well equipped to tackle many security challenges associated with the uncompromising cloud computing environment. This section specifically considers the security challenges associated with cloud computing, with a specific focus on data confidentiality or lack thereof. Due to the distributed nature of the cloud, data confidentiality is often compromised. This may happen in various ways when a piece of data is transferred from one point to the other on the distributed cloud. This data is often stored in physically separated data centers. Thus, moving and storing data in a distributed, multi-tenant environment may affect its confidentiality. This research assesses the risks associated with storage clouds.

2.3.1 Encryption and key Management

Often, CSPs offer data encryption in the cloud to preserve confidentiality of their tenants' data. Thus, most data owners send their data to cloud storages in an unencrypted format. Sending data to the cloud in an unencrypted form carries a lot of risk. For example, it opens the client data and makes it vulnerable to man-in-the-middle attacks, insider attacks, identity theft, among others. This is because intruders target

data repositories to gain unauthorized access to users' data. In order to minimize such risks and curb these types of attacks, there is a need to strengthen security mechanisms that enforce data confidentiality. Such mechanisms must be combined with strong encryption and other security mechanisms. This can help isolate confidential data from the security mechanisms used to secure it, more especially the encryption keys. This study argues that encryption keys must be kept outside CSPs data center. Data owners must take the responsibility to secure their data and keep their keys in the premises before sending it to the cloud.

Trusted third parties are often brought into the picture to provide key management and storage facilities [53]. However, the fact that users' data is stored in a remote, distributed data center and the keys to secure the data are provided by a third party might be the reason behind the increasing number of cyber breaches. A variety of encryption schemes, key recovery and trusted third party encryption requirements have been suggested by government agencies in order to enable them to conduct covert surveillance in the cloud and new rising technologies. This also enables government agencies to have "back-doors" in accessing user's data. In order to address such cases and prevent compromising the confidentiality of cloud-hosted data, [53][54][55] suggest that data protection protocols such as Secure Sockets Layer (SSL) or its successor Transport Layer Security (TLS) should be used to protect data while in transit. [54] also attests to protecting data using homomorphic encryption while at rest (i.e. stored in data repositories) and when in use.

2.3.2 Data leakages

Data leakages associated with the cloud continue to escalate [29]. This is because the cloud is flooded with "*data lakes*". Cyber-criminals target and exploit any vulnerability to get their hands on the data in an unauthorized manner. Such actions eventually compromise data confidentiality. To attest to this, the following examples are given: Dropbox was breached and user credentials were stolen and leaked [55]. In addition to this are the six million passwords which were stolen from eHarmony and LinkedIn [56]. This shows that risks associated with cloud computing are becoming increasingly high. According to [29], cloud computing represents a complicated service offering because resources are shared amongst multiple cloud consumers.

Some of the data leakage threats on the cloud target the vulnerabilities of the underlying infrastructure. For example, in a public cloud, a host machine with multiple virtual machines may be compromised by one of the guest virtual machines belonging to a malicious user. An attacker may be able to map a CSP's

underlying infrastructure in order to determine VMs running on a particular host machines and target them. Then, an attacker may create their own malicious VM to co-reside with the targeted VMs on the chosen host. If such co-residence is achieved, it makes it easy for an attacker to gain unauthorized access to other co-resident VMs. Consequently, the malicious VM may be used to leak confidential data residing on legitimate co-resident VMs [2]. Such attacks are referred to as Inter-VM attacks [2]. The next section discusses partial deletions.

2.3.3 Partial deletions

A partial deletion occurs when a data owner deletes cloud-hosted data but remnants of the data remain on the CSP's side. The data is normally not completely deleted. For example, remnants of data can exist on the CSP's servers even after the cloud consumer has stopped utilizing the services and the data has been erased by the consumer. Intentional or unintentional partial deletions affect the confidentiality of cloud-hosted data. For example, all data deletions must use effective data sanitation methods. Hence, deletion of cloud-hosted data must be handled with great care from both the client's and CSP's sides. Each party must ensure that the deletion process is complete and not partial.

Due to the virtual separation and lack of hardware separation in a multi-tenant cloud environment, the residual data stored on the physical hosts may lead to unintentional data leaks or intentional exploitation of these vulnerabilities. Moreover, forensic tools can be used by both cyber-criminals and law enforcement agencies to restore deleted data from the CSPs servers. In addition to this, a rogue employee can launch an insider attack to recover insecurely deleted customer data. An example of such cases is the eBay online store data breach in which disposed customer data was recovered from the eBay servers [57].

2.3.4 Data co-residency

One of the downsides cloud computing is co-location of confidential data. Given that the cloud is a shared environment, it brings increasing number of risks when data from two or more conflicting cloud users is placed on the same physical host. This is more so, in the case of public cloud environments. Given that data comes from different cloud users who are co-residing on the same host, unauthorized entities may try to access other residents' data. Resource sharing is economical in most instances. However, the risk is sometimes too high to bare [58].

2.3.5 Loss of physical control

This is one of the most concerning downsides of cloud environments. When users send their data to the cloud, they somehow lose control over all or part of it. With IaaS, the cloud user may have partial control of the infrastructure and its configurations. However, in PaaS, the CSP manages the hardware and the underlying infrastructure. This limits the cloud user's risk management capabilities [37]. In addition to this, in SaaS, both the underlying infrastructure and the platform are managed by the CSP [29]. This means that if the underlying infrastructure or service is not configured correctly, the data in the higher layers may be at risk.

2.3.6 Identification and authentication

Identification and authentication refers to testing whether an entity is indeed a legitimate cloud user using a unique user identity. User identification and authentication methods must be tightened to ensure that only authenticated users have access to cloud-hosted data [29]. Authenticated users are then authorized to access certain resources which they have legitimate rights to utilize. This is mainly because cloud services are accessed remotely. Hence, strong identification and authentication methods, coupled with authorization mechanisms become a critical concern in a multi-tenant and distributed environment such as the cloud.

2.3.7 Insider threats

Insider threats are malicious acts by entities within an organization. These people include current employees, former employees who still have access rights to the organization's resources, business partners, third party contractors etc. Insider threats thrive because of rogue entities that may have access to cloud users' data. Malicious entities may cause substantial damage, given the access and authorization granted to them. Given that the CSP has some control on the cloud users' data, methods to ensure data confidentiality must be improved. Literally, CSPs have physical access to the data stored on the cloud. CSPs know the location of the data while the data owner does not know which data center hosts the data. This makes it easy for the CSPs employees to become an adversary.

In a cloud environment, where resources are shared among multiple tenants, confidentiality is a huge concern. Confidentiality implies that client's data and computational processes must be kept secret from unauthorized entities. Due to loss of physical control over cloud-hosted data, confidentiality is often compromised. In public cloud deployments, data co-residency may lead to potential risks which could affect the confidentiality of cloud-hosted data. To ensure data confidentiality, cryptography is often used.

The next section delves into cryptography in a cloud computing environment. Cryptographic methods are ideal for providing data confidentiality. Most researchers implement encryption methods in order to preserve data confidentiality. Encryption methods are cryptographic tools. In a cloud environment, data confidentiality is a huge concern. Hence, cryptography must be applied to secure the cloud. However, implementing cryptographic methods in the cloud is a huge challenge given the multi-tenant and distributed nature of the cloud. The next section discusses the two types of cryptographic schemes.

2.4 Cryptographic schemes

As noted from previous discussions, cryptography and key management is one of the challenges of cloud computing. This section gives some background on cryptography and key management in the cloud. This is to set the scene on how cryptography has traditionally been used to preserve data confidentiality on the cloud. In this section, an introduction of some cryptographic schemes that are currently being used to provide data confidentiality guarantees for cloud storage is laid down. It must be noted though, that most of the cryptographic techniques presented in this section are partially meant for deployment in the CSP's side. However, they can also be applied on the client's side. This would help to protect the interests of both clients and providers.

In essence, there are two cryptographic schemes [59], namely:

- Symmetric key cryptography;
- Asymmetric key cryptography.

A fundamental distinction between the two schemes is related to key management and revocation, encryption and decryption algorithms. The following sub-section delves into symmetric key cryptography.

2.4.1 Symmetric key cryptography

Symmetric key cryptography relies on a single key that is used for both encryption and decryption. An encryption algorithm is used to conceal message contents. A decryption algorithm is used to recover the original message from the unintelligible encrypted message. In most cases, the decryption algorithm is the reverse of the encryption algorithm. The figure below is adopted from [60]. It depicts a symmetric key cryptosystem model.

A symmetric key cryptosystem consists of the following:

- Plaintext – This is the original message, which is readable to the human eye;

- Encryption algorithm – A systematic process used to encrypt the plaintext;
- Symmetric key – Used together with the encryption or decryption algorithm;
- Ciphertext – The unintelligible message which is the output of the encryption process;
- Decryption algorithm – The process of recovering plaintext from ciphertext.

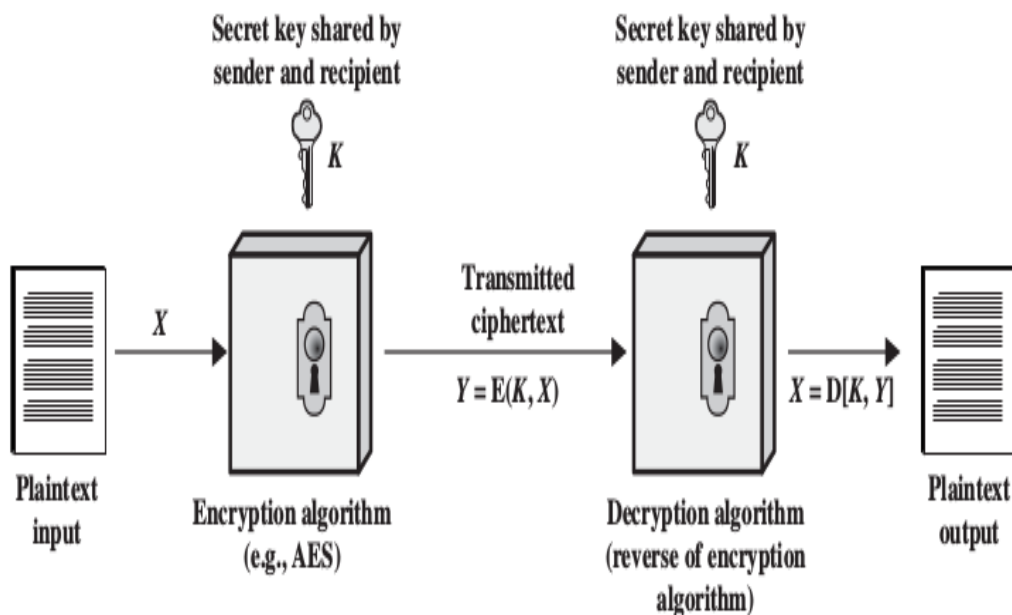


Figure 2.26: Symmetric key cryptosystem model [60]

Assuming two entities, (i.e. the sender and the receiver) want to share a plaintext file in a manner that preserves the confidentiality of the plaintext file. As illustrated in figure 2.6, a symmetric key cryptosystem takes as input, a plaintext file, X from the sender's side. An encryption algorithm is used to encrypt the plaintext. The encryption algorithm, which is implemented as a function denoted by the letter E , is used in conjunction with the secret key, K . Applying an encryption function on the input yields the ciphertext, denoted by the letter Y , i.e.

$$Y = E(K, X) \tag{2.1}$$

On the receiver's side, a decryption algorithm is used. The symmetric key (i.e. K) and ciphertext (i.e. Y) are used as inputs into a decryption function D . This yields the plaintext i.e.

$$X = D(K, Y)$$

(2.2)

This model is adopted by various cryptosystems. Examples of symmetric key cryptosystems are Data Encryption Standard (DES), its variant triple DES, and Advanced Encryption Standard (AES) etc. The latter was widely adopted for cloud use by Amazon S3 [61].

Despite the advantages of symmetric cryptography, including less computational costs, efficiency, simplicity etc., which make the scheme more suitable for processing large streams of cloud-hosted data, the scheme presents key distribution and revocation problems. The assumption is that any communicating entities are able to exchange the symmetric key first, over a secure channel. This makes it difficult when it comes to a multi-tenant, distributed environment, such as the cloud. As such, key management and revocation is a significant challenge in the cloud. Hence one of the objectives of this research is to develop, model and implement key management system. Several researches efforts have considered different cryptographic methods which could improve on cloud performance as far as encryption is concerned. This is done to provide effective and efficient methods of preserving confidentiality guarantees in the cloud.

The next section discusses asymmetric key cryptographic schemes.

2.4.2 Asymmetric key cryptography

This cryptographic scheme is based on having two mathematically related keys, referred to as the public and private keys. Any one of the keys can be used to encrypt data, but the other is used for decryption. Asymmetric key cryptography was coined to address various problems encountered with the use of symmetric key cryptography. It ensures several security properties, for example, confidentiality, integrity, non-repudiation etc. In contrast with symmetric cryptography, where communicating entities must share a symmetric key, public key cryptographic schemes have a pair of keys for each entity, the public and private key set. The public key is shared and made public to every peer while the private key is known only to the owner [60]. The figure below is adopted from [60]. It illustrates a model for an asymmetric key cryptosystem.

The components of a public key cryptographic scheme are as follows:

- Plaintext;
- Encryption algorithm;
- Encryption key – either public or private;

- Ciphertext;
- Decryption algorithm;
- Decryption key – depends on which key was used for encryption;

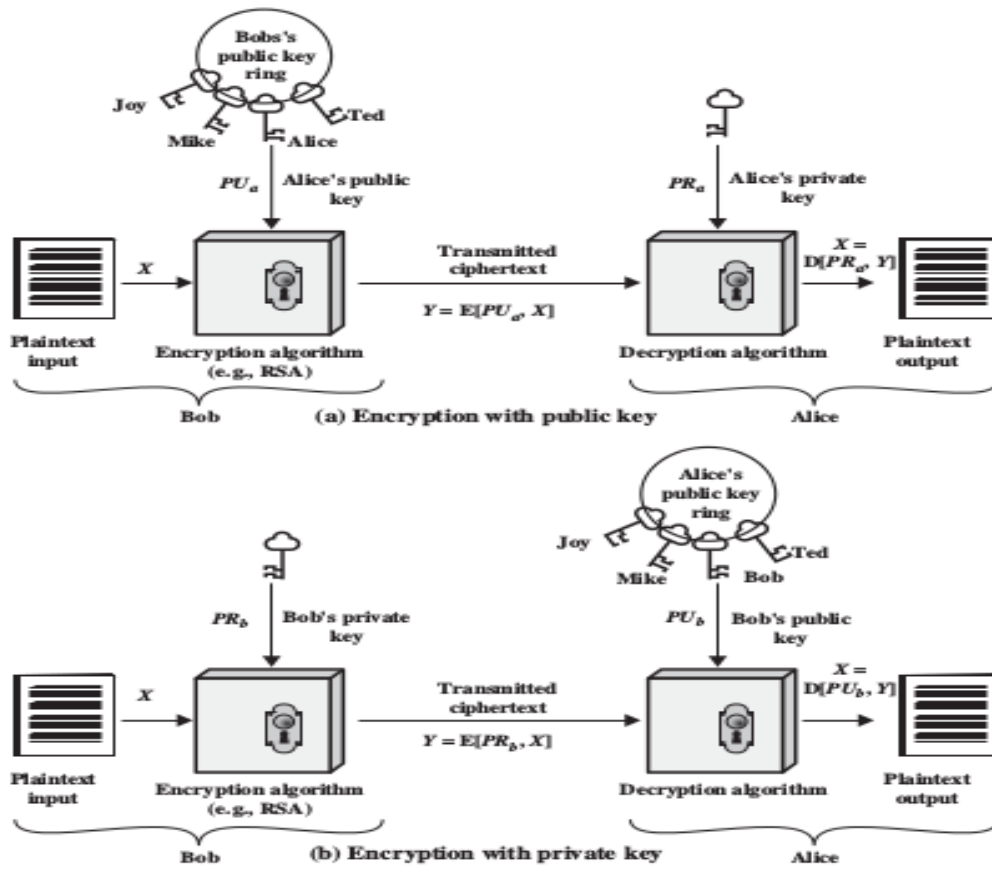


Figure 2.27: Asymmetric key cryptosystem model [60]

Assuming two entities want to share a plaintext file without compromising the confidentiality of the shared plaintext file. As illustrated by figure 2.7(a), the plaintext file, X , originates from the sender's side. The sender keeps a public key directory (also called public key ring), which is a list of all entities communicating with the sender. The sender selects a public key of the intended recipient of the plaintext file from the public key directory. The recipient's public key is denoted as PU_r . The plaintext and the recipient's public key are used as inputs into the encryption algorithm, E to produce the ciphertext, Y . i.e.

$$Y = E(PU_r, X) \tag{2.3}$$

The ciphertext is transmitted to the receiver. On the receiver's side, a decryption algorithm is used. Upon receiving the plaintext file, the receiver uses its private key, denoted as PR_r , and the ciphertext (i.e. Y) as inputs to the decryption algorithm to yield the plaintext i.e.

$$X = D(PR_r, Y) \tag{2.4}$$

It must be noted that only the receiver's private key can decrypt the plaintext file. This ensures that data confidentiality is preserved.

Asymmetric encryption can also be used to encrypt data using private keys. This scenario is illustrated by figure 2.7(b). The same assumption as above still holds. Instead of encrypting with the receiver's public key, the sender encrypts the plaintext file with its private key. The recipient of the plaintext file keeps a public key directory of all entities it communicates with. The recipient decrypts the plaintext with the sender's public key because only the public key can decrypt the plaintext file. A disadvantage of encrypting using the sender's private key is that it does not provide any confidentiality guarantees as it is susceptible to man in the middle attacks. In fact, the plaintext file can be intercepted and it can be easy for intruder to decrypt the ciphertext since the sender's public key is publicly available.

Both symmetric and asymmetric key cryptography suffer from key management problems. For example, in a symmetric key cryptosystem, the key must be shared between the two communicating entities before they can begin to exchange the secret file. In the case of asymmetric key cryptosystem, having more entities means there must be more public keys involved. The public key directory grows large and becomes difficult to manage. Thus, trusted third parties are used to perform key management and revocation on behalf of the sender and receiver. This also brings challenges on its own such as insider threats and key escrows.

Despite the number of advantages exhibited by traditional cryptographic methods, they are still inadequate for modern technologies, such as cloud computing. In fact, due to the large amounts of data being sent to the cloud, traditional cryptographic methods have provided little to offer strong confidentiality guarantees. In a cloud environment, memory, computational capacity, bandwidth etc., are a big concern. Hence the selection of adequate cryptographic methods must not be resource hungry. As such, traditional cryptographic schemes do not fare well when it comes to multi-tenant, distributed

environments like the cloud. Therefore, new ideas and an improvement on them is needed. The next section discusses concludes this chapter.

2.5 Conclusion

This chapter presented the fundamentals of cloud computing i.e. its service delivery models, deployment models and essential characteristics. From the discussions herein, there are security concerns in a cloud computing environment. To this end, it can be concluded that the cloud is plagued with various security challenges. These challenges hinder the wide adoption of cloud services. For example, lack of data confidentiality guarantees, data leakages, cyber-attacks etc. To preserve data confidentiality, symmetric and asymmetric schemes are used. Symmetric schemes are often used for the actual data encryption while asymmetric schemes are widely used for key exchange protocols. This calls for a hybrid of the two schemes. Key exchange and management thereof needs to be done in a secure manner to avoid man-in-the-middle attacks. Hence strong key exchange and management schemes are needed. Due to security challenges discussed in section 2.6, which is not an exhaustive list, it is not a surprise that most people are still reluctant to adopt the cloud computing paradigm. New and innovative security approaches are required to try and address some of these challenges and ensure strong confidentiality guarantees on client data. This might just give cloud user some confidence and help improve cloud adoption rate.

The next chapter zooms in to discuss existing literature regarding encryption and key management in the cloud. This is to determine existing gaps in the literature in the quest to improve the current state-of-the-art and contribute towards solutions in modern-day cryptography.

This page is intentionally left blank.

Chapter 3 – Literature Review

3.1 Introduction

The previous chapter discussed the background of cloud computing. It also engaged in discussions dealing with cloud computing security challenges and the use of cryptography in the quest to mediate some of the security challenges mentioned. This chapter reviews different ideas as discussed in various related literature aiming to address security challenges in the cloud. The focus is mainly on data confidentiality. The related work covers encryption, key management and secret sharing schemes in the cloud.

3.2 Related work

3.2.1 Encryption

Different researchers have cited homomorphic encryption as the new trend of cloud encryption. Homomorphic encryption exhibits a mathematical property called homomorphism. Homomorphism is a mapping between two structures of the same type, X and Y , in the manner that the first structure, X , is transformed into the second structure Y , using an operator. This is done in a manner that ensures that the properties and relations of X are preserved in Y [62]. Homomorphic cryptography allows third parties to perform certain operations on ciphertext, while also preserving the confidentiality of the data [63]. For example, addition and multiplication are operations that can be done on ciphertexts.

The idea of homomorphic encryption in the cloud was first introduced by [63]. The motivation behind homomorphic encryption is the ability to store encrypted databases on untrusted third party servers while also preserving the confidentiality of the data stored. The computations are done without decrypting the ciphertext. This allows third parties to perform queries of encrypted records in a database such that data is not leaked.

Several homomorphic cryptosystems are used in the cloud. For example, RSA, El Gamal [64] and Paillier [65]. These cryptosystems support either addition or multiplication operations to be done on ciphertexts. However, with the basic RSA scheme, there are restrictions. One of the restriction is that only one operation can be performed, not both at the same time. A study by [66] proposed a homomorphic scheme which allows both operations to be done at the same time. However, this cryptosystem was not fully

homomorphic. To improve on the study by [66], in 2009, [67] proposed the first fully homomorphic encryption (FHE) scheme. The scheme performs any number of addition and multiplication operations. Following the framework presented by [67], two other FHE schemes were introduced by [68][69]. Using these schemes, CSPs are able to perform simple operations such as multiplication and addition on encrypted data. The result of these operations is the same as when the operations are performed on plaintext. Thus, there is no need to decrypt the data to compute such operations. The assumption is that the data owner is the only entity with the secret key used to encrypt the data. For example, the size of the ciphertext might need to be computed to enable the CSP to produce a bill. Again, the cloud user's ID might be needed by the CSP to determine proper back up facilities. However, in doing all these computations, CSPs do not decrypt the entire cloud-hosted data [70] but only part of it. This may also affect the confidentiality of the data if homomorphic encryption is performed by a seasoned cryptanalyst.

A study by [71] proposes a cryptosystem bearing homomorphism as one of its properties. This idea seems to be an ideal solution to address data confidentiality problems in the cloud. However, homomorphic encryption requires high computational resources. Hence, it can be applied mostly on the CSP's side not on the client side. This is mainly because end-user's devices lack the computational capacities that CSPs possess. Another widely used encryption method is Attribute Based Encryption (ABE).

ABE is a public key cryptosystem. It was first introduced by [72] in 2005. Using ABE, the user's private key and ciphertext are associated with a set of attributes or a policy over some attributes. A data owner encrypts data on the client side and shares the encrypted data with multiple authorized cloud users. The other cloud users in the sharing scheme get the decryption key. The decryption key contains attributes. The attributes are used to define access control to the shared data. Decryption only occurs if there is a match between the private key, which entails cloud user's unique attributes, and the corresponding ciphertext [72]. Although ABE is a powerful mechanism, it is not efficient. Performing key revocations using attributes becomes a big challenge in a distributed cloud environment given that each attribute may belong to multiple cloud users [73]. In essence, in an ABE system, attributes are revoked, instead of keys. This makes it difficult because a data owner who is willing to share data must obtain all attributes of other users with whom the data is shared. This task is normally given to a third party entity. However, this also brings threats which might compromise the confidentiality of cloud-hosted data. Involving third party entities might result in insider threats. Quite often, third party entities such as key distribution centers (KDCs) are easy targets of cybercriminals.

A study by [74] proposes an encryption method for data confidentiality guarantees. The encryption scheme implements an inverse of Caesar cipher. The authors suggest that the inverse Caesar cipher is more secure than a pure Caesar cipher [74]. This study cites that this encryption scheme uses American Standard Code for Information Interchange (ASCII) codes whereas Caesar cipher uses only 26 alphabetic characters. Because of this, the original Caesar cipher may easily be defeated with a brute-force attack. Thus, [74] concludes that the inverse Caesar cipher is stronger than the original Caesar cipher. Although the proposed scheme adds ASCII characters, in the quest to increase the key space, it exhibits properties of the original Caesar cipher. However, the proposed technique does not improve that much compared to the original Caesar cipher. This is because it is a variant of the original Caesar cipher. Hence, it is still susceptible to the same attacks as the original Caesar cipher. Due to its simplicity, the technique might not be ideal for securing cloud data.

A study by [75] proposes a playfair cipher due to its ability to encrypt multiple characters. This technique uses a 5 x 5 square matrix. All alphabetical characters are arranged in the matrix, with only one square containing two characters. Using this technique, a user generates a random key (called the keyword), without duplicates and places it in the matrix. Then the remaining alphabetical characters are placed in the matrix. Research by [75] argues that using digraphs instead of monographs makes the playfair cipher stronger than a simple substitution cipher as proposed in [74]. Indeed, this is true. However, the proposed technique may be easily exploited as it is a self-inverse cipher. This means that the digraph in the ciphertext “*AB*” and its reverse “*BA*” have the corresponding plaintext digraphs “*UR*” and “*RU*” respectively. The same argument holds for the plaintext digraphs “*UR*” and “*RU*” as they map to “*AB*” and “*BA*”. This weakness in the playfair cipher makes it vulnerable to known language attacks. Hence, a brute-force attack would not even be required once the plaintext language is known. Therefore, this encryption technique has a gap which might compromise the confidentiality of cloud-hosted data.

Another study by [76] discusses and proposes an architecture which aims to provide data confidentiality in the cloud. This architecture is a client-side encryption scheme. The proposed architecture uses block cipher modes to encrypt data before sending it to the cloud. However, the performance of block ciphers is argued to be slower than stream ciphers [77]. Hence, the proposed architecture suffers from performance degradation. In addition to this, block ciphers work on larger chunks of data (i.e. blocks) whereas stream ciphers encrypt individual bytes. Therefore, they often consume more computational resources, for instance, memory. For a client-side cryptosystem, this may be a disadvantage. For example, most client platforms do not have vast amounts of memory, e.g. mobile phones. Because block ciphers encrypt the

entire data block at a time, they are susceptible to noise in transmission. This means that a change in a data block makes the entire block unrecoverable.

To protect cloud hosted data, some CSPs such as Dropbox [78], Google Drive [79], Mozy [80] and others offer data encryption mechanisms. These CSPs employ Advanced Encryption Standard (AES-192 and AES-256). This is done to give data owners confidentiality guarantees on their cloud-hosted data. On top of providing confidentiality, CSPs want to reduce the amount of data on their storage servers by removing duplicates. This is called data de-duplication. This is generally achieved through encryption techniques that reduce file sizes by compressing the ciphertext after encrypting the data [81][82]. Data de-duplication has been cited to achieve high cost savings for CSPs. For example, it is believed to reduce up to 95% storage needs for back-up applications [83]. Such savings are economical to the CSP and the cloud business.

One of the most widely used schemes is identity based encryption (IBE) which was first introduced by [84]. The reason behind the inception of IBE is to have better management of private and public key pairs without the need for provision of certificates. The assumption that IBE makes is that each cloud entity uses a public key as one of its identifiers. These identifiers uniquely identify the entity, especially in a multi-tenant and distributed cloud environment. The private key is generated by a private key generator (PKG). The PKG computes the private key to bind it to the public key identifier. In addition to this, identity based encryption has emerged as the widely-used scheme in the cloud. It is reinforced by the introduction of the Elliptic Curve Cryptography (ECC). ECC is an enhancement of IBE to strengthen the encryption scheme. ECC differs from Shamir's approach which relies on smart cards mostly for key management and revocation [85].

One problem with IBE is that the private key generator knows all the private keys of clients. Thus it allows global key escrow attacks. Although the key generator might be trustworthy, the fact that the user's keys are known to a third party brings vulnerabilities and opens doors for risks affecting data confidentiality.

The first IBE scheme was proposed in 2001 [86]. This scheme heavily relies on the use of bilinear functions. These bilinear functions bind elliptic curve points to various multiplicative groups. The multiplicative group is used for key generation. Most literature advocates for certificates to be considered as identity based encryption features. The reason cited is that certificates can be used to map the user's

public keys to their identity. While the idea of certificates provides data confidentiality, the use of certificates together with bilinear functions makes IBE cumbersome and less practical.

Furthermore, [87] assesses data confidentiality requirements with regards to VM co-location and data leakages on the cloud. The authors conclude that cloud-based encryption increases computational costs and inefficiency. This study makes an attempt to address data confidentiality challenges through VM encryption. Furthermore, [87] makes a proposal of using encryption at VM level. However, one of the drawbacks with this proposal is that it does not allow multi-tenancy. This is because a shared VM cannot be encrypted, and an encrypted VM cannot be shared [88]. Given of these reasons, VM encryption limits the services offered in the cloud computing environment and is deemed not plausible.

A study by [88] proposes use of a light-weight advanced encryption standard (AES-128), secure hash algorithm (SHA-512), and water-marking to provide confidentiality of cloud-hosted data. In addition to this, the work of [88] makes use of the Bell & Lapadula model to detect confidential data leaks on the cloud. In this study [88], the authors implement a real-time identification of unauthorized entities responsible for data leaks through a cache technique. The technique uses VM cache memory to identify authorized access. The data in cache memory is then encrypted and sent to an authentication server. The server decrypts the message header to find user credentials. The credentials are used to identify legitimate users. If the server cannot decrypt the message, the message header is corrupted and thus that attempted login is classified as malicious. In this manner, cybercriminals can be identified and stopped in their tracks. However, the problem with using virtual cache memory is that virtual memory addresses may not uniquely identify the cached data. This is because virtually indexed caches introduce the potential for aliasing. Aliasing means that a virtual address may be mapped to different physical addresses. Thus, virtual indexing is a one-to-many function. One-to-many functions may not be desirable where unique identifiers are needed, especially in a distributed case.

An analytical study by [89] proposes a neural cryptographic scheme based on mutual learning involving two feed-forward neural networks consisting of discrete and continuous weights. The idea of mutual learning can be used in various aspects of cryptography. For example, it can be used in solving the prevalent key distribution problem using synchronization. Contrary to [89] is the work of [90]; which shows how the solution laid out in [89] can be broken using genetic, probabilistic and geometric attacks.

A neural cryptographic scheme that uses a variant of information substitution is proposed in [90]. Using a recursive, modulo-2 substitution, communicating networks receive an indistinguishable input vector to

produce an output bit. Based on this scheme, a secret key of variable length is generated. This cryptosystem is a substitution cipher. The encryption of plaintext is done by the substitution phase i.e. the recursive modulo-2 substitution to produce ciphertext. This ciphertext is then enciphered to form the final ciphertext. This is done using cipher block chaining (CBC) and an exclusive OR (i.e. *XOR*) operation. The CBC and *XOR* operations are performed on identical weight vectors with identical intermediate ciphertext block lengths. Integrating CBC with neural cryptography produces strong ciphers. These ciphers employ machine learning algorithms to improve the efficiency of ciphers. This study adopts these concepts in implementing a strong cipher, especially the concept of machine learning.

Some studies call for regulations demanding CSPs to design “back-doors” on cryptosystems [91]. This is done to grant law enforcement agencies uninterrupted access to intercept client data [91]. Such “back-doors” weaken the integrity of the cryptosystems. This compromises the confidentiality, integrity and availability (CIA) of client data. For example, the widely-reported FBI vs Apple case has set the bar for future cryptosystems. This case bears significant ramifications on the future of cryptography and how best it can be used to provide strong encryption. Testimony to this is the quick move made by WhatsApp™ to provide end-to-end encryption [92][93][94]. This dissertation aims to address the problem identified in [91], which is aligned to the problem statement of this research, to advocate for strong client-side encryption. The next section discusses key management issues which come with the use of encryption.

3.2.2 Key management

Encryption goes together with key management. This section discusses key generation and management techniques used in the cloud. Cloud key management systems utilize identities of cloud users for identification and authorization to access decryption keys. Once users have been identified and authorized to access cloud-hosted data and have access to decryption keys, proper key management and revocation of these keys becomes a priority. The discussions herein focus on identity based key management schemes. This is due to its generic nature which makes it adoptable for deployment in a single or multi-user environment such as the cloud. Because such key management schemes are generic, they fit any situation. It should be noted that asymmetric key cryptography is mostly used for key management. In this case, user’s private keys are regarded as unique identifiers. This is because private keys are known to the data owner only and must be kept secret. Hence, they are used as digital signatures. Digital signatures are a mechanism for non-repudiation (i.e. non-denial). Therefore, identity based encryption key managers are ideal for cloud computing environments.

To derive a user's private key, a private key generator (PKG) is used to define a set of elements. This set is known as the set of public elements (PE) [84]. The PKG then generates three groups G_1 , G_2 , and G_3 . A function over the finite group set, $f(G_1, G_2)$, is defined as the dot product of G_1 and G_2 . The groups must be additive subgroups, of the same order q , of an Elliptic Curve (EC) with G_3 as the multiplicative subgroup of a finite field [86][95]. This key generation is used for both symmetric and asymmetric schemes, with the following conditions [85]:

- $G_1 = G_2$ for symmetric schemes.
- $G_1 \neq G_2$ for asymmetric schemes.

Once the groups have been specified, the PKG defines hash functions according to the protocols found in [101]. In general, the public key of a client is computed as a message digest of the hash function, with input to the function set as one his identities. The public key will either be a point on the Elliptic curve or a positive integer [86][97]. For example, the function could be specified as the following equation:

$$Pub_{id} = Hash(id) \tag{3.1}$$

Consequently, a private key can be generated once the public key has been computed. The PKG then securely transfers the private key to the owner. The transmission is secured with cryptographic methods, using encryption or a secure transmission device. However, this scheme suffers from numerous drawbacks such as the key escrow attacks. Key escrow is a key exchange process involving third parties. A key escrow attack is an interception of the key exchange protocol with the intention to gain unauthorized access to the decryption keys. However, the PKG can also launch a key escrow attack. This is because the PKG is able to impersonate any of the key owners since it generates private keys for them. Consequently, PKGs have the knowledge of the data owner's private keys. Thus, the PKG can decrypt any data. This compromises the confidentiality of the data being outsourced to the cloud. The next section discusses some already existing key management solutions which can help in solving the problem.

3.2.2.1. Key management solutions

The application of identity based encryption schemes in the cloud was practically explored by [98] back in 2004. The proposition made therein is that each virtual organization should have its own private key generator. With this proposal added to the scheme, more flexibility is offered during key generation. However, since this scheme still includes the untrusted PKG, data confidentiality cannot be guaranteed.

Then, in 2005, [99] introduced a new approach which employs dynamic key infrastructure for the cloud. This is made to simplify key management issues affecting the scheme proposed in [98]. The approach is hybrid in nature. It combines identity based encryption at the client level together with traditional public key infrastructure (PKI). This is done in order to support key management and revocation. This variant of the identity based encryption avoids the need to have a proxy certificate authority, which could compromise the confidentiality of client data. It also reduces the processing power and time required for the private key distribution. Unfortunately, users have to verify the parameters of each tenant they wish to share data with. In doing so, this creates vulnerabilities that could be exploited by threats such as the man in the middle attacks for eavesdropping.

Moreover, in 2006, [100] proposed a key-policy attribute base encryption scheme which is built on the foundation of identity based encryption. In this scheme, ciphertexts are labelled with a set of attributes defining each cloud user. These attributes are associated with access structures controlling which cloud user can decrypt the shared data. This scheme ensures flexibility and fine-grained access control while also providing data confidentiality guarantees. It eliminates the increasing reliance on the storage server to prevent unauthorised access to confidential data.

To address the confidentiality drawbacks mentioned earlier, [102] proposes a key life-cycle management system (KLMS). InterCloud adopted the KLMS for its key management. Strongbox file is a key management system used by Cryptonite [103] in Microsoft Azure web. Strongbox file broadcasts encryption keys to authorized entities. These entities are recorded in an access control list (ACL). The ACL is used to manage user's access control. This is done in order to provide confidentiality of cloud-hosted data [103].

In a nutshell, encryption key management systems face key distribution problems, especially in a multi-tenant environment. Existing systems have cumbersome processes to manage the keys. These processes often make the KMS less efficient. Efficiency in a cloud environment is a necessity. Often, third parties are trusted with the responsibility to manage encryption keys. However, this opens vulnerabilities which could be exploited, leading to unauthorized data access. Key escrow attacks are also increasing due to trusting third party entities. This opens many vulnerabilities which could result into security risks and confidential data leaks as key generators may have access to the decryption keys.

The next section discusses mechanisms of distributing encryption keys in a secure manner. The focus is placed on secret sharing methods employed in the cloud to strengthen data confidentiality.

3.2.3 Secret sharing

Modern day cloud security is increasingly reliant on how users manage and protect their encryption keys. For example, a threshold cryptosystem decomposes a block of secret data into multiple shares to be distributed to authorized participants. The original secret may be recovered by the authorized group of participants. To reconstruct a secret, qualified participants need to combine their shares, or a third party, called “combiner” or “dealer” can reconstruct the secret on their behalf after receiving their individual shares.

In the quest to provide data confidentiality, multiple cloud systems such as DepSky [101] and InterCloud [102] were introduced. These systems do not only use symmetric key encryption but also broadcasts encrypted data over multiple cloud storages. This way, each cloud provides a high level of data confidentiality on the share it stores. DepSky encrypts data with a random secret key, and then splits the ciphertext is split into shares which are distributed over four cloud storages. Each cloud stores a block of data share along with a key share.

CloudProof [104] is another system which tries to make key management an easy task. The system uses two separate blocks, one is the data block and the other is the key block. To distribute keys to the authorized entities in the ACL, the system uses key rotation and broadcast encryption coined by [105]. With CloudProof, authorized users can read and write data blocks depending on their access rights.

CloudSeal is a cryptosystem which integrates symmetric encryption and secret sharing schemes to manage access control and provide confidentiality on the cloud [106]. CloudSeal provides forward and backward security. Forward security ensures that no entity can access the data without being part of the authorized group. Backward security ensures that no entity has access to the data after leaving the authorized group. The next section discusses some of the secret sharing methods used to implement secret sharing schemes.

3.2.3.1. Secret sharing methods

Secret sharing schemes have been implemented using several methods [107]. This section discusses the three most relevant secret sharing methods which are often regarded as essentials for the required secrecy in a secret sharing scheme.

- Perfect Secret Sharing (PSS):

This method's perfection is measured using the following two parameters:

- If, given $t-1$ shares, where t is the threshold value, no information regarding the secret can be revealed.
- The ratio of the length of the secret to the length of each share is equivalent to 1. This is called the information rate.

➤ Information dispersal algorithm (IDA):

Using this method, a secret message S is split into n pieces such that an entity can only obtain the secret if $d < n$ pieces are available, where d is the threshold value. This concept can greatly reduce the memory needed to store the shares. However, it has the tendency of revealing the secret as the shares exhibit some patterns of the secret message frequently. Hence, if more shares can be intercepted and compromised, the secret might be revealed.

➤ Krawczyk's computational secret sharing

This method was proposed by [108]. The method combines the first two discussed above. Data is encrypted with a randomly generated symmetric key using a symmetric encryption system. Then the secret data is split into n fragments using the information dispersal concept, with a threshold value of d . This concept combines the strengths of perfect secret sharing and information dispersal.

Secret sharing provides a secure manner in which confidential data can be outsourced to the cloud. The idea of applying secret shares gives cloud users confidentiality guarantees. For a secret sharing scheme to be compromised, all shares must be compromised. This property of secret sharing schemes makes it computationally infeasible for intruders to launch data breaches.

3.3 Conclusion

Numerous researchers have suggested several mechanisms of protecting cloud-hosted data using encryption and key management techniques. Existing literature cites encryption as the widely used data protection method in the cloud. However, it is also concluded that existing encryption techniques are insufficient to provide confidentiality guarantees. This is more evident when considering cloud environments. To this end, it can be seen that encryption and key management still have a lot of gaps to be filled to offer strong cryptography and consequently confidentiality guarantees. Regarding encryption, the confidentiality of an encrypted message is concluded to rely on the secrecy of the key. This is because

revealing the encryption key to other entities or third parties compromises the confidentiality of the encrypted data. Most of the existing and reviewed literature makes use of third party key distributions centers (KDCs) or certificate authorities (CAs) for key management and revocations. This idea requires a high level of trust which, given the nature of the cloud environment, cannot be guaranteed. As such, these gaps related to encryption and key management should be addressed to preserve the confidentiality of cloud-hosted data. For example, current encryption standards put emphasis on the key length as exhibiting the strength of the cryptosystem. The argument is that the longer the key, the stronger the cryptosystem. However, even with the longest keys, cryptosystems such as AES-256 are still insufficient for cloud deployments. Cryptosystems are widely used but data breaches are reportedly increasing. This shows that the strength of the cryptosystem might not necessarily rely on the length of the encryption key.

Furthermore, encryption key management is a big issue. In a multi-tenant and distributive cloud environment, encryption key management becomes a crucial aspect. This is attested by the cloud security alliance (CSA) reports [109]. Existing key management systems rely on the issue of trust. Third party entities are trusted with generating, distributing and managing keys. This opens vulnerabilities which may be exploited by cybercriminals. Key distribution centers are often targeted by cybercriminals with the hope of gaining unauthorized access to user's encryption keys. Moreover, insider threats are also a big concern as the data owners have no capacity to perform key management. Data sharing becomes another big problem as decryption keys need to be transferred to the intended recipients. This adds more concerns to the existing key distribution and management problem.

The next chapter discusses a proposed model which aims to address the gaps identified in the current literature. As such, in this dissertation, a light-weight, strong, client-side encryption scheme is proposed. The stance taken by this dissertation is that the proposed cryptosystem tackles the prevalent key management issues and improve resource consumption (i.e. processing power and memory). This may be beneficial for deploying the proposed system on devices with low memory specifications.

This page is intentionally left blank.

Chapter 4 – Proposed Model

4.1 Introduction

The previous chapter reviewed existing literature and identified gaps. This chapter starts by outlining functional system requirements learned from the gaps identified in the previous chapter before it presents the proposed model. The following have been identified as key functional requirements for the proposed model, which forms the foundation for the proposed cryptosystem presented in the next chapter.

4.1.1 System functional requirements

To meet the objectives and fill the gaps identified in the review of existing literature, the proposed cryptosystem has the following functional requirements. These requirements form the basis on which the entire software development life cycle (SDLC) of the proposed system is developed.

- Client-side – The system must be able to encrypt cloud user’s data before it could be uploaded on the cloud. The encryption keys and everything concerning the model should be done on the client-side. The CSP should not have access to the encryption algorithm or keys.
- Light-weight – The system must be able to encrypt and decrypt data on the fly and without requiring a lot of computational resources. This is to enable our proposed solution to be suitable for devices with minimal computing resources like mobile devices.
- Secure key management – The system must be able to manage encryption keys securely and effectively. This also has to be done on the client side.
- Secure data sharing – The encrypted data must be shared with other cloud users in a secure manner.
- Secure encryption key sharing – Encryption keys must be shared securely using a secret sharing scheme that splits the key into shares. The shares are to be distributed to participating cloud users for reconstruction to enable decryption.
- Self-destruction capabilities – The system must be designed such that it can detect an attack to the encryption keys and be able to destroy encryption keys automatically once the shared key timestamp’s time to live (TTL) reaches a specific threshold.

Each of these requirements is taken into consideration in the design of the proposed model. The proposed model is presented in the next section.

4.2 Proposed model - CloudCrypt

This section presents the proposed model for this study. The model is designed based on the system requirements above. Figure 4.1 below depicts the proposed model, which we call CloudCrypt.

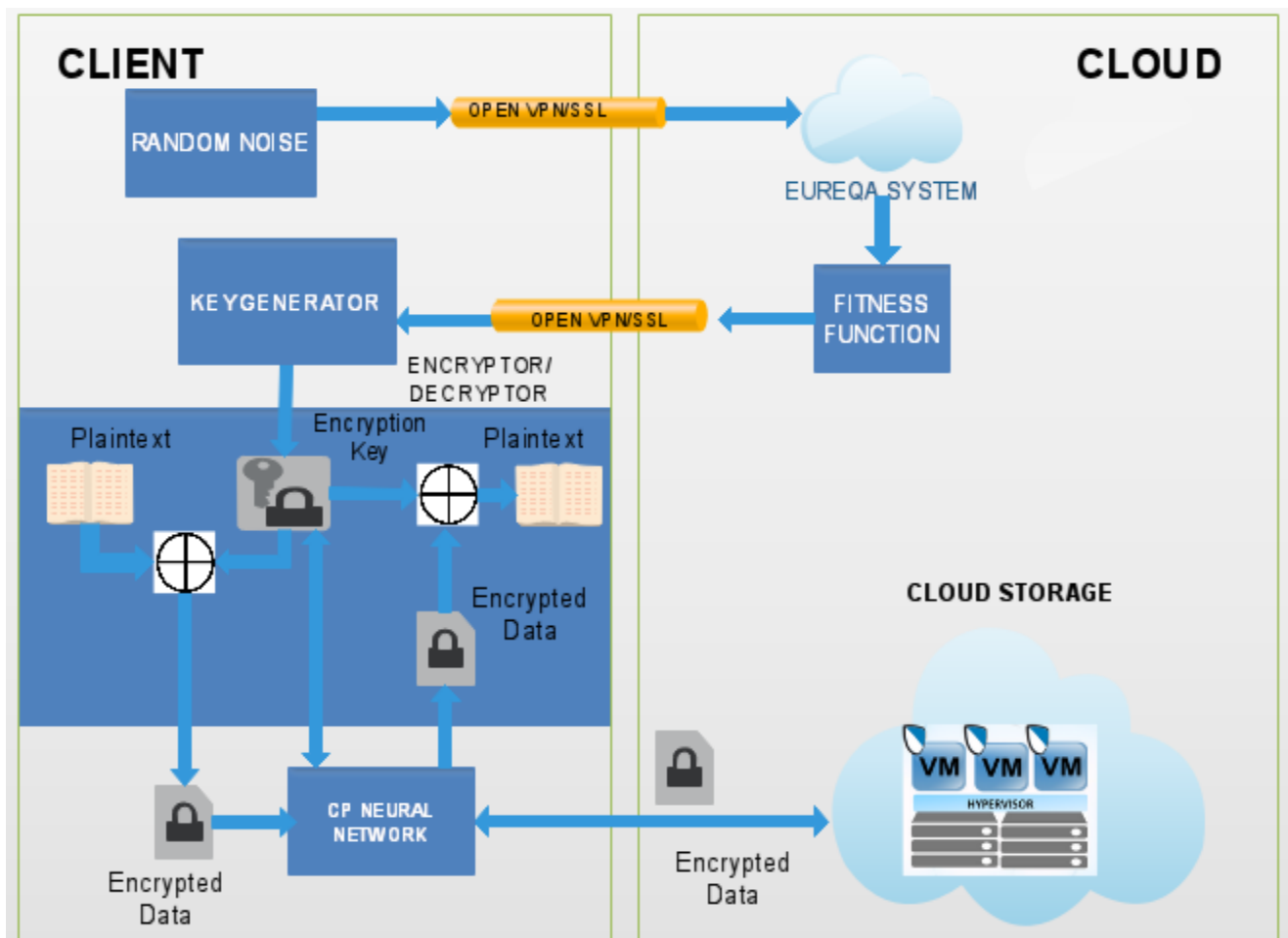


Figure 4.1: CloudCrypt – An Innovative Crypto Scheme [2]

Figure 4.1 above depicts a model for CloudCrypt – the proposed crypto scheme. The scheme is based on symmetric key encryption. This means that the same key is used to encrypt and decrypt. Furthermore, CloudCrypt is based on evolutionary computing concepts. These concepts introduce a paradigm shift in terms of conventional approaches to encrypting data associated with the Kerckhoff-Shannon principle. The concepts adhere to static algorithms and dynamic encryption keys. Furthermore, evolutionary computing concepts explore how encryption algorithms and keys can quite literally be generated ‘on the fly’. This is done so that a user can be provided with, or, better still, individually generate a personalised encryption algorithm in addition to a personalized encryption key [110].

The proposed CloudCrypt model dictates the steps to be taken in order to encrypt user data before sending it to the cloud service provider. This is initiated by generating random input noise. The noise is then transmitted through a secure channel using TLS into a Eureka system [112]. The Eureka system takes the input noise and produces a non-linear fitness function which is the best approximation of the input noise. The fitness function is then used to generate a symmetric key to encrypt cloud-bound data. The resultant key exhibits the random and chaotic properties of the input noise. Hence, the strength of the symmetric keys is derived from the chaotic and random nature of the input noise.

It is on this premise that our system's key strength does not lie on the length of the key, but on the quality of chaotic and random input noise. Previous research puts emphasis on the length of the key as exhibiting its strength [113]. The argument is that the longer the key, the stronger the cryptosystem. For example, a 256 bit AES key is argued to be stronger than a 192 bit AES key. The authors of the proposed solution do not argue against this international standard, but seeks to bring a new dimension based on chaos. Even though chaos provides a promising approach for cryptography, it cannot be argued to guarantee full-proof security. For example, no matter how strong or well designed a cryptosystem might be, shorter keys from the most powerful cryptosystems cannot survive brute-force attacks using maximum possible computational power and given enough time at the hands of the attacker [114][115]. Hence, the proposed solution does not claim to provide perfect security. However, using CloudCrypt, even if an attacker can brute-force a key, it does not mean that he or she would have all other keys generated from this system. A brute-force attack could only give an attacker just the brute-forced key and nothing more. The next section discusses how CloudCrypt generates encryption keys in the key generation module.

4.2.1 Key generation

To begin the encryption process, a symmetric key must be generated. The key is generated using a fitness function. This fitness function is the output of the Eureka system given a random input noise. The fitness function is seeded with n random floating point numbers. These numbers are normalized in the range [0, 1]. This process generates a random encryption key. This encryption key is converted into a binary stream. Once the key is successfully generated, it is sent into the encryptor as input so that the encryption may proceed. The next sub-section discusses the encryption process.

4.2.2. Encryption

To perform the encryption process, plaintext is converted into a binary string. This binary string is sent into the encryptor as input. The binary key and plaintext strings are compared in terms of length in the

encryptor. For encryption to proceed, the length of the plaintext must be equal to the length of the key. If the plaintext binary stream is longer than the key stream, a new key is generated from scratch i.e. using a new set of floats and a different random set. This idea of re-generation is meant to strengthen the key. This key is chosen randomly from a set of keys. Otherwise if the plaintext binary stream is shorter, the key stream is trimmed and the extra key bits are discarded. When the two binary streams are of equal lengths, a bit-wise exclusive OR (*XOR*) operation is performed to produce ciphertext.

The ciphertext is used as input into a counter propagation neural network (CPNN) together with the encryption key. During the training process, input vectors X and Y are used to make the neural network to detect and “remember” the encryption key. This is to ensure that once the key has been created, used to encrypt the plaintext and learned by CPNN, it can be destroyed to prevent it from being compromised. The next section discusses the CPNN. The encryption process can be summarized using the following algorithm.

Algorithm 1: Encryption

```

Result: cipher
P ← Plaintext
noise ← randomNoise
fitnessFunction ← generatedFunction
floats ← normalize with floats between [0, 1]
Encryptionkey ← random key  $k_i$ 
if  $k_i.length \leq P.length$  then
    Encryptionkey ←  $k_i + \text{random key } k_j$ 
else
    if  $k_i.length \geq P.length$  then
        trim( $k_i, P.length$ )
        encrypt( $P, k_i$ )
        cipher ←  $P \text{ XOR } k_i$ 
        CPNN(cipher,  $k_i$ )
    end if
end if

```

Figure 4.2: Encryption algorithm

4.2.3 The Counter Propagation Neural Network (CPNN)

A CPNN is a hybrid of other neural networks. It consists of a combination of a structure widely known as a competitive network [116]. The CPNN is a variant of a neural network consisting of three layers: the input, Kohonen (i.e. hidden) and Grossberg (i.e. output) layers. The structure of a CPNN is depicted in the figure below:

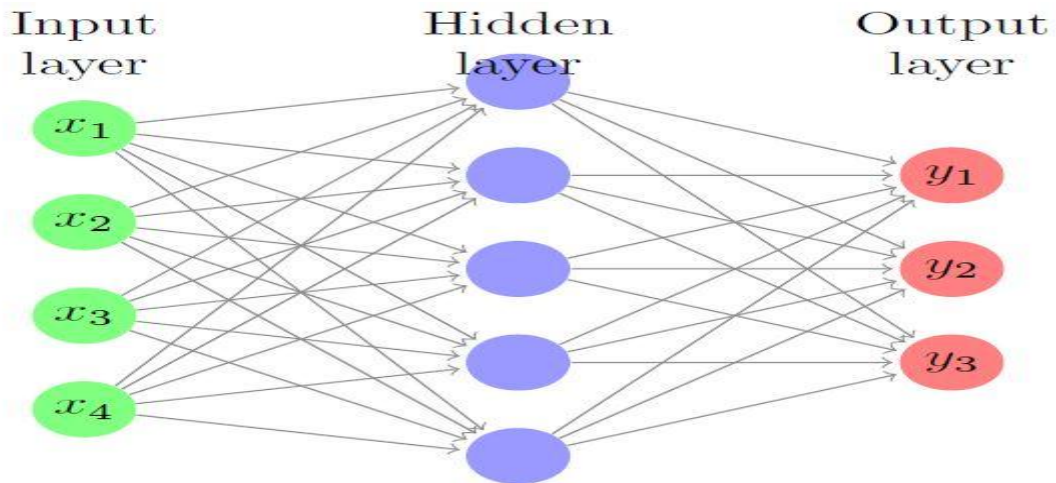


Figure 4.3: Structure of a CPNN

There are two important aspects of a CPNN. These are supervised and unsupervised learning. The Grossberg layer uses supervised learning while the Kohonen layer uses unsupervised learning. Depending on the learning scheme used, the weights of the input vectors are automatically changed [117]. A CPNN can briefly be described to function as follows:

Given a set of vector pairs $(X_1, Y_1), (X_2, Y_2), \dots, (X_m, Y_m)$, a CPNN is able to learn and associate an X vector given in the input layer with a Y vector in the output layer. If there is a continuous function, f , relating X and Y vectors such that the following equation is satisfied:

$$y = f(x) \tag{4.1}$$

The CPNN will learn how to approximate the mappings for any value of x in the range. The range is specified by a set of training vectors. In addition to this, if f^{-1} (i.e. inverse of f) exists, the CPNN can also learn the inverse mapping to get the value of x , such that the following inverse function is solved:

$$x = f^{-1}(y) \quad (4.2)$$

This makes the CPNN a powerful tool. Hence, this study uses it to perform pattern matching using a backward or reverse sweep.

The mathematical model of the network works as a perceptron. The perceptron computes an output value given a certain input value. A criterion function is used for learning and correcting errors. The criterion function is also used to minimize the output error. This is defined as the deviation from the required output [118]. The criterion equation is given below.

$$E_k = d - y \quad (4.3)$$

where d is the desired output, and y actual output from the network. E_k is the error of the k th output neuron. The aim of equation 4.3 above is to minimize the output error as much as possible. When the error value is small, the network responds more precisely to the training data. Finding the minimum error value requires weight adjustment. Adjusting weights affects the final error value for the entire network. This results to a winner node. A winner node is the one with the least error value. Unlike other neural networks such as the back-propagation networks, the CPNN avoids propagation of error values to previous neurons in the hidden layers. This gives the CPNN an added advantage over its counter-parts and makes it more efficient [119]. The following are some of the advantages of CPNNs:

- They combine the advantages of other neural networks since the CPNN is a hybrid network.
- They use both supervised and unsupervised learning schemes, instead of just one scheme.
- The learning rate is higher compared to other neural networks, hence the CPNN is faster.
- CPNNs remain an excellent choice for rapid prototyping [120][121].

The next section discusses how a CPNN is trained using learning algorithms to learn the encryption key in this study.

4.2.4 Key learning

A neural network is used for key learning and management. The process of key learning takes place in the CPNN. Once the encryption process is completed, the resultant ciphertext and key are sent into the CPNN. Supervised and unsupervised learning schemes are used to aid the learning process. The CPNN has two important functions which manipulate binary data stream [121]. However, because of computational purposes when computing weights, the input binary data streams (i.e. the ciphertext and the key) are converted to decimal equivalents before processing in the CPNN. The learning process uses both supervised and unsupervised learning algorithms. The discussion below deals with each learning algorithm independently.

4.2.4.1 Unsupervised learning algorithm

The unsupervised learning process can be summarized using the following mathematical equations: Let X be the input training vector, Y be the target output vector and W the weight vector. Then the following procedure is followed.

- Compute the Euclidean distance between the input vector and the weight vector in each hidden node, using the equation:

$$\|X - W\| = \sqrt{(X - W) \cdot (X - W)} \quad (4.4)$$

- Get the node with the shortest distance (winner node).
- Modify the weights of all nodes connected to the winner node in the hidden layer using the following equations:

- $W_{ij} = W_{ij}(\text{old}) + \alpha (X_i - W_{ij}(\text{old}))$

$$(4.5)$$

- $V_{kj} = V_{kj}(\text{old}) + \beta (Y_k - V_{kj}(\text{old}))$

$$(4.6)$$

where:

- a) X is the input training vector.
- b) Y is the target output vector.
- c) W_{ij} is the weight from the X (training) input unit.
- d) V_{kj} is the weight from the Y (target) input unit.
- e) α and β are random integers.
- f) $i = [1, n], j = [1, m]$

The following pseudo-code summarizes the unsupervised learning algorithm.

```
1   Algorithm 2: unsupervised learning
2   Begin
3   randomly select an input vector
4   normalize the selected input with random numbers in the range [0, 1]
5   compute the weights : calculate dotproducts of inputs
6   adjust the weights and correct errors
7   repeat until all nodes are processed
8   classification of winner : get node with least error margin
9   Assign winner : the value 1
10  Assign all nodes : the value 0
11  End
```

Figure 4.4: Unsupervised Learning

The next section discusses the supervised learning procedure which occurs in the Grossberg's layer of the CPNN.

4.2.4.2 Supervised learning

The supervised learning process in the Grossberg layer proceeds as described below.

- Compute the Euclidean distances between input vector and the weights of every hidden node.
- Find the node with the shortest distance (the winner node).
- Set the link from the winner node to the output node as 1, the rest must be 0.
- Adjust the weights with the following equations:

$$U_{jk}(new) = U_{jk}(old) + \alpha (Y_j - U_{jk}(old))$$

(4.7)

$$V_{ji}(new) = V_{ji}(old) + \beta (X_i - V_{ji}(old))$$

(4.8)

Where:

- a) U_{jk} is the weight from the cluster layer unit to the Y output layer.

- b) V_{ji} is the weight from the cluster layer unit to the X input layer.
- c) α and β are random integers.
- d) $K = [1, m], I = [1, n]$

Once the key has been processed by the CPNN, it can be destroyed. This is done in order to avoid storing it. This is essential these days because most attackers now target key stores in to compromise systems. CloudCrypt ensures that even if an attacker could get data from the cloud, even if they could get access to the target client's systems, they could not get any key store where all keys are kept.

Once the key has been destroyed, the encrypted data may now be sent to the cloud for storage. The next section discusses the decryption process.

4.2.5 Decryption

Decryption can only be done by the data owner. This is done in such a manner that the CSP or any unauthorized third party entities cannot decrypt the data. This provides the user with absolute assurance of the confidentiality guarantees of their cloud-hosted data. Once the ciphertext has been downloaded from the cloud, it is sent into Grossberg layer (whose target value was set as the ciphertext during the encryption) using supervised learning.

The output of the Grossberg layer is compared to the target value by computing the Euclidean distances as discussed earlier. If the target value matches the ciphertext, the corresponding key (which was used as input to the Kohonen layer during encryption) is produced. The key value is converted back to binary form. The plaintext is recovered by performing a bit-wise *XOR* operation of the ciphertext and the key retained by the CPNN.

The data owner does not need to remember key nor store it. The CPNN acts as a key storage. Based on the patterns of the ciphertext, the CPNN determines the key which was used to encrypt the data. The key is set as the target in the Grossberg layer. To make certain that the correct key is retained; a hash of the key is produced. This is to ensure data integrity. This also ensures that no key and ciphertext can produce the same hash values. Hence, the process will always produce a unique value. The algorithm below summarizes the decryption algorithm.

Algorithm 3: Decryption

Result: Plaintext

Begin

$C \leftarrow \text{download}(\text{cipher}, \text{tenant}_{id})$

$CPNN(C)$

$\text{target} \leftarrow \text{Grossberg}()$

$\text{key} \leftarrow \text{unsupervisedLearning}(C)$

if $\text{key} \neq \text{target}$ **then**

$\text{halt}()$

else

$K \leftarrow \text{convertToBinary}(\text{target})$

end if

$\text{output}(K)$

$P \leftarrow C \text{ XOR } K$

End

Figure 4.5: Decryption algorithm

The next section discusses data sharing.

4.3. Data sharing

Cloud storages have various attractive benefits. These benefits include data sharing. However, security and key management in a shared environment remains a big concern [109]. In a multi-tenant cloud environment, users may want to share their data. If the data is shared among multiple cloud users, the secret sharing scheme is used to authenticate and authorize other users to enable successful decryption. Otherwise, the decryption process will not proceed if the keys are not shared. For example, because the data is stored in an encrypted format, users that are willing to share their data need to share the decryption key. This means a key exchange between two or more cloud users needs to occur. Otherwise, the recipient of the data will not be able to decrypt it. Key exchange is a problem in current cryptosystems [109][122]. Cloud users rely on CSPs and third party entities to provide them with secure and confidential key exchange mechanisms [123].

Exchanging an entire key between two or more communicating entities opens it to the risk of cybercriminals. Criminals could intercept the key exchange and performing a brute-force attack to deduce the key [124][125]. To address such issues, CloudCrypt implements a secret sharing scheme applied

directly on the encryption key. The encryption key is sent to intended recipients as key shares instead of the entire key. This makes it hard for cybercriminals to deduce the key even if a key share gets intercepted on its way to one of the recipients in the scheme. In existing systems, such as DES, if an attacker compromises the key exchange protocol to get hold of the key, then the attacker can get hold of the encrypted file and decrypt it. However, using the secret sharing scheme utilized by CloudCrypt, the attacker cannot get any information related to the encrypted data unless all key shares have been compromised. Furthermore, the attacker would have to know the order of the shares to be able integrate them into a single key. The secret sharing scheme that CloudCrypt uses is described in the next section.

4.3.1 Secret sharing scheme

Secret sharing schemes are ideal for manipulation of highly sensitive information. For example: encryption keys, bank accounts among others. Each share generated becomes confidential information [126]. The secret sharing scheme that is used in CloudCrypt is based on Shamir's secret sharing algorithm [27]. The scheme relies on the idea that it takes at least two points to define a straight line. Three points are needed to define a quadratic, four points to define a cubic curve, and d points to define a polynomial of degree $d-1$. Hence a polynomial of degree $d-1$ is defined over a finite field of integers, Z_p to implement d -out-of- n secret sharing scheme. Where n is the number of participating entities in the scheme and p is any random prime number. The first coefficient of the polynomial is set as the secret itself [127] as shown below and the rest of the coefficients are randomly picked.

$$f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_{d-1}x^{d-1} \quad (4.9)$$

Where a_0 is the secret. Consequently, n points are constructed on the curve and one point of the form (x_i, y_i) is given to each participant in the scheme. When at least d out of n participants reveal their points, there will be sufficient information to fit a $(d-1)$ th degree polynomial, where the first coefficient is the secret through Lagrange interpolation [128]. The following diagram illustrates a secret sharing scheme which CloudCrypt uses.

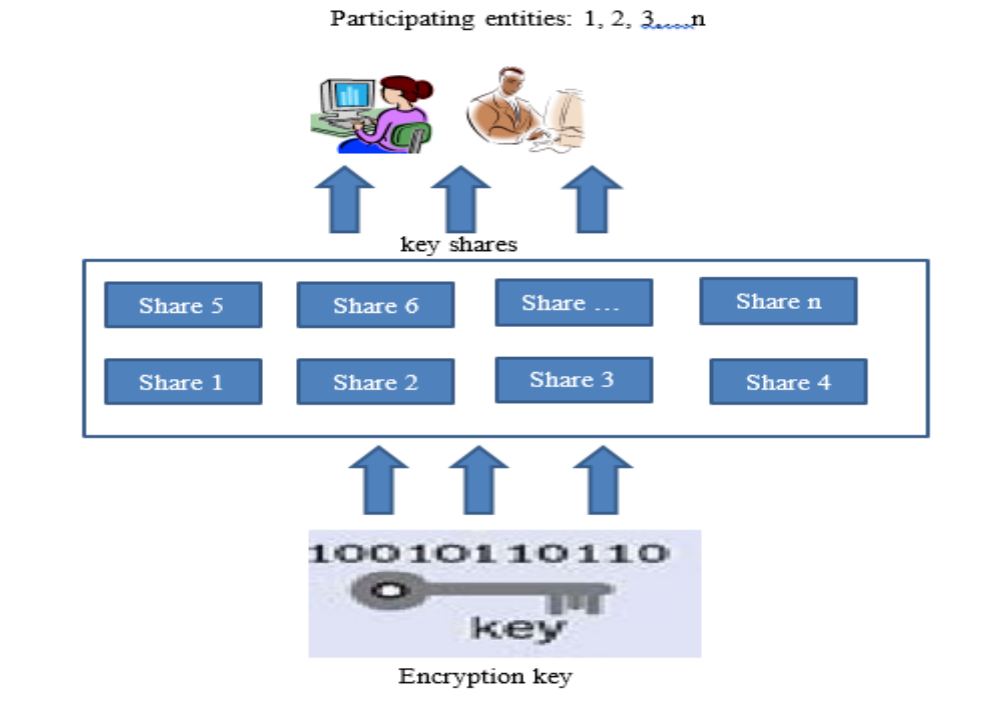


Figure 4.6: CloudCrypt's secret sharing scheme

The secret sharing scheme is part of the key management system implemented using a CPNN, discussed earlier. However, it is invoked only when there is a desire for data sharing. This adds to the strength of CloudCrypt in providing data confidentiality guarantees. Secret sharing schemes are important in cloud computing environments. Hence, a key may be distributed to participating cloud tenants in the scheme. Other authentication methods are needed to verify the identity of participating tenants. However, authentication is outside the scope of this study. The key is then reconstructed when needed, for decryption purposes. The scheme uses a threshold value of 3 (i.e. $t = 3$). This is to ensure that any group of t or more cloud users can together reconstruct the key to enable decryption. No fewer than t users may reconstruct the key. However, the threshold value is variable, with a minimum value of 3. Otherwise the decryption is not possible. This concept is called the perfect secret sharing (PSS) [107].

The secret sharing scheme can be summarized using the following steps.

1. An encryption is obtained from the CPNN
2. The key is split in n secret shares $S_i, i = 1, 2, 3, \dots, n$
3. Determine the number of participating entities in the secret scheme
4. Determine the threshold value
5. For each share, compute the hash of the secret share

6. Send each share together with its message digest (i.e. hash) to the intended participant

The next section discusses the self-destruction capabilities of CloudCrypt. This mechanism is used by the secret sharing scheme to autonomously add another layer of security in the scheme.

4.3.2 Key share self-destruction

The self-destruction capability adds another layer of security in cases where the key shares might be intercepted by a malicious attacker. Given that data is sent to the cloud in encrypted format, data confidentiality is preserved. The self-destruction of key shares reduces the risk of eavesdropping. It allows the key shares to exist within a specified period. The key shares self-destruct when the time to live elapses. Given the nature of the cloud, the assumption made when implementing the self-destructor is that the longer the keys are in the cloud space, the more vulnerable they become to interceptions by eavesdroppers.

Once the data owner decides to share their cloud hosted data which is encrypted using CloudCrypt, a specified time-stamp is created. The time-stamp dictates the amount of time set for the decryption to be possible. If the time-stamp period elapses and the key shares have not been reconstructed to generate the full decryption key, all the key shares are destructed automatically. Corresponding key shares have the same time stamp value. This concept is widely used in secret sharing schemes such as SeDas [129]. The destruction of key shares is irreversible. Thus, the data owner will need to re-generate new key shares and distribute them to participating cloud users accordingly. The next section concludes this chapter.

4.4 Conclusion

Adoption of cloud services is hampered by concerns over data confidentiality. In the quest to make cloud services secure, this chapter presents and discusses client-side cryptography. It shows how client-side cryptography can be used to secure data before it is uploaded to a public cloud infrastructure. The proposed solution meets all the system requirements and gives cloud consumers full control over their cloud-hosted data. One of the most difficult processes in a cloud computing environment is key management. To address the prevalent key management problem, this chapter proposes the use of a counter propagation neural network (CPNN) which can be trained on the patterns of the ciphertext and the corresponding key combination. Successful implementation of this cryptosystem enables users to avoid storing keys and the key management problem thereof.

The CPNN uses machine learning processes to perform pattern recognition on the symmetric key. Once the symmetric key has been learned, it is discarded. The data owner may send data to the cloud in an encrypted format. Consequently, to enable successful data sharing, this chapter proposes a secret sharing scheme based on Shamir's secret sharing algorithm [27]. To differentiate this scheme from that of [27], a self-destruction mechanism is implemented. This mechanism allows decryption to happen within a specified period, called the time-to-live (TTL). Should the TTL elapse and the key shares have not yet been constructed into a decryption key, the decryption process halts and the key shares self-destruct. Thus, CloudCrypt ensures strong confidentiality guarantees for cloud-bound data to be stored in cloud servers. The next chapter discusses experimentations of the proposed model for CloudCrypt laid out in this chapter together with the results obtained.

This page is intentionally left blank.

Chapter 5

Implementation and Evaluation of CloudCrypt

5.1. Introduction

The previous chapter introduced CloudCrypt – a client-side cryptographic system. This chapter presents implementation details of the proposed system followed by the experimental results and evaluation. Furthermore, this chapter evaluates the proposed CloudCrypt system against some of the existing cryptosystems. These include Data Encryption Standard (DES), triple DES (3-DES), Advanced Encryption Standards (AES 128, 192, 256). The next section discusses the system implementation of the proposed solution.

5.2. Implementation

CloudCrypt’s graphical user interface (GUI) is developed using PHP. It uses the Java Security Library (JSL), java BASE64 library, java extension mail library (i.e. javax.mail), java system monitoring library (i.e. java sysmon), java pairing-based cryptography (i.e. jPBC) library, java MXBean library, Java weka library [130], and MySQL for data storage. The system is implemented on the Netbeans Integrated Development (IDE) version 8. The libraries are used as java archive (jar) files and imported into the project. The following screenshot shows the specification of the computer used.



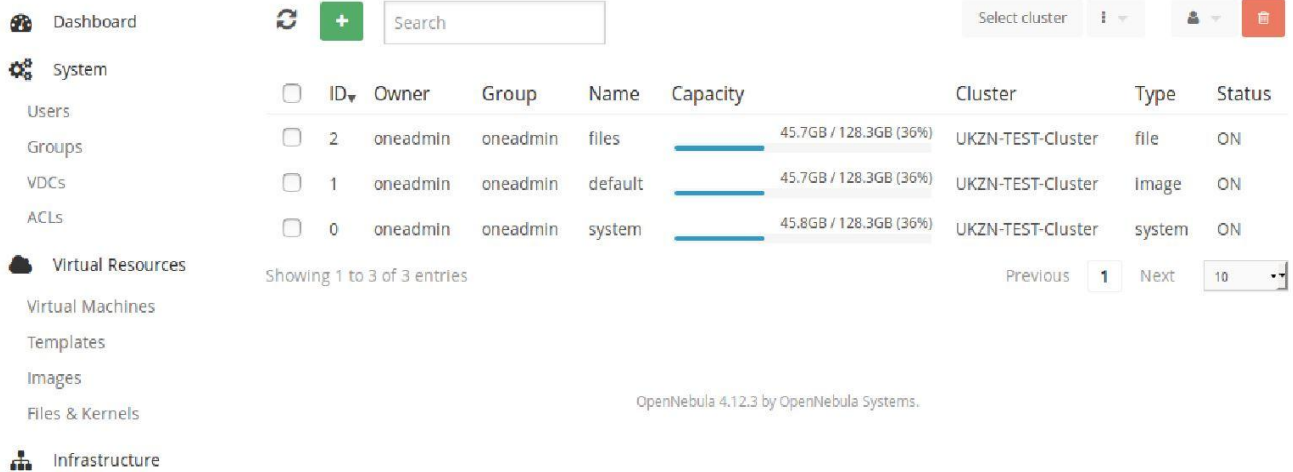
Figure 5.1: System's specifications

A cloud infrastructure was set up using openNebula. The version of openNebula used is 4.12.3. This version uses a kernel-based virtual machine (KVM) as its hypervisor. The KVM is used for virtualization techniques [131]. This is done in order to test the application and practicability of CloudCrypt on a live cloud infrastructure. CloudCrypt is used as an external plugin in openNebula. This enables CloudCrypt to be used as an encryption service on the cloud. The following screen shots show some parts of the openNebula platform.



Figure 5.2: OpenNebula Sunstone Server login

OpenNebula has a client-server architecture model. Within it, a server is created. The server can also be a client. In this study, openNebula was set up in a manner that allows one server and multiple client machines. The clients connect to the server for services. The machine specified in figure 5.1 was configured as a front-end node i.e. server. One cluster named UKZN-cluster was created. A cluster refers to a group of data centers with the same specifications. Within this cluster, three data centers were created. Each data center consists of 128 GB of memory. Each data center stored different type of files. For example, one was used to store images. Another was used to store system files such as operating systems images. The following screenshot shows the created data centers.



OpenNebula 4.12.3 by OpenNebula Systems.

Figure 5.3: Data centers

In order to connect to the server, host machines were created. A host machine is a server which is capable of running virtual machines. Three host machines were added to the UKZN cluster. The hosts were of different specifications. For example, Host1 and host 2 respectively, had the following specifications:

Host1: Memory: 7.18Gib, Processor: Intel Core i7-2600 @ 3.40GHz x 8

Host2: Memory: 2Gib, Processor: Intel Pentium @ 1.80GHz

The following screenshot shows how openNebula was used to monitor a host machine.

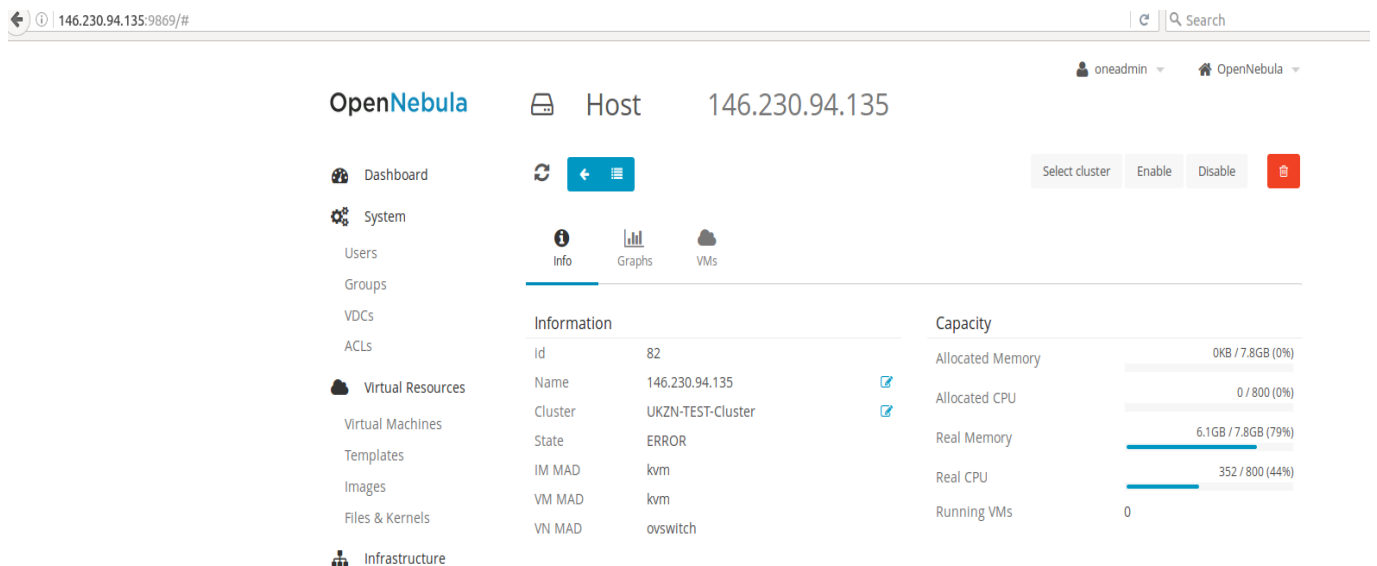


Figure 5.4: Host 1 monitoring in openNebula

Virtual machines (VMs) were created in all hosts. The VM sizes must not exceed the size of the host machine. The following screenshot shows some of the results when VMs are created in openNebula.

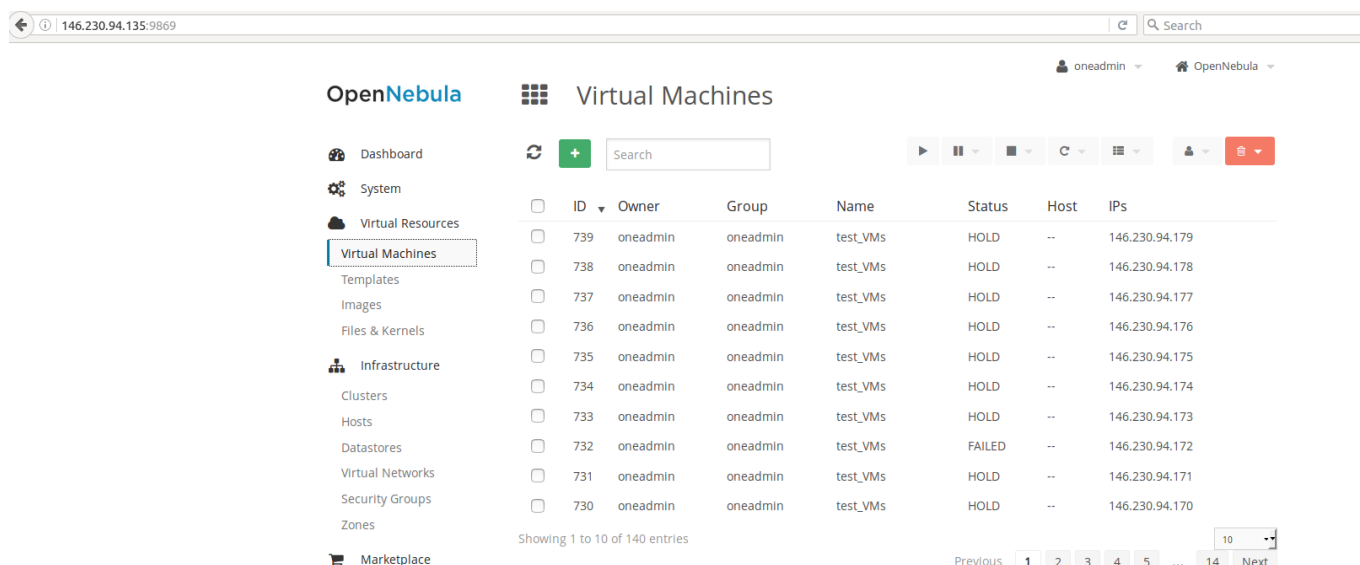


Figure 5.5: VMs on the UKZN cluster

5.3. Experimental results

The following figure shows how Eureqa optimizes input random noise.

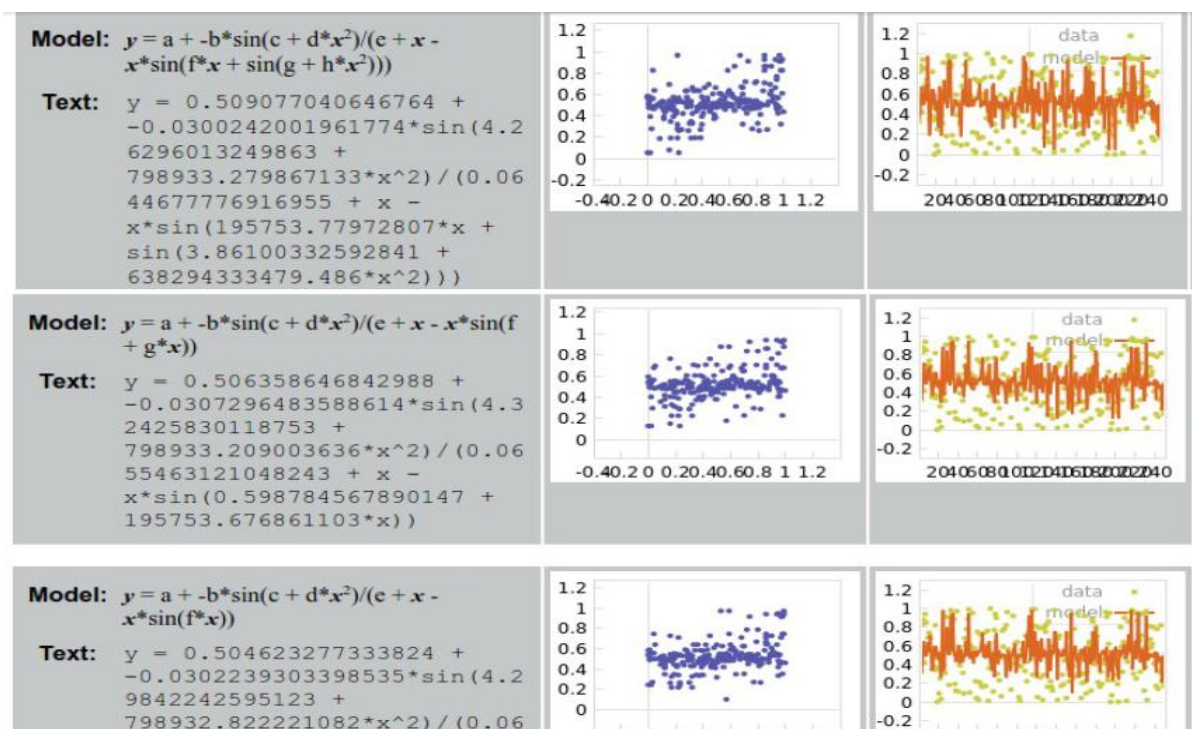


Figure 5.6: Generation of fitness functions

Figure 5.6 above depicts some the various fitness functions that were produced, given different chaotic noise inputs from [111]. Several experiments were conducted to test the proposed solution. The experiments were run to measure the system's resource (i.e. memory and processor usage) consumption, effectiveness and efficiency. The initial tests were run to generate the random noise using a cloud-based Eureka system. The system ran on different computers for a minimum of 24 hours to produce various non-linear fitness functions that best approximates the input random noise. Different sources of noise inputs were used on the Eureka system.

5.3.1. Key generation

As shown in the above snap-shot, the best fitness function which resembles an approximation of the input random noise is shown on the left. The input random noise is graphically shown using an $x-y$ plot in the middle and the resultant $x-y$ plot fitness function on the right. The fitness function is then used to generate symmetric encryption keys. The encryption keys are used to encipher cloud-bound data. The strength of the encryption keys is dependent on the distribution of the random input noise. For example, the following figure shows a graphical representation of the input noise. It shows the random distribution of the input noise when plotted on a 2-dimensional graph. The graph shows how randomly distributed the input is. This goes on to show that having a random input will improve the strength of the encryption keys.

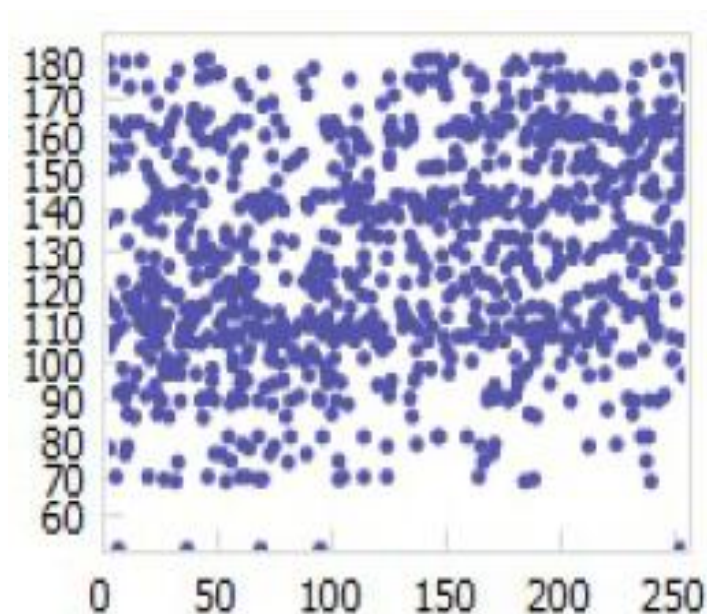


Figure 5.7: Random noise

The random noise depicted in figure 5.7 above resulted into the following non-linear fitness function.

$$\begin{aligned}
 f(x) = & 128.007009910725 + 14.0651581392666 * \cos(17.04314865355865 * x) + \\
 & \sin(128.69389713836 * x) + factorial(1.21540061377888 * \cos(17.04314865355865 * \\
 & x)) - 38.0330632716394 * \sin(5.273797071530436 + 0.00104272337047579 * x^2 + \\
 & \sin(128.69389713836 * x) - 0.226280766046429 * x)
 \end{aligned}
 \tag{5.1}$$

Equation 5.1 shows the generated fitness function. It is used as input into the key generator. The key generator randomly generates an encryption key. The following diagram shows how random floating point numbers are generated in order to create an encryption key.

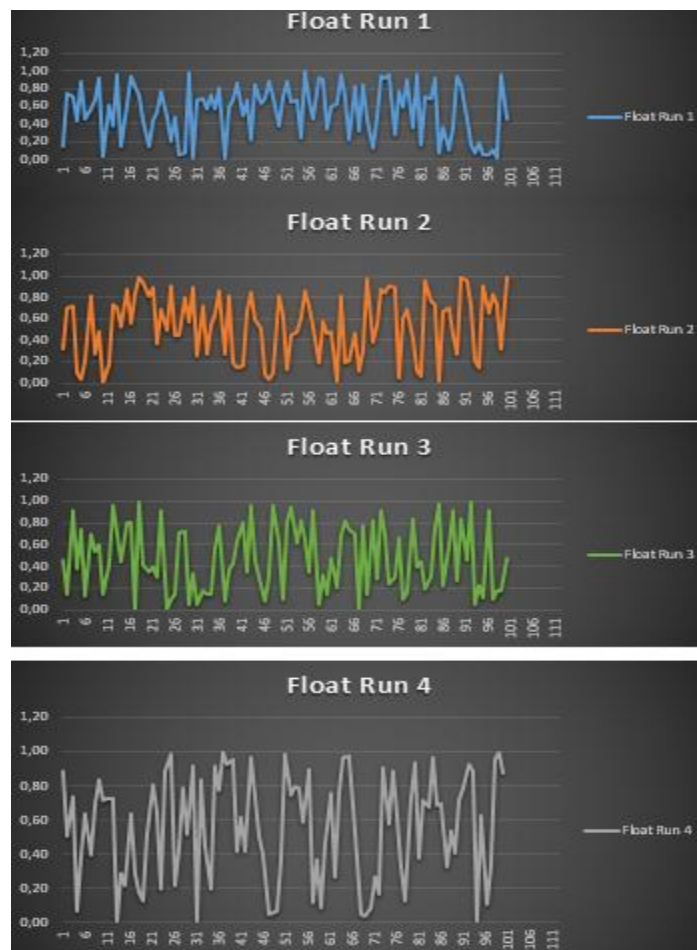


Figure 5.8: Generation of floating point numbers

Figure 5.8 above depicts four runs of floating point generation. The generated floats are used for normalizing the fitness function. Given the fitness function, random floating point numbers are generated. From this list of random floats, an initial random 56-bit key is generated. The fitness functions are normalized with these random floating point numbers in the range [0, 1]. There were four runs because the plaintext length was longer than the initial length of the encryption key (i.e. 56 bits). This resulted into four key generations. The keys are then iteratively concatenated to create a final key. If the key length is greater or equal to the plaintext length, the key generator stops. The final key is used for encryption. If the key length is greater than the length of the plaintext, the key bits are trimmed and extra bits are discarded. The following graphs show a set of four keys resulting from the four floating point normalizations above. Float run1 results into key gen 1, float run 2 results into key gen 2 etc.

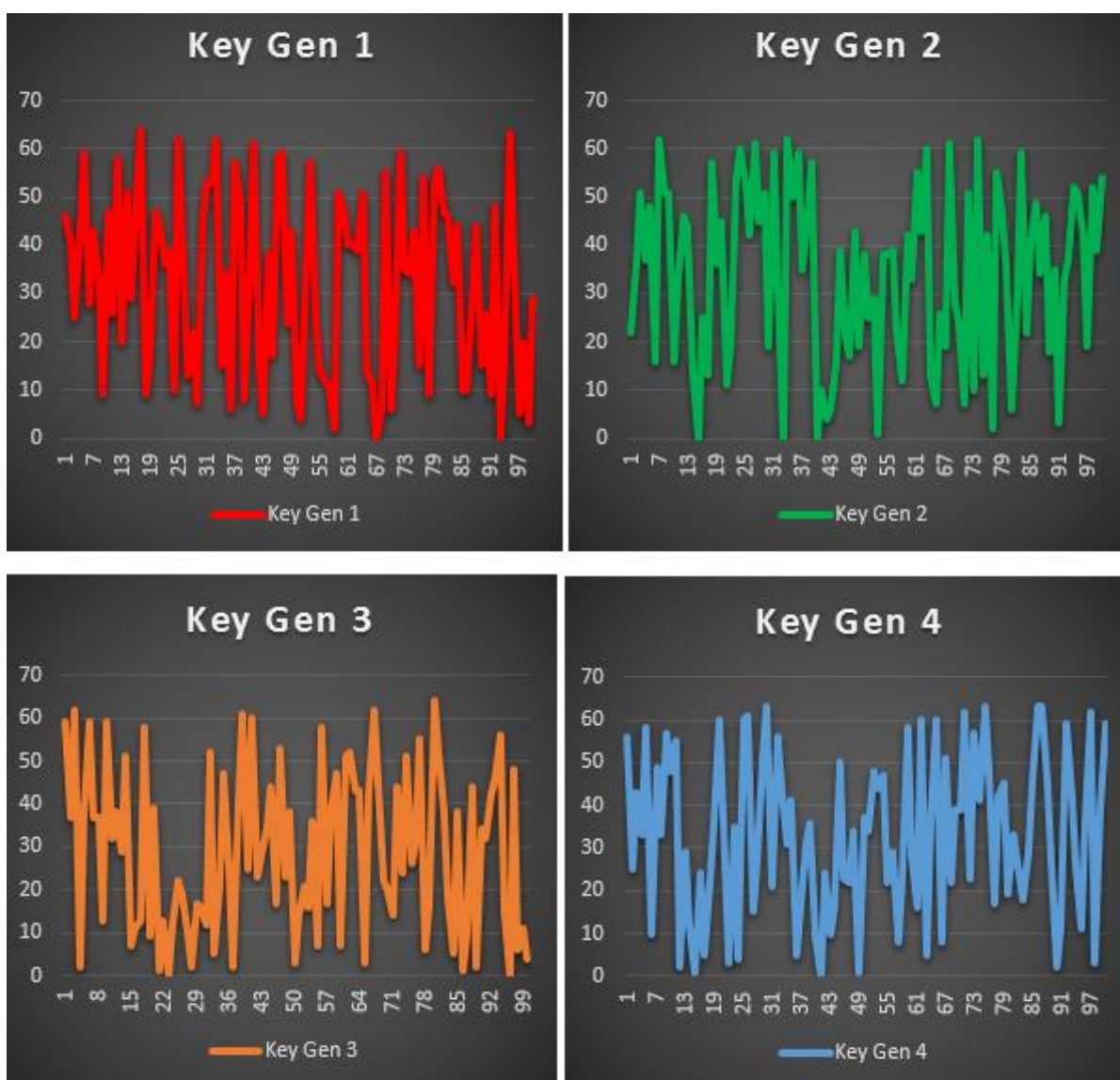


Figure 5.9: Key generation

The next section discusses experiments conducted encrypting the plaintext.

5.1.1. Encryption

The following table illustrates the experiments done using CloudCrypt, Data Encryption Standard (DES), triple DES (3-DES) and variants of Advanced Encryption Standard (AES). The results shown in the table below were obtained after 10 executions of the systems and an average of the results was recorded for each cryptosystem.

Cyptosystem	Cloud Crypt	DES	3-DES	AES 128	AES 192	AES 256
Key Size (bit)	56 \geq	64	192	128	192	256
File Sizes (byte)	167	167	167	167	167	167
Encryption Time (ms)	126.1	12.4	13.4	87.9	164.5	149.6
Decryption Time (ms)	128.3	10	14	67.7	143.7	130.4
Memory (mb)	8	4	4	6.3	11.1	15.3
CPU (percent)	1.008739	1.286582	1.321932	1.004848	1.006667	1.0094462

Table 5.1: Comparison of cryptosystems

Table 5.1 shows the key sizes used by each cryptosystem, encryption and decryption times taken to encrypt a text file of 167 bytes. Resources measured include memory and CPU. Furthermore, table 5.1 shows for example that CloudCrypt uses a 56 bit or more encryption key size while DES uses a 64 bit key, 3-DES uses a 192 bit key, AES-128 uses a 128 bit key, AES-192 uses a 192 bit key and AES-256 uses a 256 bit key.

Table 5.1 illustrates the amount of time (in milliseconds) recorded to encrypt and decrypt a text file of 167 bytes. It also shows CPU (in percentages) and memory usage (in megabytes) recorded for each cryptosystem's execution. Consequently, from the entries in table 1, independent graphs were recorded. The results below show encryption and decryption times in milliseconds (ms), memory usage in megabytes (MB), and processor usage in percentages (%).

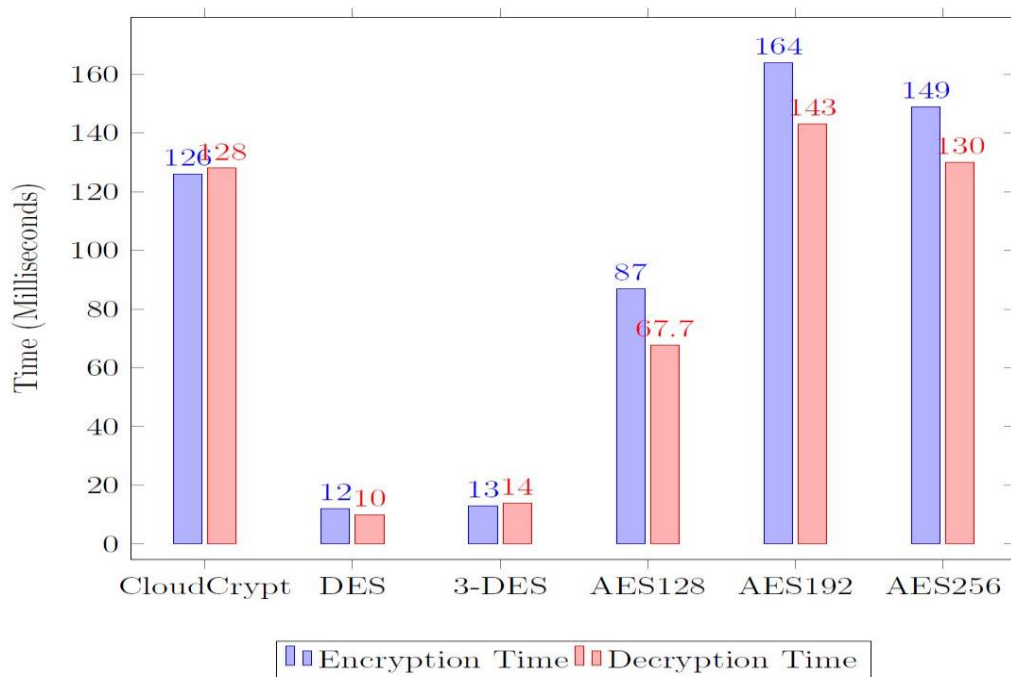


Figure 5.10: Encryption and decryption times

Figure 5.10 depicts encryption and decryption times for all cryptosystems. CloudCrypt took 126 milliseconds to encrypt the text file. DES encrypted the text file in 12 milliseconds. 3-DES encrypted the text file in 13 milliseconds. AES-128 took 87 milliseconds, AES-192 took 164 milliseconds and AES-256 recorded 149 milliseconds to encrypt the same text file. In this regard, only DES, 3-DES and AES-128 encrypted the text file quicker than CloudCrypt. The former encrypts the text file with least amount of time. CloudCrypt outperformed AES (192 and 256). It must be noted that this results were obtained after ten (i.e. 10) consecutive runs. This proves that on average, CloudCrypt performs better than the other two variants of AES (i.e. AES-192 and AES-256). It should also be noted that CloudCrypt is a stream cipher while the other cryptosystems are block ciphers. This means that CloudCrypt encrypts each individual plaintext bit to produce a corresponding ciphertext bit. The other cryptosystems encrypt a block of plaintext bits to produce a block of ciphertext bits. For example, DES takes an input plaintext of 64 bits to produce a 64 bit block of ciphertext. This aspect may affect the results as stream ciphers are slower. The next paragraph discusses decryption results.

Figure 5.10 also depicts a plot of decryption results. It should be noted that each cryptosystem produced different ciphertext files. These ciphertext files have different sizes. For example, given a 167 bytes input plaintext file, DES produced a 120 bytes ciphertext file. CloudCrypt produced a 162 bytes ciphertext. AES (AES-128, AES-192, and AES-256) produced 150 bytes, 146 bytes, 162 bytes respectively. For

each of the 10 runs, the ciphertext file was somehow having different sizes. The ciphertext file sizes discussed herein are approximated and averaged. It took 128 milliseconds for CloudCrypt to decrypt the ciphertext. DES decrypted the ciphertext file in 10 milliseconds and 3-DES took 14 milliseconds to decrypt the ciphertext. AES-128 took 67.7 milliseconds. AES-192 took 143 milliseconds while AES-256 took 130 milliseconds to decrypt its ciphertext. Once again, DES recorded the least amount of decryption time followed by 3-DES and AES-128. CloudCrypt performed better than AES-192 and AES-256 in terms of decryption times. CloudCrypt might produce larger ciphertext files because each byte is converted into the American Standard Code for Information Exchange (ASCII). Therefore, all plaintext characters, including white spaces, are converted and encrypted. The next discussion is on CPU and memory usage for all cryptosystems

The following figure illustrates CPU usage results.

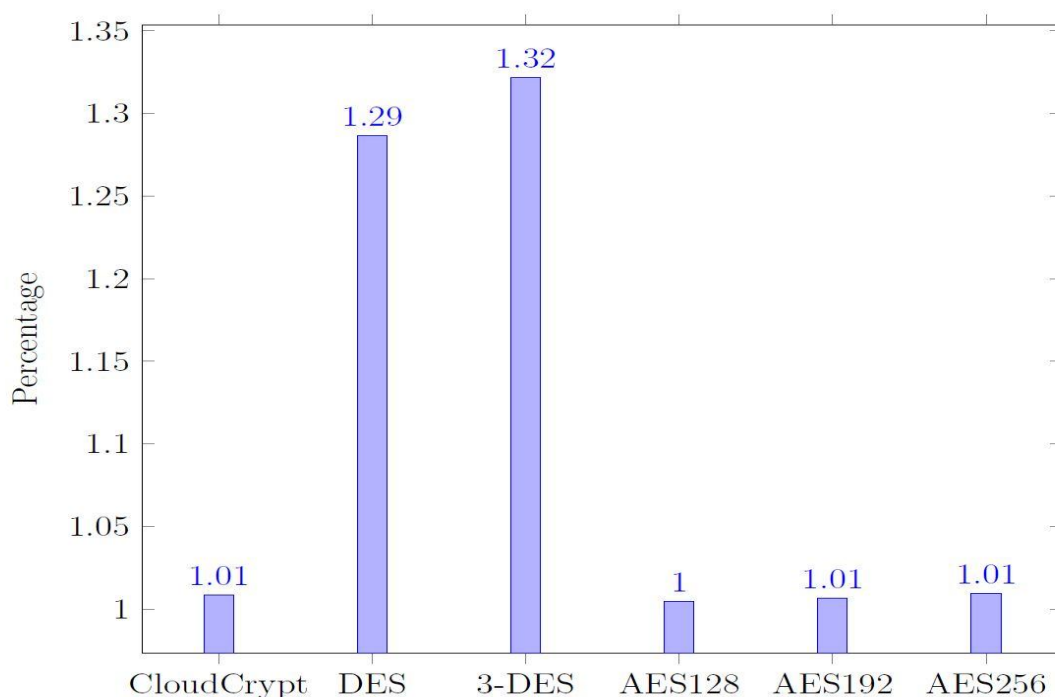


Figure 5.11: CPU usage

Figure 5.11 is an illustration of CPU usage. CloudCrypt, AES-128 and AES-256 occupied the CPU 1.01% of the time. AES-128 occupied the CPU 1% of the time. DES had 1.29% share of the CPU and 3-DES had 1.32%. These results mean that while the application was running, a process was created. Each process was then recorded using a task manager. However, the task manager the values for each process from when the application to execute. Consequently, such readings would not be a true measure

as the result we are interested in pertains to the encryption process. The Java system monitoring library (i.e. java sysmon) was used to get the encryption process, from the time the cryptosystems started the encryption and decryption. This library is used to get live OS independent process to get system performance. To this end, CloudCrypt and AES had closely related CPU usage. DES and 3-DES encryption occupied the CPU the most. The next discussion is on memory usage.

The following diagram shows memory consumption for the six cryptosystems.

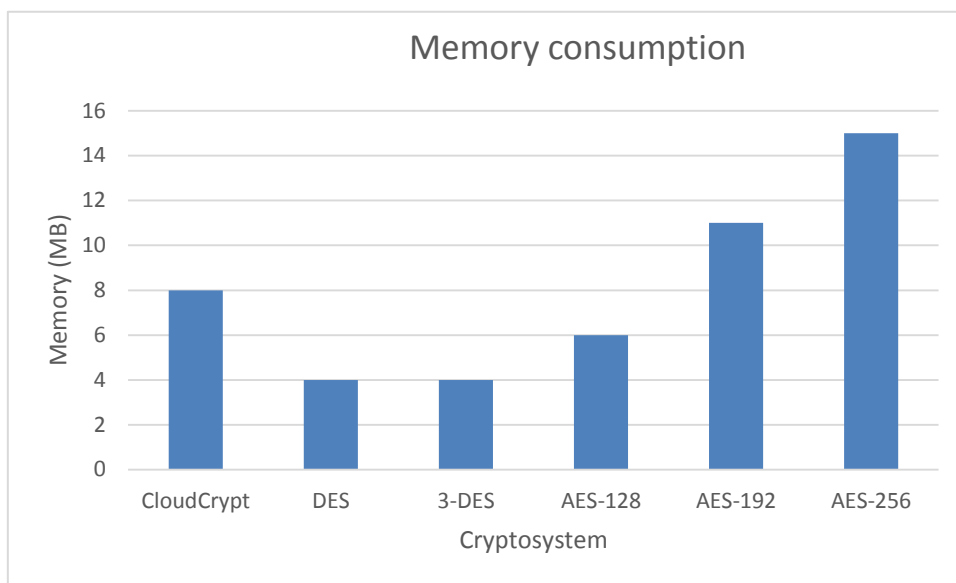


Figure 5.12: Memory consumption

Figure 5.12 is an illustration of memory usage for each cryptosystem after 10 runs. The readings are rounded up to the nearest MB. Java system monitoring library was used to get the exact records for each cryptosystem. CloudCrypt occupied the random access memory (RAM) and took 8MB of the system's memory. DES and 3-DES were tied at 4MB. AES-128 consumed 6MB, AES-192 used 11MB and AES-256 used 15MB of the system's memory. CloudCrypt performed better than AES-192 and AES-256. This is a positive result as one of the system's requirements was to develop a lightweight cryptosystem. This means CloudCrypt may be embedded in devices with low CPU and memory specifications e.g. mobile phones. The next section discusses the process of learning the key.

5.1.2. Key learning

The figure below depicts the CloudCrypt’s key management module results when being trained to learn the generated key.

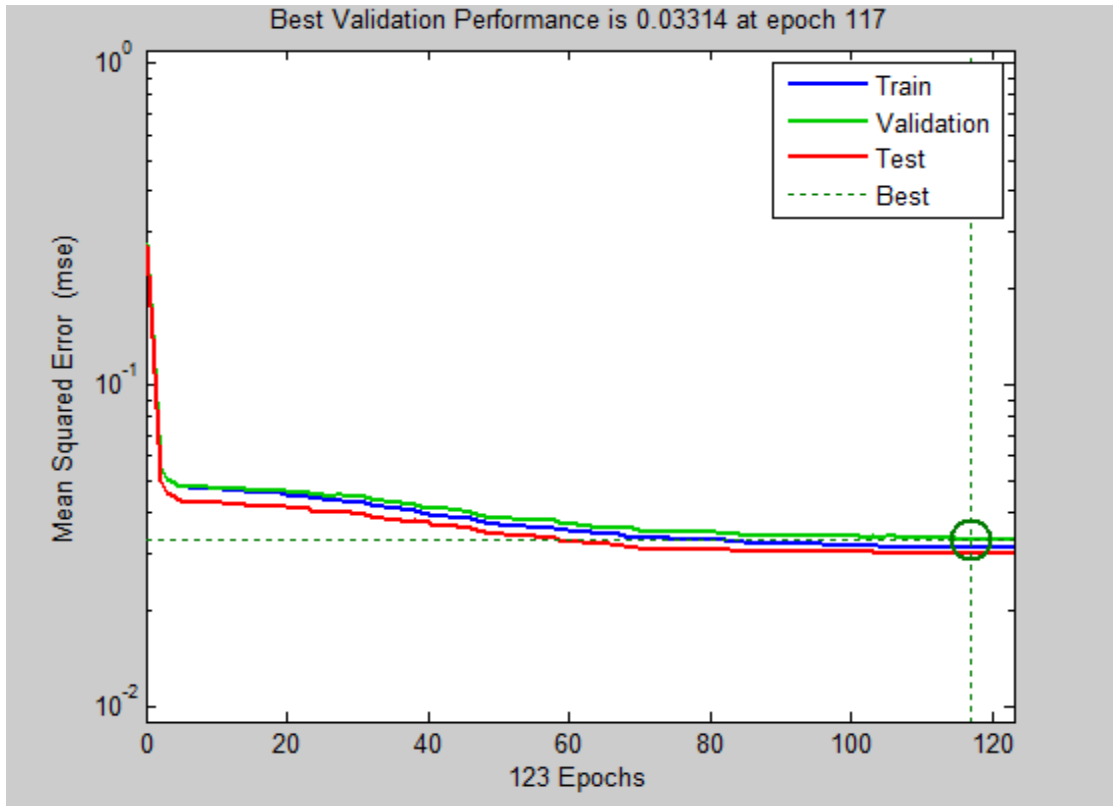


Figure 5.13: Training the neural network for key learning

Figure 5.13 shows the best validation of the encryption key when training the neural network. For graphical presentations, the experiment was conducted on MatLab version 2016a. The CPNN took 117 epochs (i.e. iterations) to do pattern matching in learning the encryption key structure. This means that the neural network managed to learn the key after 117 iterations. However, it has been noted that large inputs (i.e. keys and ciphertext) result in a huge neural network. Due to the huge size of the neural network, the system consumes large amounts of memory and processing power which eventually led to longer execution times. However, it should be noted that this aspect was evaluated on its own, not as part of the entire cryptosystem.

The next section discusses a secret sharing scheme implemented with CloudCrypt to aid cloud users in sharing their confidential data.

5.1.3. Secret-Sharing

After encrypting the plaintext using the CloudCrypt algorithm, the next task is to evaluate the sharing scheme. The secret sharing scheme enables data owners to distribute encryption key shares to participating entities in the scheme. The participating entities need not be the entities with whom the encrypted data is to be shared. Once the data owner and other users have acknowledged being part of the scheme, the data owner may invoke the secret sharing scheme process.

This process begins with a hand-shaking process. The hand-shaking process enables participants to join the scheme with legitimate identities. Thus, cloud tenants use their unique tenant identifications (IDs) for authentication. The data owner then begins the secret sharing process on the key. The following results show communication times (i.e. the time it takes to transfer each secret key share from the owner to the intended recipient). This time includes the amount of time it takes to apply the secret sharing scheme on the key, splitting it into shares and generating its corresponding hash using a one-way hash function such as Secure Hash Algorithm (SHA-256). It is important to note that the key length is variable and depends on the length of the plaintext. For this reason, five runs were conducted and an average of the runs was plotted as the final result. This was done to get a clear picture on the performance of the secret sharing scheme with regards to the different key sizes. This was done in order to remove any biasness of one key size. The following table depicts the number of key shares required for a $(3, n)$ i.e. 3 out-of n secret sharing scheme given a plaintext file size in kilobytes, where n is the total number of shares. The number of participants is determined by the data owner. For these experiments, 5 participants were part of the scheme.

Number of shares	File size (Bytes)
11	10
13	20
25	30
28	40
51	50

Table 5.2: Number of shares vs File size

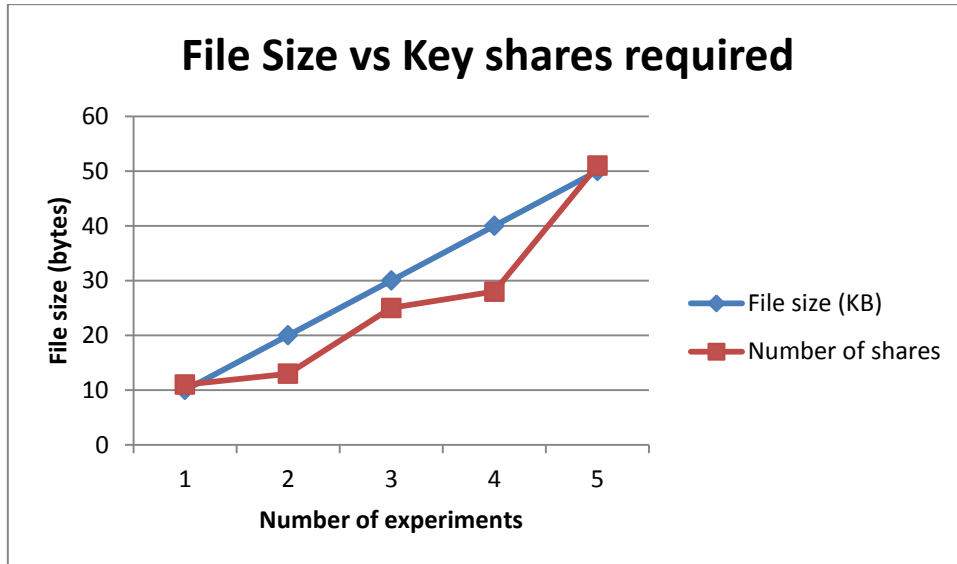


Figure 5.14: File size vs key shares

As demonstrated in figure 5.14 above, the bigger the file gets, the more key shares are required. This is true for all secret sharing schemes. The longer the shared message becomes, the more shares there will be [126]. Furthermore, CloudCrypt works like a one-time pad (OTP) encryption algorithm, where the key size must equal the plaintext length.

The next results show the various key lengths plotted against the number of key shares required to establish a secret sharing scheme. The key shares are based on the number of participants (i.e. 5) used in the scheme.

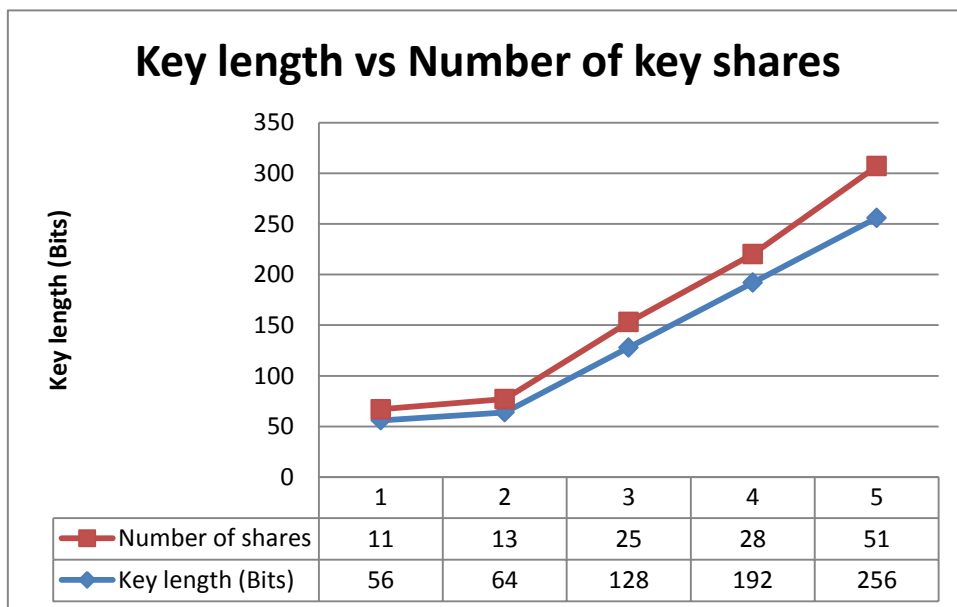


Figure 5.15: Key length vs number of key shares

Figure 5.15 above depicts the number of encryption key shares plotted against the number of bits making the key. It can be observed that the more bits there are, more key shares are needed. The scheme involved five participants only, with the minimum key bits being 56 bits. This resulted in generating eleven (i.e. 11) key shares. As the plaintext length increases, the key size also increases, which results into having more shares. The number of encryption key shares is rounded off to the nearest whole number. The next section delves into the discussion of the above results.

5.2. Discussion of results

The results presented herein compare CloudCrypt with other encryption systems in the field of Cryptography. The encryption systems used for comparisons are DES, 3-DES and AES-128, AES-192 and AES-256. These cryptosystems were chosen because they are symmetric. Hence, they have some similarities with CloudCrypt. CloudCrypt performed better than AES-192 and AES-256 in terms of encryption and decryption times. The result shows the success CloudCrypt has when encrypting text files. It should be noted that the result is based on text files alone. Encrypting other digital media such as images, videos or audios could produce different results.

In terms of computing resources such as CPU and memory, CloudCrypt also has better results compared to DES and 3-DES. This result means that CloudCrypt is indeed a light-weight cryptosystem. It therefore meets the objectives of this study as set out in chapter 1. CloudCrypt can be deployed on computing devices with low memory and CPU specifications given that it is designed to be a client-side cryptosystem. Most client devices have low specifications. Hence, CloudCrypt can be executed from them.

Secret sharing is another attribute of CloudCrypt, which the other cryptosystems do not possess. It is used for encryption key distribution. This attribute makes CloudCrypt ideal for deployment in a multi-user environment such as the cloud. Secret sharing is based on the number of participants used in the scheme. For experimental purposes, five participants were involved in the scheme. Key shares also depend on the file sizes (in kilobytes). For example, file of size 10 bytes requires 11 key shares. Consequently, a 56 bit key is required to have 11 shares. These shares are distributed to the participants in the scheme. In the same manner, 13 shares are required for a file size of 20 kilobytes. This means that a 64 bit key is needed to have those key shares, as illustrated in figure 5.15. A file size of 30 kilobytes requires 28 key shares as demonstrated in figure 5.14. As a result, figure 5.15 shows that a 192 bit key is required.

Moreover, it should be noted that the file size was 167 bytes. This means that a 192 bit key is needed for the secret sharing scheme to work, instead of the initial 56 bit key used. This only applies to secret sharing. Padding is used to increase the plaintext length in binary form if the secret sharing scheme is to be used. The padding bits are zeros (i.e. 0). Because the key generator concatenates initial keys to newly generated keys, the key is not padded. Then encryption can proceed with the secret sharing scheme activated.

5.3. Conclusion

The analyses presented above depict various measures, i.e. CPU, memory, key size, encryption and decryption times. From the results obtained it is evident that the AES and CloudCrypt perform better than the rest on CPU usage. However, AES-192 and AES-256 are the worst in terms of its memory usage, encryption and decryption times.

CloudCrypt makes use of variable key sizes with the shortest key being 56 bits. It is for this reason that multiple executions (i.e.10 runs) were conducted. From the ten experiments, an average performance across various characteristics was recorded. CloudCrypt performs in a similar manner to AES. This is in terms of CPU and memory usage. On this aspect, CloudCrypt very closely matched one of the highly favoured cryptosystems in AES. Based on the results obtained, the memory usage is low high for CloudCrypt. It would therefore be a better cryptosystem if adopted for industry use. It can be deployed on devices with low CPU and memory specifications.

Moreover, in terms of efficiency in learning the key patterns, CloudCrypt performed better than expected, with a success rate of 93.3%. This is one of the strongest attributes of CloudCrypt which could be the reason why it compares with AES in terms of encryption and decryption times while it out-performs DES and 3-DES.

In terms of secret sharing abilities, the system performed exceptionally well. However, other bits were lost due to optimisation techniques using Lagrange's interpolation. Hence, the entire key could not be reconstructed, leading to partial decryption at times. This is also because of truncation errors which result from rounding off the floating point numbers to ten (i.e. 10) decimal places. Also, trimming the bits has an effect as some encryption key bits were lost after calculating the weights in the CPNN. Lastly, based on the results depicted above, we conclude that CloudCrypt is a better cryptosystem than some of the current and widely used encryption schemes.

Overall, we conclude based on the results obtained, that CloudCrypt is a lightweight, strong client-side encryption scheme. CloudCrypt's performance against the existing, widely known cryptosystems makes it a better encryption scheme in terms of encryption and decryption times. Furthermore, its overall low memory consumption makes it suitable for deployment in devices with low memory specifications. The next chapter focuses on the conclusions drawn from this dissertation together with future perspectives regarding this study.

This page is intentionally left blank.

Chapter 6 – Conclusion and future work

6.1. Introduction

The previous chapter discussed the implementation details as well as the experimental results of the proposed cryptosystem – CloudCrypt. It also discussed the implementation of openNebula cloud infrastructure. This chapter provides a conclusion of the work done in this dissertation. It also recommends future perspectives.

6.2. Conclusions

Cloud computing brings various benefits. However, adoption of cloud services is hampered by security concerns. These concerns include lack of data confidentiality in cloud storages. In the quest to make cloud services secure, this dissertation presents a client-side cryptographic solution. It also discusses how it can be used to secure client data before it is uploaded to a public cloud infrastructure. The proposed solution gives cloud users full control over their cloud-hosted data. To enforce data confidentiality, encryption techniques are used. The proposed solution employs symmetric key encryption methods. It is a client-side cryptosystem which encrypts data before it leaves a trusted zone.

Throughout the course of this research, the main objective was to design, model and implement an efficient and strong cryptographic scheme. The proposed scheme uses chaotic random noise to improve the strength of encryption keys. The crypto scheme improves the strength of encryption through use of variable key sizes, while the encryption is static. The strength of the encryption keys does not necessarily rely on the length of the key but the random and chaotic nature of the input noise.

This study presented a model for the proposed cryptographic scheme which is the main contribution. The model presented in chapter 4 provides end-to-end encryption of cloud-bound data. This solution provides data confidentiality guarantees as it gives the data owner the responsibility to encrypt data before sending it to the cloud. Thus, not even the cloud service provider nor any third-party entity will have the knowledge of the encryption keys used to encrypt the data. The proposed solution is a lightweight mechanism as shown by the results. It can be integrated in devices with low system specifications (i.e. CPU and memory).

Cloud security alliance notes that key management is a recurring problem in the cloud. To address the prevalent key management problem, this dissertation proposes the use of a counter propagation neural

network (CPNN). The CPNN is trained on the patterns of the ciphertext and the corresponding key combination. Adoption of this cryptosystem enables cloud users to avoid cumbersome key storage requirements or retaining compromised keys unaware.

As discussed in the previous chapters, key management consists of any process dealing with the key, excluding the encryption and decryption process itself. It involves creation and destruction, revocation, activation or deactivation, transportation, storage of keys etc. As much as CloudCrypt addresses key management processes, it puts a lot of burden on the data owner. This is because with CloudCrypt, the data owner creates groups of participants who will have access to the encrypted data.

Efficient data sharing in a distributed, multi-tenant cloud environment remains a big challenge. The solution proposes and implements a secret sharing scheme. The scheme achieves perfect secrecy. This is made possible by encryption key splitting into the desired number of shares. The key shares are distributed to authorized and authenticated users. To show the proof of concept, an OpenNebula IaaS cloud was successfully set up as a test bed. OpenNebula is currently without an encryption module as part of its architecture.

Through several experiments conducted, CloudCrypt meets its objectives. A cloud infrastructure was set up using OpenNebula. This was done in order to test the application and practicality of CloudCrypt on a live cloud infrastructure. OpenNebula was successfully integrated with CloudCrypt to enhance cloud security, regarding data confidentiality. The experiments conducted reveal that indeed, CloudCrypt performs well and meet its objectives. The successful implementation of CloudCrypt addresses the problem statement of this dissertation. Adoption of CloudCrypt provides data confidentiality guarantees through a client-side encryption and key management system. Using CloudCrypt, data owners have their data encrypted before it can be send to the cloud. Additionally, encryption keys do not need to be stored. This eliminates the cumbersome and often problematic key management associated with encryption systems. With all the success achieved in this research, the belief is that data confidentiality is still full of challenges and of paramount importance in the cloud, and many research problems remain to be identified and investigated. The next section points to the recommendations for future work.

6.3. Future work

In order to enforce data confidentiality guarantees, various security issues still need to be analysed. As such, future perspectives of interest include:

- Investigation of CloudCrypt adaptability to encrypt multi-media digital content.
- With the prevalence of mobile devices, a mobile application solution offering cloud storage such as Dropbox can be used and the proposed cryptosystem be implemented on a mobile device for integration with cloud storage infrastructure.
- Generate keys for other symmetric ciphers then use the same key to evaluate encryption.
- Implement the system to have rounds of encryption to make it computationally infeasible to crack.

References

- [1] Gantz J. and D. Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView*, (December):1–16, 2012.
- [2] Mosola N.N., Dlamini M.T., Eloff J.H.P., Eloff M.M. (2016). *Evolutionary Neural Crypto-System for Cloud-bound Data*. Southern Africa Telecommunications Networks and Applications Conference (SATNAC), George, South Africa. Pp 1-2.
- [3] 451 Research. Enterprise cloud computing poised for explosive growth during next two years, September 2013. [Online] Available: https://451research.com/images/stories/Marketing/press_releases/cloud_wave_5_press_release_final.pdf. [Accessed: 18 – Sep - 2016].
- [4] Boose S. Cloud computing adoption by federal agencies increases 400%, December 2013. [Online] Available: <http://www.tripwire.com/state-of-security/latest-security-news/cloud-computing-adoption-federal-agencies-increases-400>. [Accessed: 18- Sep – 2016]
- [5] Michael A., Armando F., Rean G., Anthony D.J, Randy K., Andy K., Gunho L., David P., Ariel R., Ion Sto-ica., et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.
- [6] Mark D. Ryan. Cloud computing privacy concerns on our doorstep. *Communications of the ACM*, 54(1), 2011.
- [7] Khajeh-Hosseini A., Sommerville I., Sriram I., (2010). Research Challenges for Enterprise Cloud Computing. Submitted to the 1st ACM Symposium on Cloud Computing, SOCC 2010.
- [8] A. Shawish and M. Salama, 2014. *Cloud Computing: Paradigms and Technologies*, F. Xhafa and N. Bessis (eds.), *Inter-cooperative Collective Intelligence: Techniques and Applications*, *Studies in Computational Intelligence* 495, DOI: 10.1007/9783-642-35016-0_2, Springer-Verlag Berlin Heidelberg.
- [9] Dlamini M.T., Eloff J.H.P., Eloff M.M. (2014). Conflict Based Allocation Control for Cloud. The 9 th International Conference for Internet Technology and Secured Transactions (ICITST-2014), London,2014,Pp447-448.URL:<http://ieeexplore.ws/stamp/stamp/jsp?tp=&arnumber=7038854&isnumber=7038754>
- [10] Ratsoma, M.S., Dlamini, M.T., Eloff, J.H.P. and Venter, H.S. (2015). A Conflict-Aware Placement of Client VMs in Public Clouds. In: 10th International Conference on Cyber Warfare and Security. Reading: Academic Conferences and Publishing International Limited, pp.501-503
- [11] Trend Micro, (2016). Security Threats To Evolving Data Centers. *Virtualization and Cloud Computing*. [online] pp.12-15. Available at:http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/reports/rpt_security-threats-to-datacenters.pdf. [Accessed 29 Apr. 2016].

- [12] Szefer J, and Lee R.B. (2014). A Case for Hardware Protection of Guest VMs from Compromised Hypervisors in Cloud Computing, in Proceedings of the Second International Workshop on Security and Privacy in Cloud Computing (SPCC 2011), Kyoto, Japan, June 2011; DOI: 10.1109/ICDCSW.2011.51.
- [13] Perez-Botero D., Szefer J., and Lee R.B. (2013). Characterizing Hypervisor Vulnerabilities in Cloud Computing Servers. In Proceedings of the Workshop on Security in Cloud Computing (SCC 2013). Hangzhou, China, 8 May 2013.
- [14] National Institute of Science and Technology. [Online] Available: <http://csrc.nist.gov/publications/nistpubs/800-146/sp800-146.pdf>. [Accessed 17 – Sep – 2016].
- [15] "Lessons From ATM Cash-Out Scheme in Japan", *Bankinfosecurity.com*, 2016. [Online]. Available: http://www.bankinfosecurity.com/lessons-from-atm-cash-out-scheme-in-japan-a-9140?rf=2016-05-26-eb&mkt_tok=eyJpIjoiTkdkNEE1HWXpNekU1WTJKbSIsInQiOiJvS0d1YWIMQWdwY0o1RW5nQ0FUSEJtN2plbDcxS1k0dU5VUFNERXI3R3lvQjVnd0poeTY3TEtDRHZzNF15T3h1YkZaeHFiUk5jczFsRzRSeEd5aWZ4bWRlOTNpTXdOMmtqMnFCWWhlVlZzM0ifQ%3D%3D. [Accessed: 18- Sep- 2016].
- [16] "Standard Bank computer was hacked in R300m ATM fraud hit - report", *Fin24*, 2016. [Online]. Available: <http://www.fin24.com/Tech/Cyber-Security/standard-bank-computer-was-hacked-in-r300m-atm-fraud-hit-report-20160630>. [Accessed: 18- Sep- 2016].
- [17] D. Newton, "Dropbox authentication: insecure by design", *Dereknewton.com*, 2011. [Online]. Available: <http://dereknewton.com/2011/04/dropbox-authentication-static-host-ids/>. [Accessed: 18- Sep- 2016].
- [18] "The Sony hack: One year later", *CNBC*, 2015. [Online]. Available: <http://www.cnbc.com/2015/11/24/the-sony-hack-one-year-later.html>. [Accessed: 18- Sep- 2016].
- [19] "Target Hackers Broke in Via HVAC Company — Krebs on Security", *Krebsonsecurity.com*, 2016. [Online]. Available: <http://krebsonsecurity.com/2014/02/target-hackers-broke-in-via-hvac-company/>. [Accessed: 18- Sep- 2016].
- [20] Cloud Security Alliance, "Security Guidance for Critical Areas of Focus in Cloud Computing V2.1," <http://www.cloudsecurityalliance.org/csaguide.pdf>.
- [21] "IT Security - Dropbox Business", *Dropbox.com*, 2016. [Online]. Available: <https://www.dropbox.com/enterprise/security>. [Accessed: 25- Sep- 2016].
- [22] "Solutions", *Spideroak.com*, 2016. [Online]. Available: <https://spideroak.com/solutions>. [Accessed: 25- Sep- 2016].
- [23] B. Schneier, "Dropbox Security - Schneier on Security", *Schneier.com*, 2011. [Online]. Available: https://www.schneier.com/blog/archives/2011/05/dropbox_securit.html. [Accessed: 25- Sep- 2016].

- [24] S. Vaughan-Nichols, "No Privacy on Amazon's Cloud Drive | ZDNet", *ZDNet*, 2011. [Online]. Available: <http://www.zdnet.com/article/no-privacy-on-amazons-cloud-drive/>. [Accessed: 25- Sep- 2016].
- [25] M. Pinolla, "How To Add a Second Layer of Encryption to Dropbox [Updated]", *Lifehacker.com*, 2011. [Online]. Available: <http://lifehacker.com/5794486/how-to-add-a-second-layer-of-encryption-to-dropbox>. [Accessed: 26- Sep- 2016].
- [26] "2015 Cyber Attacks Statistics", *HACKMAGEDDON*, 2016. [Online]. Available: <http://www.hackmageddon.com/2016/01/11/2015-cyber-attacks-statistics/>. [Accessed: 25- Sep- 2016].
- [27] Shamir A. "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.
- [28] L. Badger, R. Patt-corner, J. Voas, L. Badger, R. Patt-corner, and J. Voas, "DRAFT Cloud Computing Synopsis and Recommendations of the National Institute of Standards and Technology," National Institute of Standards and Technology (NIST). [Online] Available: <http://csrc.nist.gov/publications/drafts/800-146/Draft-NIST-SP800-146.pdf>.
- [29] P. Mell and T. Grance, "The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology," *Natl. Inst. Stand. Technol. Spec. Publ. 800-145* 7 pages, PP 2–3.
- [30] K. Stanoevska-Slabeva, T. Wozniak, *Grid and Cloud Computing-A Business Perspective on Technology and Applications*, Springer-Verlag, Berlin, Heidelberg, 2010.
- [31] E. Naone, Technology overview, conjuring clouds, *MIT Technology Review*, July–August, 2009.
- [32] Merrill Lynch, *The cloud wars: \$100+ billion at stake*, Merrill Lynch, 2008
- [33] S. V Nandgaonkar and P. A. B. Raut, "A Comprehensive Study on Cloud Computing," vol. 3, no. 4, pp. 733–738, 2014.
- [34] Amazon. Amazon simple storage service (amazon s3)
- [35] Amazon Inc. Amazon Elastic Compute Cloud (Amazon EC2). Amazon Inc., <http://aws.amazon.com/ec2/#pricing>
- [36] H. Takabi, J. Joshi and G. Ahn, "Security and Privacy Challenges in Cloud Computing Environments", *IEEE Security & Privacy Magazine*, vol. 8, no. 6, pp. 24-31, 2010.
- [37] Google. Google app engine. Development, 2009(28th April 2010):1–10, 2011.
- [38] D. Chappell. Introducing the Windows azure platform. October, 30(October):2010, 2010.
- [39] www.github.com
- [40] Salesforce. (2011, Accessed February) Salesforce products. [Online]. Available: <http://www.salesforce.com/crm/products.jsp>

- [41] H. Wortmann, "Business consequences of cloud computing," Innovate IT 2010 Conference, October 2010.
- [42] R. Attebury, J. George, C. Judd, B. Marcum, and N. Montgomery. Google docs: a review. *Against the Grain*, 20(2):14–17, 2008.
- [43] <http://www.salesforce.com/>
- [44] <https://www.dropbox.com>
- [45] "Microsoft Cloud Platform - Advantages of Cloud Computing | Microsoft", *Microsoft.com*, 2016. [Online]. Available: <https://www.microsoft.com/en-gb/cloud-platform/default.aspx>. [Accessed: 01- Oct- 2016].
- [46] S. Yadav, "Comparative Study on Open Source Software for Cloud Computing Platform: Eucalyptus, OpenStack and OpenNebula", *ResearchGate*, 2013. [Online]. Available: https://www.researchgate.net/publication/262335770_Comparative_Study_on_Open_Source_Software_for_Cloud_Computing_Platform_Eucalyptus_OpenStack_and_OpenNebula. [Accessed: 01- Oct- 2016].
- [47] G. Gajala, "Cloud Computing: A State of Art of the Cloud", *International Journal of Computer Trends and Technology*, vol. 4, no. 1, pp. 1-4, 2013.
- [48] Liu, F. et al., "NIST Cloud Computing Reference Architecture," National Institute of Standards and Technology (NIST), Tech. Rep., Sep 2011.
- [49] Mell, P., Grace, T., "The NIST Definition of Cloud Computing," National Institute of Standards and Technology (NIST), Tech. Rep., Sep 2011
- [50] "Public and Private Clouds Explained", *Eci.com*, 2016. [Online]. Available: <http://www.eci.com/cloudforum/private-cloud-explained.html>. [Accessed: 01- Oct- 2016].
- [51] I. Llorente, "Cloud Interoperability and Portability with OpenNebula – OpenNebula", *Opennebula.org*, 2010. [Online]. Available: <http://opennebula.org/cloud-interoperability-and-portability-with-opennebula/>. [Accessed: 01- Oct- 2016].
- [52] A. Freier, et al. The Secure Sockets Layer (SSL) Protocol Version 3.0. Internet Engineering Task Force (IETF). Request for Comments: 6101. August 2011, <http://tools.ietf.org/html/rfc6101>
- [53] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. Network Working Group. RFC Editor, RFC 5246. issn 2070-1721. August 2008, <http://tools.ietf.org/html/rfc5246>
- [54] A. Ghosh, "68 million hacked Dropbox account details are available for free download", *International Business Times UK*, 2016. [Online]. Available: <http://www.ibtimes.co.uk/68-million-hacked-dropbox-account-details-are-available-free-download-1584861>. [Accessed: 12- Oct- 2016].

- [56] C. Brook, "LinkedIn Hacked and 6.5 Million Passwords Stolen - OPUSfidelis", *OPUSfidelis*, 2012. [Online]. Available: https://www.opusfidelis.com/insights/linkedin_hacked_and_6-5_million_passwords_stolen/. [Accessed: 12- Oct- 2016].
- [57] Ebay Online Shopping. Accessed October 10th, 2016, <http://www.ebay.com/>
- [58] S. Northcutt, "The Attack Surface Problem", *Sans.edu*, 2016. [Online]. Available: <http://www.sans.edu/research/security-laboratory/article/did-attack-surface>. [Accessed: 01- Oct- 2016].
- [59] W. Diffie and M. Hellman. *New directions in cryptography*, 1976.
- [60] Stallings W., *Cryptography and Network Security: Principles and Practice*, 5th ed.
- [61] Amazon. Amazon simple storage service (amazon s3). www.amazon.com
- [62] T. Rowland and E. Weissstein, "Group Homomorphism -- from Wolfram MathWorld", *Mathworld.wolfram.com*, 2016. [Online]. Available: <http://mathworld.wolfram.com/GroupHomomorphism.html>. [Accessed: 12- Nov- 2016].
- [63] R.L. Rivest, L. Adleman, and M.L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations on Secure Computation*, Academia Press, pages 170–173, 1978.
- [64] T Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *Information Theory, IEEE Transactions on*, 31(4):469–472, 1985.
- [65] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th international conference on Theory and application of cryptographic techniques, EUROCRYPT'99*, pages 223–238, Berlin, Heidelberg, 1999. Springer-Verlag.
- [66] D. Boneh, E. Goh, and K. Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Proceedings of the Second international conference on Theory of Cryptography, TCC'05*, pages 325–341, Berlin, Heidelberg, 2005. Springer-Verlag.
- [67] C. Gentry. A fully homomorphic encryption scheme. PhD thesis, Stanford, CA, USA, 2009
- [68] C. Van Dijk, M., and Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Tech-niques, EUROCRYPT'10*, pages 24–43, Berlin, Heidelberg, 2010. Springer-Verlag.
- [69] N. P. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Proceedings of the 13th International Conference on Practice and Theory in Public Key Cryptography, PKC'10*, pages 420–443, Berlin, Heidelberg, 2010. Springer-Verlag.
- [70] M. Naehrig, K. Lauter, and V. Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop, CCSW '11*, New York, NY, USA, 2011. ACM.

- [71] Maha TEBAA, Saïd EL HAJJI, Abdellatif EL GHAZI “Homomorphic Encryption Applied to the Cloud Computing Security” Proceedings of the World Congress on Engineering, London, U.K. ISBN: 978-988-19251-3-8 ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online), Vol 1, July 4 - 6, 2012.
- [72] A. Sahai and B. Waters. Fuzzy identity-based encryption. In Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques, EUROCRYPT’05, pages 457–473, Berlin, Heidelberg, 2005. Springer-Verlag.
- [73] R. Lakshmi, R. Laavanya, M. Meenakshi and C. Dhas, "Analysis of Attribute Based Encryption Schemes", *International Journal of Computer Science and Engineering Communications*, vol. 3, no. 3, pp. 1076-1081, 2015.
- [74] Padmapriya, Subhasri, “Cloud Computing: Reverse Caesar Cipher Algorithm to Increase Data Security”, *International Journal of Engineering Trends and Technology (IJETT)*, Volume 4, Issue 4, 2013.
- [75] V.U.K. Sastry, N. Ravi Shankar and S. Durga Bhavani, “A Generalized Playfair Cipher involving Intertwining, Interweaving and Iteration”, *International Journal of Network and Mobile Technologies*, pp 45-53, 2010.
- [76] Sugumaran, BalaMurugan. B, D. Kamalraj,” An Architecture for Data Security in Cloud Computing”, *IEEE World Congress on Computing and Communication Technologies* 2014.
- [77] A. Ciphers, "Advantages and disadvantages of Stream versus Block Ciphers", *Security.stackexchange.com*, 2016. [Online]. Available: <http://security.stackexchange.com/questions/334/advantages-and-disadvantages-of-stream-versus-block-ciphers>. [Accessed: 15- Nov- 2016].
- [78] Dropbox, A file-storage and sharing service. (2016). [Online]. Available: <http://www.dropbox.com>
- [79] Google Drive. (2016). [Online]. Available: <http://drive.google.com>
- [80] Mozy, Mozy: A File-storage and Sharing Service. (2016). [Online]. Available: <http://mozy.com>
- [81] Y. Zheng., D. Wenxiu, Y. Xixun., Z. Haiqi., H.D. Robert. *Deduplication on Encrypted Big Data in Cloud*. *IEEE Transactions on Big Data*, Vol. 2, NO. 2., April – June 2016.
- [82] D. Harnik, B. Pinkas, and A. Shulman-Peleg. Side channels in cloud services: Deduplication in cloud storage. *IEEE Security And Privacy*, 8(6):40–47, 2010.
- [83] Opendedup. (2016). [Online]. Available: <http://opendedup.org>

- [84] A. Shamir. Identity-based cryptosystems and signature schemes. In Proceedings of CRYPTO 84 on Advances in cryptology, pages 47–53, New York, NY,USA, 1985. Springer-Verlag New York, Inc.
- [85] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.
- [86] D. Boneh and M. FraBenklin. *Identity-based encryption from the weil pairing*. In Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology, CRYPTO '01, pages 213–229, London, UK, UK, 2001. Springer-Verlag.
- [87] Mashayekhy L., Nejad M.M., and Grosu D.A. (2014). Framework for Data Protection in Cloud Federation, 3rd International Conference on Parallel Distribution.
- [88] Kumar N., Katta V., Mishra H., and Garg H. (2014). Detection of Data Leakage in Cloud Computing Environments. *6th International Conference on Computational Intelligence and Communication Networks, IEEE Computer Society*; 803 – 807.
- [89] W. Kinzel, and I. Kanter, “Neural Cryptography”, Proc. of the 9th Int’l Conf. on Neural Information Processing (ICONIP’02), vol. 3, pp.1351-1354, 2002.
- [90] D. A. Karras, and V. Zorkadis, “On neural network techniques in the secure management of communication systems through improving and quality assessing pseudorandom stream generators”, *Neural Networks*, vol.16, issues 5-6, pp. 899-905, June-July 2003.
- [91] Abelson H., Anderson R., Bellovin S.M., Benaloh J., Blaze M., Whitfield D., Gilmore J., Green M., Landau S., Neumann P.G., Rivest R.L., Schiller J.I., Schneir B., Specter M., and Weitzner D.J. (2015). Keys under Doormats: Mandating Insecurity by Requiring Government Access to all Data and Communications. Available at: <http://dspace.mit.edu/bitstream/handle/1721.1/97690/MIT-CSAIL-TR-2015-026.pdf?sequence=8>
- [92] Griffin, A. (2016). WhatsApp just made maybe its most important ever update. [online] The Independent. Available at: [tech/news/whatsapp-update-encryption-end-to-end-messages-security government-privacy a6970101.html](http://www.independent.co.uk/tech/news/whatsapp-update-encryption-end-to-end-messages-security-government-privacy-a6970101.html) [Accessed 25 Oct. 2016].
- [93] Metz, C. (2016). Forget Apple vs. the FBI: WhatsApp Just Switched on Encryption for a Billion People. [online]. Available at: [http://www.wired.com/2016/04/forget-apple-vs-fbi-whatsapp-just-switched- encryption billion-people/](http://www.wired.com/2016/04/forget-apple-vs-fbi-whatsapp-just-switched-encryption-billion-people/) [Accessed 25 Oct. 2016].

- [94] WhatsApp.com. (2016). WhatsApp Security. [online] . Available at: <https://www.whatsapp.com/security/> [Accessed 25 Oct. 2016].
- [95] I. Blake, G. Seroussi, N. Smart, and J. W. S. Cassels. *Advances in Elliptic Curve Cryptography* (London Mathematical Society Lecture Note Series). Cambridge University Press, New York, NY, USA, 2005.
- [96] D. Ratna, B. Rana, and S. Palash. *Pairing-based cryptographic protocols : A survey*. 2004. <http://eprint.iacr.org/>
- [97] R. Sakai and M. Kasahara. Id based cryptosystems with pairing on elliptic curve. *Cryptology ePrint Archive*, Report 2003/054, 2003.
- [98] H W Lim and M J B Robshaw. On identity-based cryptography and grid computing. *Lecture Notes in Computer Science*, pages 474–477, 2004.
- [99] H W Lim and M J B Robshaw. A dynamic key infrastructure for grid. In *Proceedings of the 2005 European conference on Advances in Grid Computing, EGC’05*, pages 255–264, Berlin, Heidelberg, 2005. Springer-Verlag.
- [100] Vipul Goyal, Omkant Pandey, and et al. Attribute-based encryption for fine-grained access control of encrypted data, 2006.
- [101] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, “DepSky: dependable and secure storage in a cloud-of- clouds,” in *Proceedings of the sixth conference on Computer systems*. ACM, 2011, pp. 31–46.
- [102] C. Cachin, R. Haas, and M. Vukolic, “Dependable storage in the intercloud,” *IBM Research*, vol. 3783, pp. 1–6, 2010.
- [103] A. Kumbhare, Y. Simmhan, and V. Prasanna, “Cryptonite: a secure and performant data repository on public clouds,” in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. IEEE, 2012, pp. 510–517
- [104] R. A. Popa, J. R. Lorch, D. Molnar, H. J. Wang, and L. Zhuang, “Enabling security in cloud storage slas with cloudproof,” in *Proc. USENIX ATC*, 2011.

- [105] A. Fiat and M. Naor, "Broadcast encryption," in *Advances in Cryptology CRYPTO93*. Springer, 1994, pp. 480–491.
- [106] H. Xiong, X. Zhang, D. Yao, X. Wu, and Y. Wen, "Towards end-to-end secure content storage and delivery with public cloud," in *Proceedings of the second ACM conference on Data and Application Security and Privacy*. ACM, 2012, pp. 257–266.
- [107] Zeng, L., Chen, S., Wei, Q., & Feng, D. (2012). Sedas: a self-destructing data system based on active storage framework. In *APMRC, 2012 Digest* (pp. I -8). IEEE.
- [108] H. Krawczyk, "Secret Sharing Made Short," in *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, 1993.
- [109] cloud security alliance (2011). "*Security guidance for critical areas of focus in cloud computing v3.0*". Available on: <https://cloudsecurityalliance.org/download/security-guidance-for-critical-areas-of-focus-in-cloud-computing-v3/>
- [110] J M Blackledge, S Bezobrazov, P Tobin and F Zamora, *Cryptography using Evolutionary Computing*, Proc. IET ISSC2013, Letterkenny, Ireland, June, 2013
- [111] www.random.org
- [112] [www. Nutonian.com/products/eureka](http://www.Nutonian.com/products/eureka)
- [113] "An Overview of Cryptography", *Garykessler.net*, 2016. [Online]. Available: <http://www.garykessler.net/library/crypto.html#keylen>. [Accessed: 10- Aug- 2016].
- [114] Zorz, M. (2015). The importance of encryption and key management for security practitioners - Help Net Security. [online] Help Net Security. Available at: <https://www.helpnetsecurity.com/2015/05/21/the-importance-of-encryption-and-keymanagement-for-security-practitioners> [Accessed 8 Nov. 2016].
- [115] Helm, S. (2014). Cryptographic Keys: Why is Key Security So Important? The Art of Data Protection. [online] Data-protection.safenet- inc.com. Available at: <http://data-protection.safenet-inc.com/2014/07/securing-the-breach-part-5-cryptographic-keys-why-is-key-security-so-important/>
- [116] V. Sagar and K. Kumar, "A Symmetric Key Cryptographic Algorithm Using Counter Propagation Network (CPN)", in *International Conference on Information and Communication Technology for Competitive Strategies*, New York, 2014, pp. 1-6.
- [117] Negi, A., Singh, M. and Kumar, S. (2015). An Efficient Security Framework Design for Cloud Computing using Artificial Neural Networks. *International Journal of Computer Applications*, 129(4), pp.17-21.
- [118] P.Beño. The simulation of the angle velocity using NN- thesis. MTF STU, Trnava 2004.

- [119] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig and J. Wernsing, "CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy", *Microsoft Research*, 2016. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/cryptonets-applying-neural-networks-to-encrypted-data-with-high-throughput-and-accuracy/>. [Accessed: 09- Nov- 2016].
- [120] P. Xie, M. Bilenko, T. Finley, R. Gilad-Bachrach, K. Lauter and M. Naehrig, "Crypto-Nets: Neural Networks over Encrypted Data", *Arxiv.org*, 2016. [Online]. Available: <https://arxiv.org/abs/1412.6181>. [Accessed: 09- Nov- 2016].
- [121] Negi, A., Singh, M. and Kumar, S. (2015). An Efficient Security Framework Design for Cloud Computing using Artificial Neural Networks. *International Journal of Computer Applications*, 129(4), pp.17-21.
- [122] M. Björkqvist, C. Cachin, R. Haas, X.-Y. Hu, A. Kurmus, R. Pawlitzek, and M. Vukolić, "Design and implementation of a key-lifecycle management system," in *Financial Cryptography and Data Security*. Springer, 2010, pp. 160–174.
- [123] W. Li, K. Xue, Y. Xue and J. Hong, "TMACS: A Robust and Verifiable Threshold Multi-Authority Access Control System in Public Cloud Storage", *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 5, pp. 1484-1496, 2016.
- [124] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Future Generation Computer Systems*, vol. 28, no. 3, pp. 583–592, 2012.
- [125] H. Krawczyk, "Secret Sharing Made Short," in *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, 1993.
- [126] "Secret sharing", *En.wikipedia.org*, 2016. [online]. Available: https://en.wikipedia.org/wiki/Secret_sharing. [Accessed: 21-Nov-2016].
- [127] E. Ukwandu, J. Buchanan, L. Fan, G. Russel, and O. Lo, "RESCUE: Resilient Secret Sharing Cloud-based Architecture", in *2015 IEEE Trustcom/BigDataSE/ISPA*, 2015, pp. 872-878.
- [128] "Lagrange's Interpolation Formula", *Mat.iitm.ac.in*, 2016. [Online]. Available: https://mat.iitm.ac.in/home/sryedida/public_html/caimna/interpolation/lagrange.html. [Accessed: 05- Oct- 2016].
- [129] Rao, C., Rodi, P., Palande, A., & Bhusari, V. (2015). SEDAS: A Self Destructing Data System Based on Shamir's Secret Sharing Algorithm. *International Journal of Scientific Research and Education*, 3(03).
- [130] www.waikato.ac.nz/ml/weka
- [131] KVM Driver — OpenNebula 4.14.2 documentation", *Docs.opennebula.org*, 2016. [Online]. Available: <http://docs.opennebula.org/4.14/administration/virtualization/kvmg.html#requirements>. [Accessed: 13- Dec- 2016].

Appendix A

A1 – Turnitin similarity report for the dissertation

Turnitin - Mozilla Firefox
https://turnitin.com/newreport.asp?i=98,1446682273422&svr=01&lang=en_us&oid=739565512&ipbd=2&ft=1

preferences

turnitin
Originality Report
Document Viewer

Processed on: 01-Dec-2016 12:10 AM CAT
ID: 739565512
Word Count: 20241
Submitted: 3

Draft

Similarity by Source	
Similarity Index	8%
Internet Sources:	5%
Publications:	4%
Student Papers:	3%

exclude quoted exclude bibliography exclude small matches mode: show highest matches together

Abstract Cloud computing brings flexible, scalable and cost effective services. It is a computing paradigm whose services are driven by the concept of virtualization and multi-tenancy. These concepts bring various attractive benefits to the cloud. Among the benefits is reduction in capital costs, pay-per-use model, enormous storage capacity etc. However, there are overwhelming concerns over data confidentiality on the cloud. These concerns arise from various attacks that are directed towards compromising data confidentiality in virtual machines (VMs). The attacks may include inter-VM and VM sprawls. Moreover, weaknesses or lack of data encryption make such attacks to thrive. Hence, this dissertation presents a novel client-side cryptosystem derived from evolutionary computing concepts. The proposed solution makes use of chaotic random noise to generate a fitness function. The fitness function is used to generate strong symmetric keys. The strength of the encryption key is derived from the chaotic and randomness properties of the input noise. Such properties increase the strength of the key without necessarily increasing its length. However, having the strongest key does not guarantee confidentiality if the key management system is flawed. For example, encryption has little value if key management processes are not vigorously enforced. Hence, one of the challenges of cloud-based encryption is key management. Therefore, this dissertation also makes an attempt to address the prevalent key management problem. It uses a counter propagation neural network (CPNN) to perform key provision and revocation. Neural networks are used to design ciphers. Using both supervised and unsupervised machine learning processes, the solution incorporates a CPNN to learn a crypto key. Using this technique there is no need for users to store or retain a key which could be compromised. Furthermore, in a multi-tenant and distributed environment such as the cloud, data can be shared among multiple cloud users or even systems. Based on Shamir's secret sharing algorithm, this research proposes a secret sharing scheme to ensure a seamless and convenient sharing environment. Keywords – Cloud computing, Security, Encryption, inter-VM attacks, Virtualization, VM sprawl, Neural Network, Shamir Secret Sharing Chapter 1 - Introduction In this dissertation, a cryptographic system is proposed. This system is called CloudCrypt. The system aims

to address some of the security concerns in 4

the cloud. The focus is on providing data confidentiality guarantees. This chapter discusses the dissertation's fundamental

- 1% match (Internet from 07-Oct-2014)
<http://ntnu.diva-portal.org>
- < 1% match (Internet from 16-Nov-2016)
https://en.wikipedia.org/wiki/Secret_sharing
- < 1% match (Internet from 27-Sep-2016)
<https://ar.scnbd.com/doc/287048738/Essentials-of-Cloud-Computing>
- < 1% match (Internet from 23-Nov-2014)
<http://wingsman2.mine.nu>
- < 1% match (Internet from 01-Nov-2014)
<http://www-public.it-sudparis.eu>
- < 1% match (publications)
[Security Privacy and Trust in Cloud Systems, 2014.](#)
- < 1% match (publications)
[Buchanan, William, David Lanc, Elochukwu Ukwandu, Lu Fan, and Gordon and. "The Future Internet: A World of Secret Shares".](#)

A2 – Abstracts of published papers

Two publications resulted from this dissertation. Below are the abstracts for each. The full paper (i.e. paper 1) can be found on the 2016 SATNAC proceedings via this link: <http://www.satnac.org.za/proceedings/2016/SATNAC%202016%20Proceedings%20Final.pdf>.

The full papers are also on the DVD accompanying this dissertation.

Title: Evolutionary Neural Crypto-System for Cloud-bound Data

Abstract— The concept of multi-tenancy and virtualization brings various attractive benefits to cloud computing. However, concerns on data confidentiality are overwhelming on the cloud. These concerns arise from various attacks that are directed towards compromising data

confidentiality. The attacks may include inter-VM and VM sprawl. These attacks exploit the vulnerabilities of the hypervisor. Moreover, weaknesses or lack of data or VM encryption make such attacks to thrive. Hence, this paper presents a novel light-weight but strong user-side cryptosystem derived from evolutionary computing. The proposed solution makes use of chaotic random noise such as atmospheric noise to generate a fitness function. The fitness function is used to generate strong symmetric encryption keys. The strength of the encryption key is derived from the chaotic and randomness properties of the input noise. Such properties increase the strength of the key without necessarily increasing its length. However, having the strongest key does not guarantee confidentiality of data if the key management system is flawed. Hence, the proposed solution further addresses the prevalent key management problem by using a counter propagation neural network to perform key provision and revocation. Using this technique there is no need for users to keep or retain a key which could be compromised. The results show that our solution is effective and efficient.

Keywords— Cloud computing, Security, Encryption, inter-VM attacks, Virtualization, VM sprawl, Neural Network.

The following abstract was presented and published in the proceedings of the University of KwaZulu-Natal (UKZN), College of Agriculture, Engineering and Science (CAES), 2016 postgraduate research and innovation day on Nov 29th 2016. This publication's presentation was awarded a second prize in the school of Statistics, Mathematics and Computer Science (SMCS) on Nov 29th 2016.

Title: Cloud Multi-Party Secret Sharing Crypto-System

Abstract—Numerous features come with cloud computing. Among them is inexpensive cloud storage. Cloud storages attract both individuals and organizations to store confidential data on the cloud. However, the cloud is plagued with various security concerns. Various attacks are launched to compromise the confidentiality of cloud hosted data in virtual machines (VMs). These attacks include inter-VM, malicious VM sprawls, brute force and insider attacks [1]. Consequently, confidential data is at a risk from such attacks. Hence, data confidentiality is a major issue on the cloud. Encryption offers a solution to data confidentiality breaches. This can be achieved by encrypting data before it can be uploaded to the cloud.

Often, data owners are at the mercy of cloud service providers (CSPs) to protect their data against attacks. Data owners normally entrust their confidential data to CSPs. The assumption is that CSPs ensure that such data remains confidential. In most cases, this is not true. For example, a rogue employee working for a CSP may launch an insider attack and gain access to the data. Relying on a CSP's encryption algorithm and encryption keys cannot be trusted to provide the right confidentiality guarantees. In order to provide confidentiality guarantees, this paper proposes a user-side and secret sharing encryption scheme with self-destructing capabilities.

The proposed model uses chaotic random noise. The noise is used as input into a cloud-based Eureka system to produce fitness functions. The model chooses a fitness function which best represents the input noise. The chosen fitness function is used to generate encryption keys. The strength of the keys lies on the chaotic and random properties of the input noise, not necessarily the length of the key.

The proposed crypto scheme also addresses the prevalent key management problem. There is no need to store the encryption keys. A counter propagation neural network (CPNN) is used to learn the encryption key from the corresponding cipher-text. The key is discarded after encryption. Should a user want to share data, sharing is done based on the Shamir secret sharing algorithm [2]. The key is re-generated and split into shares. The shares are sent to the other parties in the scheme. A pre-set decryption time is set. If the decryption time elapses and the shared data is not decrypted, the data will not decrypt afterwards.

Keywords – Cloud computing, confidentiality, secret sharing.

References:

- [1] Mosola N.N., Dlamini M.T., Eloff J.H.P., Eloff M.M. (2016). *Evolutionary Neural Crypto-System for Cloud-bound Data*. Southern Africa Telecommunications Networks and Applications Conference (SATNAC), George, South Africa. Pp 1-2.
- [2] Shamir A. "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.