Trusted Cloud Computing Modelling with Distributed End-User Attestable Multilayer Security

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy (PhD)

in

Department of Electronic and Computer Engineering

College of Engineering, Design and Physical Sciences

Brunel University London

by

Sule Mary-Jane

April 2016

Abstract

As cloud computing continues to gain popularity and its economies of scale continue to improve, stakeholders want to minimise the security risk, protect their data and other resources while maximising the gains of using any cloud resources and its application. It is predicted that by the end of 2017, bulk of spending on any IT infrastructure would be on cloud infrastructure and services as many critical applications – power, medical, finance among others continue to be migrated onto cloud platforms. For these sectors, the security challenges of cloud adoption continue to be of a great concern even with its benefits.

The ability to trust and measure security levels of any cloud platform is paramount in the complete adoption and use of cloud computing in many mission critical sectors. In-depth study and analysis of the trustworthiness of various cloud based platforms/systems are often limited by the complex and dynamic nature of cloud and often do not correctly foresee or practically determine the varying trust relationship between and across the cloud layers, components (schedulers), algorithms and applications especially at a large scale***.***

Tradition security and privacy controls continue to be implemented on cloud but due to its fluid and dynamic nature, research work in the area of end-user attestable trust evaluation of the cloud platform is limited. Most of the current simulation tools do not cater for modelling of Trust on scalable multi-layer cloud deployments (including workflow and infrastructure).Even as these tools continue to be implemented none has been used to cater for all the layers of the cloud platform.

This research presents a deployment of trusted computing applied in cloud computing suited for mission critical applications. It attempts to simplify the integration of trusted platform module based integrity measurement into cloud infrastructure. Using Eucalyptus cloud software on server-grade hardware, a trusted community cloud platform was deployed on the Brunel Network as presented in Chapter 3. Security is enhanced by the integration of an end-user accessible TPM integrity measurement and verification process; this guarantees trusted ownership and integrity of the uploaded data and provides additional level of trust for the cloud platform.

This research further presents a technique which allows data owners to first secure their data offline by inserting colour drops into the data using steganography. The colour drops are used to detect unauthorised modifications, verify data owner in the event the copyright of the data is in dispute and identify the path through which it was tampered with. This process ensures integrity and confidentiality of the resources.

This thesis also presents a trust model using fuzzy logic which was simulated using Simulink in Matlab and subsequently evaluated on an experimental platform deployed on the Brunel network. Using this model, end-users can determine the trust values for a cloud platform or service, as well as, classify and compare various cloud platforms. The results obtained suggest that the outputs of this research work can improve end-user confidence when selecting or consuming cloud resources with enhanced data integrity and protection.

Dedication

To my beautiful and selfless family – this is for you!!!

Declaration of Authorship

The work detailed in this thesis has not been previously submitted for a degree in this University or at any other and unless otherwise referenced, it is the author’s own work.

Acknowledgements

I would like to give all praise, glory and honour to our Lord and saviour Jesus Christ for His faithfulness all through my life and studies, He made everything possible!

I would like to express my gratitude to my Supervisor Prof Maozhen Li, who with great dedication and high standards for research he has painstakingly supervised this work, indeed I am very grateful for all your dedication and support throughout the years. I say thank you to Prof Gareth A. Taylor for his support and guidance during my research.

To my parents and my siblings – I am highly indebted! I could not have done it without you all. Thank you for all the prayers, support, encouragement and sacrifices to give me the best, God bless you all indeed.

To Sr. Mary Kenefick SMG, Fr Nicholas Schofield, Fr Stephen Wang, Richard Parker, Simon Furber, Clement, James, Christy, Titi and the Vice Chancellor University of Jos I say Thank you.

List of Figures

List of Tables

List of Nomenclature

AIDE	Advanced Intrusion Detection Environment
API	Application Program Interface
BCG	Backward Colour Generator
CC	Cluster Controller
CCTV	Closed Circuit Television
CI	Cloud Infrastructure
CLC	Cloud Controller
CoT	Chain of Trust
CSA	Cloud Security Alliance
DC	Data Colouring
DCP	Data Centre Policy
DNO	Distribution Network Operators
EBS	Elastic Block Store
EC2	Elastic Compute Cloud
ELB	Elastic Load Balancing
FCG	Forward Colour Generator
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPD	Apache Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IAM	Identity and Access Management
IDE	Intrusion Detection Environment
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention System
M2M	Machine to Machine
MIME	Multi-Purpose Internet Mail Extensions
MLSTM	Multi-Layer Security Trust Model
NC	Node Controller
NG	National Grid
NIST	National Institute of Standards and Technology
NTP	Network Time Protocol
OGS	Open Grid Systems
OpenPTS	Open Platform Trust Services
PaaS	Platform as a Service
PKI	Public Key Infrastructure
QoS	Quality of Service
RSA	Rivest-Shamir-Adleman (Cryptosystem)
S3	Simple Storage Service
SaaS/AaaS	Software as a Service / Application as a Service
SC	Storage Controller
SLA	Service Level Agreement
ssh	Secure Shell
SSL	Secure Socket Layer
TC	Trusted Computing
TCP/IP	Transmission Control Protocol / Internet Protocol
TPM	Trusted Platform Module
TSO	Transmission System Operators
UFS	User Facing Services
UTC	Coordinated Universal Time
VM	Virtual Machine

vTPM	Virtual Trusted Platform Module

List of Publications

- M.-J. Sule, M. Li, G. A. Taylor, and S. Furber, “Deploying trusted cloud computing for data intensive power system applications,” in *2015 50th International Universities Power Engineering Conference (UPEC 2015)*, 2015, pp. 1–5.
- M.-J. Sule, M. Li and G. A. Taylor, “Modeling Trust in Cloud Computing”, 10th IEEE Intl Symposium on Service Oriented Systems Engineering, 29th March – 1st April 2016.
- M.-J. Sule, M. Li and G. A. Taylor, “Modeling Trust in Cloud Computing based on Fuzzy Logic ”, (Submitted to IEEE Cloud Computing in March 2016)

Table of Contents

Chapter 1

Introduction

Cloud computing while achieving economies of scale, have made scalable computing resources widely available; however, as noted in [1]–[4] a global challenge to its full deployment and adaptation are the security and integrity of various components on the Cloud Infrastructure (CI), such as the deployed instances/virtual machines, and the hosted data among others. The National Institute of Standards and Technology (NIST) [5] defines CI as the combination of both the hardware and software that enable cloud computing.

With cloud computing, there are three fundamental service models known as the delivery models [1], [2], [5], [6] , which are *Infrastructure as a Service (IaaS)*, *Platform as a Service (PaaS)* and *Software/Application as a Service (SaaS/AaaS*).

Infrastructure as a Service (IaaS) which provides access to virtualised hardware services like servers, disk space and memory to clients for software deployment/development. The IaaS service enables the deployment of the instances/virtual machines that are the “work-horses” in cloud computing.

Platform as a Service (PaaS) – provides a platform for clients to deploy/develop application but clients have no control of the underlying cloud infrastructure.

Software/Application as a Service (SaaS/AaaS) – provides client access to software applications running on the cloud infrastructure.

The above service models are quite distinct from the following deployment models [1], [2], [5], [6]:

Private Cloud – this deployment model allows only users of a single organisation access to resources on privately owned and dedicated cloud infrastructure.

Public Cloud – this deployment model provides the general public with access to resources on shared (common) cloud infrastructure.

Community Cloud – this deployment model allows users from different organisations with the same mission to share and access cloud infrastructure resources.

Hybrid Cloud – this deployment model combines two or more cloud models together in a manner that allows users of both models to seamlessly scale resources between the combined models.

Figure 1.1 by [7] provides a visual overview of Cloud Computing and shows hierarchical relationship between cloud service modes and deployment models.

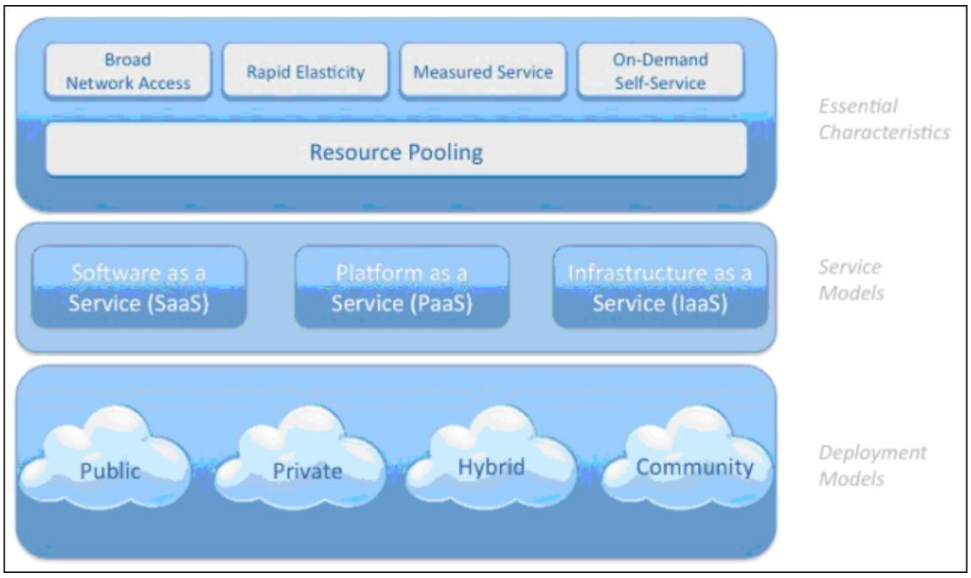


Figure 1.1: Visual Model of NIST working definition of Cloud Computing (Source: [7])

Cloud Computing allows remote processing using multiple varied instances and multiple computers running at the same time. The large, multi-tenant and distributed nature of cloud systems means they are relatively easy targets for intruders and security can easily be compromised if care is not taken [4], [6]. With the centralised nature of most cloud resources, data owners and end-users lack direct control on cloud remote resources and the cloud provider's perceived complete control over hosted data on its infrastructure is beginning to make data owners and end-users care about how the cloud provider handles and uses their hosted data or other resources while on the CI.

As shown in Figure 1.2 from [6], a cloud deployment should have five distinct roles, these are the cloud provider, cloud consumer, cloud auditor, cloud broker and cloud carrier. For the purpose of this research work, the cloud consumer and cloud provider roles have been further subdivided as follows:

End-User: usually the end-user is the ultimate consumer of the reliable and available cloud services and only pays for consumed services.

Data Owner: a data owner is one who has the legal rights over the data and is usually also accountable for it. In some cases, the end-user is distinct from the data-owner or the data owner could also be an end-user.

Cloud Service Provider: is usually a provider of one or more cloud services to any cloud consumer. The cloud service provider acquires, manages and maintains the cloud infrastructure and services over the network and provides these services through virtualisation, resource pooling.

Cloud Application Vendor: usually sells the on-demand cloud applications to end-user. It could also be distinct from a service provider or a service provider could also be an application vendor.

Cloud Tool Provider: provides cloud support and manageability tools to do accounting, monitoring and usage reporting.

While, the roles of end-user and data-owner are associated with cloud consumer, the latter three (service provider, application vendor and tool provider) are associated with the cloud provider. The cloud auditor whose role includes security audit is expected to be independent for transparency purposes.

From Figure 1.2, it is clear that within the architecture, cloud consumers (end-users and data owners) only have an "indirect" way (via the cloud auditor) of knowing the security status of the cloud infrastructure and how their data is processed. Basically, some commercial cloud auditors are listed via initiatives such as the CSA Security Trust and Assurance Registry (STAR)[8]. The open certification framework used by STAR recommends three distinct levels of certification, which are self-assessment, 3rd – party assessment and continuous monitoring based certification [9]. Most of the security information available from STAR seems to be out-dated self-assessments (by cloud providers) and provide little or no useful information about wire-level security of individual services.

A cloud user or data owner wants to be confident and trust that the cloud resources they are accessing is secured, and would be available, while its data integrity is tamper proof and not compromised [4], [5], [10]. In addition, a data owner wants to know that the originality of his or her data can be ascertained in the case of any misunderstanding, theft or tampering.

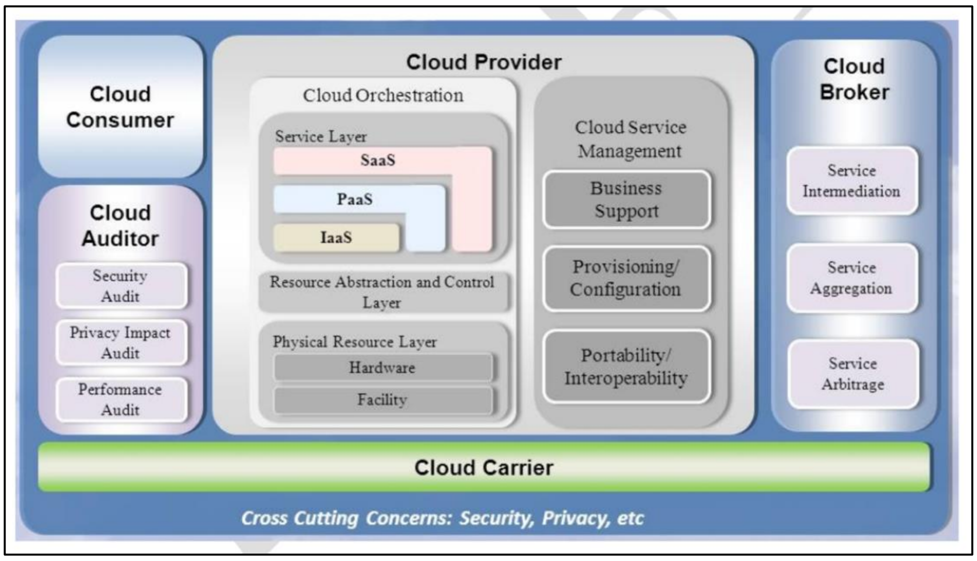


Figure 1.2: NIST Cloud Computing Reference Architecture (Source: [6])

Improving the trust relationship between cloud users, data owners and the cloud providers for wider adaption of cloud services is a global challenge not adequately addressed by existing approaches where end-users have very limited ability to directly measure or attest the security and integrity of any single cloud service. This research work addresses this challenge through the investigation of multiple security mechanisms across the cloud layers. The results obtained allows a data owner / cloud user to measure the security of the cloud infrastructure at each layer (Infrastructure, Platform and Software), while also protecting their data through data-colouring, which ensures that a data owner can highlight or trace data-loss-path in the case of theft or data compromise and can also prove ownership, copyright or originality of owned data.

1.1 Motivations

Even as cloud computing improves computing and economies of scale, its security plays a critical part in its full deployment and adoption by the end-users [11]–[14]. Therefore, the overall motivation of this research has been to provide a means for users to attest to and verify the security status and integrity of a cloud infrastructure thereby deciding whether to trust the platform with sensitive information and processes. This would improve the trust relationship between cloud users, data owners and cloud providers. Three specific motivations that drove the research were:

1. To provide enhanced distributed Virtual Machine (VM) security against man-in-the-middle attack and Denial of Service (DoS)
2. To improve end-user's ability to measure / attest cloud security and data integrity
3. To improve trust relationship between end-users and cloud providers

1.2 Methodology

The research method has been experimental with modelling and simulation. The approach demonstrated the feasibility of using the solution to solve a global challenge in cloud computing security as it relates to cloud user’s ability to attest and verify CI security status while also checking the integrity of its hosted data.

An experimental Eucalyptus cloud in a box community platform was deployed on the Brunel Network on an Intel server which was used to evaluate the research work presented in this thesis. The specifications of the platform and the instances that were deployed on it are presented in Chapter 3.

The research answers the following questions:

- How can “end-user” multilayer security on cloud infrastructure be achieved using Trusted Computing (TC), file integrity checking/Intrusion Detection System (IDS) and Data Coloring (DC)?

- How can TC, IDS and DC be combined to enhance distributed CI security (against man-in-the-middle attacks and denial of service)? How can TC, IDS and DC be combined to establish the trust status of the CI?

- What is the performance impact on real world applications of a multilayer security based on TC, IDS and Data Coloring?

The research work includes a detailed literature review to provide a clear understanding of the security and trust challenges encountered by end-users and cloud providers and also the deployment of a cloud platform that will be used to represent readily available resource allocation and utilisation within a community cloud system. This also helps in understanding potential security issues and how they could be addressed.

1.3 Major Contributions to Knowledge

The major contributions of this research work are summarised as follows:

- The thesis presents a “patch-free” trusted cloud deployment integrated with end-user accessible Trusted Platform Module (TPM) integrity measurement and verification suited for mission critical applications. This deployment enhances overall security by the inclusion of an instance-level file or directory integrity checker for selected files and directories. This combined approach guarantees confidentiality and integrity at instance-level while at the same time providing end-users with the ability to attest/verify on-demand, the integrity and state of the underlying platform/service. A working prototype based on the Eucalyptus cloud software was deployed on the Brunel network and made available to industry partners in the energy sector to test power system application like Cimphony during mission critical usage, in a context that guarantees ownership and integrity of the uploaded data as each operator is provided with a diversified model based on the data sharing needs.
- Implements a shell-script for data colouring which secures the data that is being processed or stored on the cloud platforms. The implementation is based on establishing and using concatenated fingerprints for watermarking through steganography. Using this technique, cloud and data owners are able to secure their data offline before uploading it onto any cloud platform. The fingerprint based “colours” are used to detect unauthorised modifications and suggest or highlight possible path of data loss or theft. The implementation was evaluated on the deployed experimental cloud platform.
- This thesis presents a multi-layer security model based on fuzzy logic that combines the application of multiple traditional security and privacy mechanisms/controls across the different layers of the cloud platform in the determination of trust values. This model is resilient to failures of individual security mechanisms and allows continuous discrete testing/probing of cloud platforms. Unlike other models that converge to allow secure, the model from

this research shows that a cloud platform always include some inherent risk. It provides the end-user with a tool to check service reliability, accountability and non-repudiation.

1.4 Structure of the Thesis

The description below outlines the overall structure of the thesis and the purpose of each chapter.

Chapter 2 presents aspects of Cloud Computing Security, discusses the implementations of security mechanisms at various cloud layers, and other relevant studies in the context of cloud security research.

Chapter 3 provides further details about trusted computing as a security mechanism for the IaaS layer, where integrity measurements are taken and stored on a trusted platform module (TPM) with the aim of verifying the measurements. This chapter also, discusses the use of an independent intrusion detection system (IDS) for instances at the PaaS layer. It also presents a Eucalyptus community cloud deployment test-bed on the Brunel network.

Chapter 4 discusses the implementation of data colouring based on steganography for data-protection. This is shown to provide a traceable path in the case of data theft, tamper or proof of data originality for the SaaS layer.

Chapter 5 provides details about the matlab based simulation and testing of the Multi-Layer Security Trust Model (MLSTM) that uses Fuzzy-Logic Maths to provide users with information about the state of the cloud infrastructure. This chapter also presents results of MLSTM test carried out on the deployed eucalyptus test-bed.

Chapter 6 concludes the research and proposes some future work for further research improvement in the field of cloud computing security and trust.

Chapter 2

Literature Review

As cloud computing deployment continues to provide savings on IT investment, its popularity is rising even among sensitive mission-critical sectors like the health and energy sectors. Cloud computing can provide timely, cost effective scalable deployment of ICT services and infrastructure to these sectors. However, the availability, confidentiality and integrity of data are paramount and are of great concern to these sectors and other cloud users. This chapter provides an overview of cloud computing security; it introduces the techniques employed presently in securing the cloud and then introduces some security mechanisms that can used to enforce data integrity for cloud processing.

2.1 Computing Security

Every automated system needs to be protected to preserve its integrity, availability and confidentiality [7] . In any field of IT and computing, computer security involves the protection and fortification of computing resources, data integrity, limiting unauthorised activities, keeping malicious users out, and of paramount importance also maintaining and enforcing data confidentiality. For any system to satisfy the security requirement its resources have to be available, timely, consistent, not exposed to malicious destruction and would not be disclosed or accessed by unauthorised users.

As reliance on computer technology globally continues to increase, measures to secure these resources remain of utmost importance. Various measures to maintain confidentiality minimise failure or loss continues to be put in place by researchers and industries. Implementing security has never been an easy task and most times security of these resources are sometimes an afterthought which makes implementation even more difficult.

Some of the common threats to security in computing and information systems may be categorised into the following [7]:

- Operator's error: this error which may be caused by any user who has the privilege of creating or modifying data. Errors affect the fundamental integrity of the data and compromise any processing that may depend on the data. Depending on the severity of these errors, they could be a threat or may cause the system to be vulnerable [15].

- Social engineering: Mitnick and Simon explained in [16] why even the strongest enforced firewalls and encryption protocols may not be sufficient enough to stop a would-be attacker from having unauthorised access to a computer system. Social engineering involves determining (sometimes through blackmail, bribery, deceit) the needed or authorised information such as passwords or technical “workings” of the system from an insider or getting it through their social interactions including trash items.
- Malicious Hackers: Some people have a sophisticated knowledge of the computer systems and may exploit these errors or privileged know-how of these systems to gain unauthorised access to the systems [15]. Even though initially their expert knowledge of the computer systems was to positively use or maintain the system beyond other ordinary users, it becomes malicious when used negatively to the detriment of the systems and other users. Though the group of people who are called hackers are further divided into two – white hats and black hats. White hats use their expertise to assist developers while the black hats use their expertise to inflict harm to a targeted system.
- Fire and Natural Disasters: Fire and natural disasters such as earthquakes can highly affect the availability of computing resources even when the probability of such occurrence is low. While, it is impossible to predict or control such events, they can be catered for through adequate contingency planning [7].
- Espionage: In computing, espionage involves the collection and acquisition of privileged or confidential information by an authorised privilege user. The collected data or information may be sold or bought, circulated or used to directly aid a competitor or another organisation. Espionage may be impossible to control especially when it is perpetuated by government agencies in response to terrorism or with the collaboration of a compromised insider with privileged access [17].

- Physical: there could be an act of sabotage by physically attacking or damaging computer systems or resources. A physical attack for instance on data storage would render the data completely or partly unreadable.

These treats can be broadly categorised as accidental disclosures where any component of the computer systems may fail thereby exposing it to attacks, deliberate penetration where there is deliberate effort to acquire information about or contained in the system and physical attack where there is an attack on the physical system or the environment where these systems are placed or hosted physically.

Cloud computing security is not different from traditional IT and computing security as previously described, instead security in cloud computing is more complex as processing on cloud platforms involves virtualisation, multi-tenancy and almost every interaction is carried out over the network. From the end-user's perspective, clouds are not confined to a single or exact physical location and many-a-times, it is impossible to specify the exact location of where one's data is stored or processed [18]. Many of the security threats and/or vulnerabilities affecting cloud computing are in reality linked to the described computing security categories/topics of user (operator) error, malicious (ordinary) hackers, social engineering, natural disasters, terrorism and in recent times even espionage as in the case of the US government and Edward Snowden [19].

The risk of computing resources being compromised due to security threats are minimised through the enforcements of adequate defence mechanisms or having security measures put in place to protect computing resources and ensure they remain safe even when accessed by nefarious or malicious users. Some examples of these computing security measures include:

- Access control: These are the measures put in place to safe guard against theft and deny unauthorised access to the computer systems and data resources. Techniques involve in limiting or granting access include passwords, user authentication, access control, intruder alarms, physical barriers among others. Users are encouraged to have strong passwords that meet the password requirement and it's hard to guess by malicious attackers. Due to advances in computing technology, the resources are more vulnerable to attacks now more than ever.

- Physical security involves limiting physical access to computing resources. This may take the form of providing special access for privileged users.
- Organisational control: This could include security clearance and operating procedures for employees of an organisation.
- Network and Internet security: The inception of the Internet and now cloud computing has broadened the scope of computing security. Firewall which regulates remote network communication of computer systems is one of the most effective forms of protection on the Internet, network or cloud. Antivirus software is another form of protection for computer systems that connect to any form of network, internet or the cloud. Users are also encouraged to have strong password which meets the password requirement and it’s hard to guess by malicious attackers.
- Natural Disasters: While disasters may be impossible to avoid, their impact may be mitigated through effective planning in order to avoid the unavailability or loss of data and other computing resources. A disaster recovery plan should include keeping regular back-up or copies of critical data/systems as well as the physical replication or duplication of resources/data both on-site and at a remote safe location.
- Policies and legislation: Due to the changing nature of security attacks, it is usually necessary to have suitable or specific policies that ensure computing resources are regularly validated for security holes, security mechanisms are adequately patched or updated to address new threats or variations of existing ones. For example, a Data Centre Policy is typically used to safe-guard physical access to computing resources, high-availability of critical resources (including energy) as well as multiple levels of backups of critical databases. In many countries, the handling of data about persons is governed by suitable legislation aimed at protecting the privacy and rights of individuals against abuse while allowing the correct use of such data. Sometimes, legislation may go further to include items such as mandatory logging of access, mandatory periodic changing of access credentials and specifying limitations on storage or transfer of data outside determined boundaries often with penalties.

2.1.1 Physical Security and Trusted Platform Module

Physical security is a measure that is designed to deny unauthorised access to the data centre environment, equipment and resources. Physical security in computing is the foundation of all other security policies. As such, the improper physical security of any computing system carries grave consequences for its overall security regardless of the comprehensiveness of all other security mechanisms that were put in place. For example, regardless of how much is spent on other part of security like intrusion detection, anti-virus and others, the confidentiality of resources are in doubt as long as the critical (physical) infrastructure is not adequately protected even though physical security measures vary according to organisational structure and needs.

Beginning with the site or premises on which these resources are kept or processed some measures need to be enforced. In the UK, the storage or usage of government resources classified above “Restricted” is only possible at locations that are listed as satisfying security requirements; they are called the “list X” facility [6]. Workers are not permitted to use their personal computers or other IT equipment that are located in a “list X” facility as these are thought to be less secure than on a “list X” site [20].

Other physical measures include protection of the server room, laptops and desktops. While the systems maybe located and secure in a room there must be measures put in place to make sure only authorised users access the room or resources. These include [21]:

- Biometrics: This involves the use of matching physical characteristic(s) of the individual such as retina, fingerprint or facial recognition to provide access to the secured room or resources.
- Access Cards: an access card is linked to a specific user and is expected to be in the possession of that user at all times. Access cards are non-transferable, are not to be shared and the loss of an access card is expected to be reported immediately.
- User Awareness: Being aware of who is authorised to access certain locations allow users to confront any unauthorised user that may be in a restricted area.

- Locks: Special cables and locks (for example Kensington locks) are used to physically connect a system to a desk, this makes it impossible to move devices or steal them.
- OS Hardening: Refers to the process of disabling unused services, de-installation of un-needed application/software and sometimes disabling of USB ports and CD drives to discourage copying of the resources off the system.

Physical security is further enhanced when access to resources by users is based on a ring approach.

It is clear that all the measures discussed above are aimed at limiting un-authorised access or theft of computing resources, but they provide little or no information about the overall integrity of a system, that is, if it has been tampered with by possibly an insider with malicious intent. An alliance formed by several industry partners namely: Microsoft, Intel, IBM, HP and AMD developed the TC industry standard for a more secure computing platform.

In TC, the goal is to provide a seemingly tamper proof computing platform with the ability to manage digital rights, detect/fight software piracy and even facilitate the rental access to software (and/data). The design of TC provides for all computing platforms to have a monitoring and reporting component (with some storage), known as TPM, implemented directly on the motherboard, that coordinates with TC enabled operating system kernel and other TC applications for protection enforcement and possibly direct access to a registry of online security servers maintained by hardware and software vendors

Since 2006, new computers were sold with a built-in TPM chip, that independently secures computing hardware using integrated cryptographic key(s) that may be integrated into various computing operations such as firmware (BIOS) loader, system boot-loading, O.S. kernel and application start-up to monitor and store cryptographic (almost unforgeable) hash data about the integrity and trustworthy state of the computing chain and/or computer platform.

That is the data stored in the TPM may be used to determine if individual components of a platform behave as intended. The TC process of attestation (verification) involves comparing the TPM data against a prior set of hash key summary for certifying the tamper-proof state of hardware and software configuration. The prior set of hash keys may be stored off-line and periodically used even by third-parties and over suitable remote connections to verify that the software has not been tampered with or changed [22]. The TPM device may also be used to securely store third-party artifacts such as passwords, certificates and encryption keys designed to uniquely identify/authenticate individual computer platform(s) even for remote attestation and/or authentication by authorised third-parties.

The use of a TPM backed TC authentication and attestation process provides a safer computing environment that may be used to enhance the level of protection offered to mission critical applications [23].

Figure 2.1 shows the functional block components of a TPM device. The TPM device may also be used in other devices like mobile phones and other network equipment as it would ensure that the critical information (stored on the device) is better protected from external software attack and tampering.

It is important to note that the current implementation of TPM devices cannot control the software running on the computer system, as it only stores the cryptographic hash measurements taken during a chained start-up (pre-run time configurations).

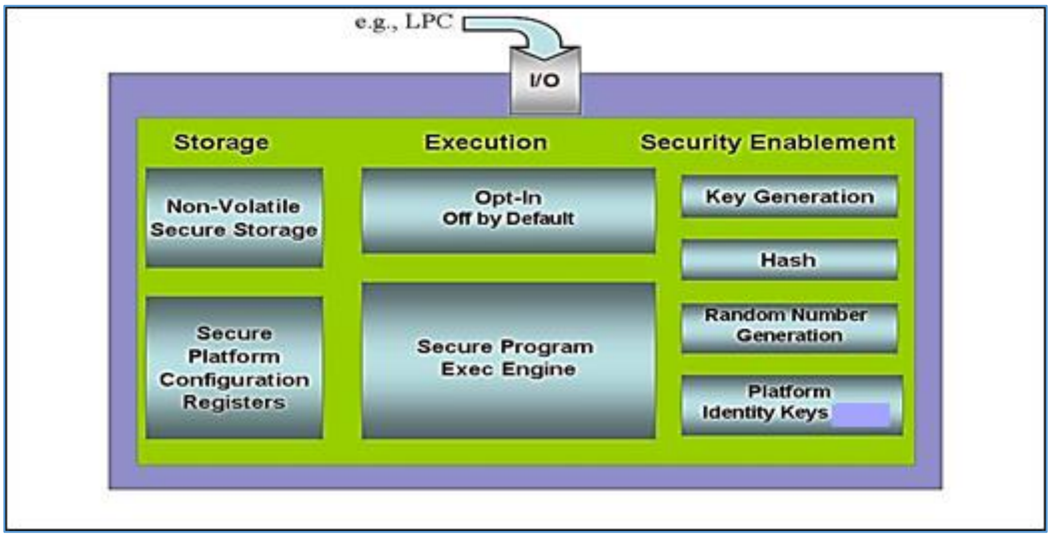


Figure 2.1 : TPM components (source [23])

For hashing, TPM uses a form of asymmetric key cryptography called RSA that makes it infeasible to modify data without changing the hash key; this guarantees integrity and provides a form of protection for the stored data. The TPM provides a secure root of trust (starting from the BIOS/firmware) for both reporting and storage and each separate measurement of a component sub-system (of the computing platform) is a single/independent transaction that generates a separate hash key.

It is note-worthy that, in a context of machine-to-machine (M2M) communication, trust could be based on confirmed identity as well as the verified expected (predictable) reliability of each party.

2.1.2 Intrusion Detection and Prevention

The rise in the number of security issues experienced by users has been attributed to the explosive deployment and processing of computing resources over network [24]. NIST in [25] describes intrusion as an attempt to compromise confidentiality, integrity and availability. Intrusion detection is the process of monitoring the events and/or changes on a computing platform and analysing them for any sign of intrusion. Software or hardware systems that automate the process of monitoring occurring events and/or changes in a computer system and analysing these systems for any security problems are called intrusion detection systems and they are vital in computing systems security as network based attacks have greatly increased in number in recent times [25].

Intrusion detection systems can detect attacks and other security violations that result from breaches or failures by other security measures. In large enterprises, intrusion detection systems serve as quality control for security implementation and of course these systems provide information about intrusions and attack that may have taken place.

Sometimes once a system has been installed, the administrators rarely go back to update patches and other improvements either due to lack of time or knowledge, the IDS detects when an attacker exploits these flaws and penetrates the systems.

Security attacks on a computer system may occur in some predictable manner- for example, at first the attacker may use probes to examine (or scan) the system [25] exhaustively and choose an attack vector capable of inflicting the highest possible damage or threat. A monitoring IDS analysing traffic in near real-time would trigger alerts to the necessary entity for further action, while, an intelligent IDS capable of operating in prevention mode would identify the probes as suspicious and appropriate actions taken to limit further probes/attacks by introducing evasive behaviours (time-limited blacklisting of all passwords from the offender, during the time period even the right password is rejected), or outright firewalling (blocking or preventing total communication with) the offending parties.

There are two main strategic approaches to implementing intrusion detection as reported in [25] , the host target co-location approach was used in the early days of mainframes, where most IDS were installed on the target system, this was done then as having another computing system to run IDS was expensive. The other, host target separation approach was introduced with the advent of personal computers and as mainframe systems became cheaper. In host target separation, the IDS were now installed on separate systems from the target host and this approach also improved security of the systems as the existence of the IDS is hidden from the attackers. IDS may also be classified as network based, where the IDS detect attacks by capturing and analysing network packets arriving at the target system or host based, where the IDS detects attacks by information collected within the target system.

An intrusion detection system is composed of the following fundamental components: Information source(s): which is used to determine if any intrusion has taken place. The analysis/decision engine: that analyses the events and extracts security related information from it and the response engine: which is the set of action(s) the system takes once it detects any intrusion. The IDS components are coordinated to provide security related to accountability, response and control.

Accountability is provided by associating the intrusion activity or event to the entities responsible for initiating such activity, however, the accuracy of such association may be severely limited when the attacker uses a distributed network of compromised hosts for scanning or possibly uses forged identities. It is important that the system enforces a strong identification and authentication mechanism to further make it more difficult for the attacker to use a forged identity.

Once a malicious probe, scan or attack is identified and classified by the IDS, appropriate response action is taken even when it originates from an authorised user, typically through some mechanism to control several elements of the computing system such as the authentication modules or ip firewall.

2.2 Cloud Computing

Cloud computing has made scalable computing resources more widely available and has also achieved economies of scale; however, as noted in [2], [3], [5], [26] a global challenge to its full deployment and adaptation are the security and integrity of various components such as the CI and the deployed instances/virtual machines.

Virtual machines (VMs) are software machines created to emulate or imitate a computer hardware system; they operate based on the function and capabilities of real computer hardware which also host the VMs. Instances are virtual machines or servers that are created from the cloud infrastructure.

Cloud computing is an integration of different computing technologies; therefore its security challenges are not entirely “new.” Cloud computing security and trust challenges are only more noticeable because transactions are most times carried out with clients or providers whom the end-users have never met or do not see them physically [27] [26] [28].

There are three popular layers in cloud computing which are called the delivery models IaaS, PaaS and SaaS. IaaS provides the cloud user with virtualised shared storage and computing resources, which include CPU, disk space and memory. The IaaS delivery model enables the deployment of the instances/virtual machines that are the “work-horses” in Cloud computing [2], [26] and [5]. PaaS provides the cloud user with a platform for testing any application. While, SaaS provides the user with software and applications without the user having to worry about licenses and installation, this usually runs on the cloud provider’s infrastructure [29].

Conceptually, one can refer to a cloud stack as similar to a TCP/IP stack [30], which can then be represented thus: Physical → Virtualization → IaaS → PaaS → SaaS. Based on the user’s perspective, IaaS would require low-level hardware to be virtualized and as shown in Figure 2.2, each layer requires the structure and standards of the layer below it. In the cloud service delivery model, IaaS is at the lowest level and is usually the foundation of all cloud computing. It may be stated that all cloud services fundamentally require an underlying cloud infrastructure that is the IaaS layer even when it is not directly offered as a service. The SaaS layer relies completely on the security configurations put in place by the provider. At the PaaS layer, the developer can configure further restrictions on the application or platform while at the IaaS layer the tenant can further customise the security configurations but there is no doubt that all these configurations still would have to rely on the underlying mechanisms to be completely secure [31].

Different cloud services provide different core services; Table 2.1 presents a summary of the services provided by each delivery model.

Table 2.1: Services provided by each delivery model

Services	**IaaS**	**PaaS**	**SaaS**
Networking	Yes	No	No
Storage / Disk Space	Yes	No	No
Server Hardware Resources	Yes	No	No
Application Layer	No	No	Yes
Integration	Yes	Yes	Yes
Infrastructure Management	Yes	No	No
Payment Per Utility	Yes	Yes	Yes
Resource Elasticity	Yes	Yes	Yes
Application Development	No	Yes	No
Multi-tenancy Architecture	Yes	Yes	Yes

Depending on the need of the users, cloud services may be deployed in a number of ways - *Private, Public, Community or Hybrid* cloud deployments with varying security considerations even for major issues like confidentiality, integrity (of data) and availability [5], [32], [33].

When resources are deployed within a single organisation for only that organisation even when the offices are physically at different geographic locations or data-centres, it is referred to as a private cloud. Private cloud when compared to other cloud models present the least security concern as they are only accessible via internal network and sensitive and mission critical application are being protected behind the enterprise firewall [34].

Community cloud deployment on the other hand is when organisations or communities with similar mission, needs and requirements come together to share resources. It made be managed by one or more of the organisations or by a third party, though allowing a third party manage it comes with its inherent security risk as that also means accessing it over the internet [35], [36].

The resources deployed on a public cloud are usually accessed and shared by the general public and it’s usually managed by a commercial provider. It also goes without saying that of all these deployments mentioned, public cloud is the most vulnerable to attack.

Another cloud deployment model that combines any two or more deployment models and techniques mentioned above is the hybrid cloud. Even when the deployments are combined each deployment still maintains its uniqueness.

NIST [5] defines Cloud Infrastructure (CI) as the combination of the hardware and software that enable cloud computing.

Many challenges are apparent when sharing the same platform; these include denial of service, data integrity, operating system, applications piracy and copyright issues among others. Considering that the IaaS layer involves the sharing of common computing hardware resources, common security challenges obvious on this layer would include denial of service and man in the middle attack, among others. Generally, clouds enforce security across infrastructure including the network and the platform layer through tools such as firewalls, Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS) or encrypting and securing data in transit (for example DNSSec) among others; and may involve the sharing of a common storage where data is either hosted on the platform or processed without any form of protection. Figure 2.2 also shows some of the possible security measures commonly adopted for the different cloud layers.

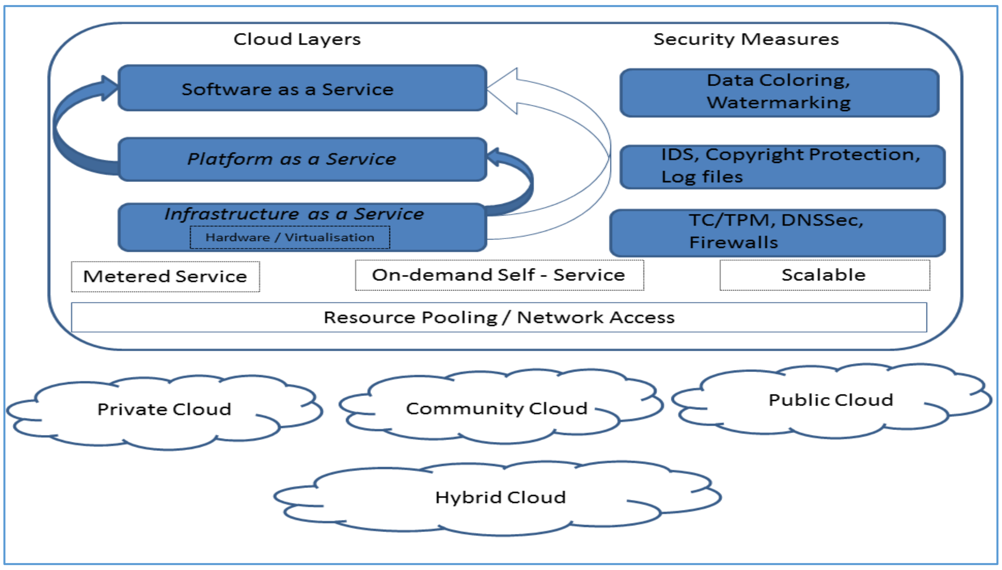

Cloud Layers
Security Measures
Software as a Service
Data Coloring, Watermarking
Platform as a Service
IDS, Copyright Protection, Log files
Infrastructure as a Service
Hardware / Virtualisation
TC/TPM, DNSSec, Firewalls
Metered Service
On-demand Self - Service
Scalable
Resource Pooling / Network Access
Private Cloud
Community Cloud
Public Cloud
Hybrid Cloud

Figure 2.2: Cloud Layers and Possible Security Measures

Mission critical services require on-time availability and reliability of scalable computing resources, which is only possible through the deployment of Cloud computing. However, more needs to be done with regard to the security and integrity of the cloud infrastructure and the virtual machines (instances) running on the "cloud". There is also a need for the secured processing of data whether on the "cloud" or in the traditional computing environment. The security of the data and applications can never be over-emphasised as mission critical sectors rely heavily on efficient and secured data [2], [3], [5], [26].

2.3 Cloud Security

For any security mechanism to be effective, it must be tailored to complement the target platform. Therefore, for any platform to be considered secure and trusted, the security mechanisms across every layer must be enforced and configured as it is the individual security mechanism on each layer that make up the overall security status of the cloud platform and therefore each layer must be properly secure as there is no "one fit all" security solution that would be applied on the platform but all the layers contribute to the security of the whole platform. Cloud computing relies heavily on remote access and the virtualization of servers and network connectivity among others. Cloud security is also of paramount importance for the full adoption of cloud by mission critical sectors and other end-users [4] [37].

In cloud computing, the trust relationship between the users and the providers is important as the deployment and control of some security mechanisms such as firewalls, installation of software patches, and appropriate use of security groups/zones among others are completely managed by providers. Within, the Open Security Framework by CSA, Cloud Auditors (not end-users) are expected to ensure adequate security compliance by providers.

Security groups or zones directly address multi-tenancy issues that may create more opportunities of misconfigurations, malicious conduct or data compromise. Securing the multi-tenancy environment also requires the proper isolation of identity verification as well as enforcing limits access to resources and locations. For cloud users they could be group based on their activity where privilege users may be grouped according to roles like administrator, ordinary user and so on and the provided with a single standard-based user login capability to allow easy, quick and authenticated cloud service.

Network or IP firewalls are used to block access to computing services/resources from untrusted persons and other nefarious computer based attacks. The deployment of firewalls follows several different concepts that influence the deployment and usage of cloud computing services. In the Closed/ Walled garden concept, end-users are restricted only to approved applications and services. Additionally, there is usually some form of security to the allowed services from unwanted users. The key disadvantage of the closed/walled garden concept is that it limits end-users from seamlessly deploying or implementing public or community clouds. The Open garden concept on the other hand allows unrestricted access to services and applications by all end-users, this concept is not suitable for storing sensitive data and can lead to the data integrity been tempered with. In the end it best to keep sensitive data behind the closed/walled garden firewalls. Additionally, some regular monitoring is required to ensure adequate safety of the sensitive data even when clouds are also deployed on the network.

The way data is stored, accessed or processed on a platform also plays a massive role in the integrity and availability of such data. Various mechanisms can be applied on a cloud platform to enforce security as applying only a particular traditional security mechanism is unsuitable for the cloud platform [38] . Data information needs to be properly segregated on the cloud platform. Encryption is one way of protecting data while it is transit or during storage. Encryption maintains data confidentiality and integrity at the expense of accessibility as only authorised users with the necessary keys can decrypt data for processing. Encryption requirement is also different from one user to another. While some may require very high restrictions, others may require only for specific data and delegate the key management to a third party. Data colouring is another mechanism for securing cloud data which, in contrast to encryption, maintains accessibility and integrity at the expense of confidentiality.

For data, it is important that both users and providers are aware of national legislation, policies or other intellectual property or privacy laws that could govern the ability to store or process data in geographical locations outside national boundaries. In the medical fields, the high fidelity processing of images is also important in order to preserve the integrity of sensitive information.

2.3.1 Cloud Models and Security

Researchers in [39] carried out an analysis where they presented a variety of security issues across the different cloud computing models. These researchers also suggested counter measures though the counter measures seem to be more for implementation from the provider’s side they none the less acknowledged that outsourcing resources to third party in the cloud environment makes it harder to secure the resources.

In the SaaS model, which is the most popular deployment model, the deployment, control and enforcement of security mechanisms is mainly the responsibility of the provider. End-users access to the service is mainly via a web interface controlled by the provider, who may choose to restricted access based on IP address or geographical location.

In the PaaS model, users have more responsibility in managing their configuration and security mechanisms than in the SaaS model. Although, choices of security mechanisms are limited to configurations selected by the provider, the user may be able change or expand on the configurations even by deploying additional mechanisms.

In the IaaS mode, majority of the responsibility for the correct deployment, control and enforcement of security mechanism is on the end-users. In the multi-tenancy situation, the end-user’s choices and decisions are limited only to the virtualized infrastructure (images, network and storage). For example, it is possible that a different tenant would not pay as much attention to security or engage in activities that adversely affect overall security or the security of physical infrastructure is not adequate. As noted by researchers in [40], some attacks include the existence of cross-virtual machine attacks among VMs that co-locate with others. These attacks can be minimised by checking the logs of the VMs and also enabling a trigger alarm based on the IDS.

Many users with mission critical applications choose to deploy private clouds where they can also maintain adequate control over every aspect with clearer visibility and less concerns, while other users choose to only deploy non-critical application and resources on the community clouds.

Another area of concern for a cloud user is the case of denial of service (DoS) or distributed denial of service (DDoS) attacks [41]–[43]. Availability requirement vary also from user to user so also the time requirement needed to recover from failure is greatly determined by the security compliance checks to be performed. It is important that the service provider extends the clients’ security capability as at when needed.

2.3.2 Fuzzy logic in Cloud Computing Security and Trust

Boolean Logic, which modern computing is based on, cannot be replicated in the case of cloud computing security; rather the fuzzy logic approach is applied. Fuzzy logic as described by Zadeh in [44] is a class of object that has degrees of truth characterised by membership function raging between zero and one rather than the Boolean zero or one. It is difficult to describe trust with accurate probability based on only true or false, Zadeh advanced the idea of fuzzy logic understanding of normal language to represent some measurement of vagueness.

Furthermore, fuzzy logic which is based on fuzzy set allows operations of inclusion, union intersection and complement on its sets. Fuzzy logic allows all things to represent a degree of some form. It is descriptive instead of just black and white, it allows something to be represented with a bit of black and white a bit of grey, it allows one to represent an item as to what degree it is black and to what degree it white at the same time.

Now to fully represent any item using fuzzy logic, some basic processes must be employed, these are

- **Fuzzy Rule**: This allows the representation of items in linguistic form assigns linguistic values to them. This is usually a conditional statement which allows one to be able to represent the item with some degree of membership. The difference between classical rules and fuzzy rules are that classical rules is binary logic 1 or 0 which with fuzzy they are represented based on “membership degree”. For example if a man is short, it can also be represented in fuzzy as “to what degree is the man tall?” but for classical representation, the man can never be tall (0). Fuzzy rule have various parts that when put together provide a crisp solution to a problem.
- **Fuzzy Inference Type:** this maps a given input to an output. There are various inference techniques but two popular types are the Mamdani and Sugeno types[45]. Mamdani method [46] is commonly used method and it was chosen over the Sugeno method because it is intuitive, has widespread acceptance and well suited for human input.
- **Defuzzification**: This allows the final output to be a crisp number as it takes as input the various aggregate to produce a final single/crisp number.

No cloud platform is completely secured or completely trusted instead it should be “to what degree is this platform secure or trusted?” Systems that may be presently may not be secure later as it may need to update or upgrade some patches but that doesn’t now mean it is totally unsecure.

This therefore means that in terms of security and trust of a cloud platform, fuzzy logic would allow the representation of “to what degree is the cloud platform secure across all the layers and so therefore to what degree can I trust this platform?

2.3.3 Existing and Related Works in Cloud Computing Security and Trust

As the security and trust concerns continue to be more pronounced in cloud, various researchers continue to provide solution in this area but these works seem to be limited to the IaaS layer and nothing seem to have being develop for the whole infrastructure and no tool seem to be available to provide the user with the ability to check and probe continuously the security status of the cloud platform therefore the user is not able to completely trust the cloud platform.

Researchers in [47] only considered security on the IaaS layer with emphasis on TPM and the contract documents but how can a user probe and attest to the quality of service provided by the provider only by just what is written and signed in the contract?

Gonzales et al in [48] considered the IaaS infrastructure and provided alternative cloud security controls for defence called the cloud-trust model, the model provided by the researchers strengthens cloud service providers and minimises discovery of live virtual machines but this also doesn’t consider the end-user or the different layers that make up the cloud platform.

In [49], Saadat and Shahriari proposed a framework that would analyse and categorise customer’s security and trust concerns, it further considers a cycle that would continuously monitor and solve any problem on the cloud environment but this was just a framework and was not evaluated using any technology.

Sirohi and Agarwal in [50] considered a security framework that would provide protection to the data resources on the cloud and providing transparency between the cloud provider and cloud user. The real time monitoring of the resources is a very important and worthy but the use of only encryption technique to protect the data may attract unnecessary attention from the unwanted intruders to want to probe further why that data is encrypted.

In [51], the researchers considered a trusted model to verify and evaluate the security controls claimed by a cloud service provider to meet the customer’s requirement. In carrying out the evaluation and verification, the model considers trusted third parties and user’s past experience but these options do not provide an unbiased judgement so the judgment maybe inconsistent/flawed.

In the same vain, CSA STAR[8] created the self-assessment framework for cloud providers to make public their platform's security and control capabilities but these are sometimes outdated and no way for the end-user to verify before subscribing and even after subscribing there is no way to continuously probe the platform, so how can an end user trust the platform?

Wallom et al in [52] seem to make an effort in deploying trusted computing for mission critical applications in the cloud but the work had used a lot software patches and this automatically means the cloud platform software cannot be upgrade or updated and this is a great security flaw that must be tackled else the platform is never up to date and no matter the security controls and mechanisms it would always be prone to attacks.

Hwang and Li [53] also proposed a method of securing data on the cloud based on data colouring which generates unique colour drops from three different characteristics but this method has not been implemented or evaluated.

Researchers in [54] deployed a fuzzy based trust model but this model assumes all cloud platform are secure and after continuous probing user's trust level begins to drop when the necessary mechanisms have not been applied.

Even as Cloud computing provides virtualised, metered real time, on demand computing resources, the cloud user wants a secure trusted platform with its resources available anytime it is needed and an assurance that its data integrity is not tampered with.

The research reported in this thesis intends to address the following:

- Improve user's ability to measure and attest to a cloud platform security status
- Improve the trust relationship between end-users and providers
- Provide a technique for users to protect their data integrity and claim right to ownership.

2.5 Summary

Due to the need for mobility and improved access, users are gradually embracing cloud technology and its security cannot be over emphasized. Cloud computing security like any computing technology must also be examined based on security principles of confidentiality, integrity and availability and the individual associated contributions by the deployed security mechanism some of which are examined in this chapter.

The cloud infrastructure needs to be physically secure. Mechanisms that may be put in place to enforce physical security include monitoring of physical access, biometrics, CCTV among others. While accessing the cloud platform remotely, it is important to make sure that only authorised users have access to the platform and every authorised user has access to only the resources which they are authorised to access.

In the different cloud models, responsibility for security is shared between the provider and users. In the IaaS model, the user has a major responsibility while in the SaaS model, they have the least responsibility.

Due to the multi-tenancy nature of clouds, the security of the shared environment including network and other physical infrastructure are the sole responsibility of the providers as it is important to ensure that adequate leak proof separation (security zones or domains) is maintained between users.

As users take advantage of the operational and financial benefit of cloud, they are concerned that their information is kept safe and not disclosed to unauthorised persons, processes or devices. It is also important users are able to evaluate and attest to the platform and services they would be accessing even after signing the agreed contract which is usually referred to as Service Level Agreement (SLA), the SLA defines the level of service quality the end-user is entitled to from the provider [55]. Any security negligence leaves a gap and a wide door open to cloud threats and data breaches. Just signing an SLA doesn’t prove that appropriate security has been put in place.

Chapter 3

Deploying Trusted Cloud Computing for Data Intensive System Applications

Trusted cloud computing is a form of trusted computing applied in cloud computing to improve the confidentiality and integrity of the platform. As providers are making substantial effort to secure their platforms, there is a clear need among cloud end-users for a solution that guarantees confidentiality and integrity while at the same time providing users with the ability to verify the state of this solution or service.

Presently existing virtual machines running on various cloud platforms have limitations that prevent end-users from verifying their security states and also lack adequate protection from unauthorised access by privileged users such as the cloud providers. Trusted cloud computing provide cloud end-users with a possible tool to assess the cloud provider, the trustworthy state of the cloud platform, enable on-demand monitoring and application of industry standard based security solution in the field of cloud computing.

This chapter presents the deployment of trusted cloud computing for mission critical applications in the energy sector. The research presented in this chapter simplifies the integration of trusted platform module based integrity measurement into commodity cloud infrastructure by eliminating the need for “custom” software and patches; it also enhances instance-level security by including a distributed file and directory integrity checker for added security. The deployment of trusted cloud computing using the Eucalyptus Cloud software on server-grade hardware is presented, as well as the results of a comparative evaluation of the additional overhead in the instance creation/start-up based on a simulation of low, medium and high security settings. The trusted cloud computing infrastructure was made available to the power system application developers and users for deployment and testing.

3.1 Introduction

Mission critical services such as the electrical power grid require on-time availability and reliability of scalable computing resources, which is only possible through the deployment of cloud computing. However, more needs to be done with regard to the security and integrity of the cloud infrastructure and the virtual machines (instances) running on the “cloud” [56]–[59]. There is also a need for the secured processing of data whether in the “cloud” or in the traditional computing environment. The security of the data and applications can never be over-emphasised as the energy sector relies heavily on efficient and secured data.

This chapter presents a modified use-case secured platform for the use by UK power industry (which includes the Distribution Network Operators (DNOs), the Transmission System Operators (TSOs) and the UK National Grid (NG)). Cloud computing can facilitate and simplify the exchange of data between the Distribution Network Operators (DNOs) and the Transmission System Operators (TSOs) as required in the UK by National Grid (NG) [6]. The NG plays a vital role in the daily capacity planning and distribution of electricity in the U.K and makes extensive use of models based on real-data from DNOs and TSOs conforming to a strict standard and format as defined by the NG [60]. For example, data from a DNO would include the state of its electrical network; connectivity arrangements; electrical loads; sub-transient/transient currents, power injection values such as power, average voltage and power factor and reactance/resistance ratio at each grid connection point [61].

In a cloud-based environment, the integrity of the entire UK power system would depend on the ability of DNOs/TSOs to update and maintain only sections of data related to their equipment/network. The exchange of data between DNOs and TSOs is greatly facilitated by the adoption and use of a common file exchange format. However, the data is of great value (potentially highly sensitive to national security) as they describe the state of portions of the national transmission and distribution networks. The energy sector requires a trusted cloud computing infrastructure that can guarantee secure ownership and integrity of the uploaded data even when it is decrypted for processing. The research presented in this chapter focuses on intrusion detection (integrity / trust) and security of the virtual infrastructure (architecture) as a possible approach to addressing security when deploying cloud computing in mission critical sectors [62].

A group of DNO or TSOs (end-users) carried out the up-load, validation and storage of data in the secure storage locations (Elastic Block Store (EBS) storage volumes). Typically, the end-users operate exclusively over the secure interface of applications such as an SSL-enabled web-interface. Direct attestation and verification of the cloud Infrastructure's security and integrity by end-users (DNO/TSOs) is possible if enabled by both the CI provider and the application providers (NGs).

The Open Grid Systems (OGS), as a SaaS provider (and PaaS user), are responsible for creating the instances from the supplied images, installing and configuring additional or needed applications for use by end-users. As PaaS users, OGS have secure shell (SSH) access to their running instances complimented with SSH access to the CI, the latter is useful for conducting security / integrity testing and verification. Brunel University London as the IaaS provider is responsible for setting up the secured images and volumes from which the instances are created.

The use case adopts the OGS Cimphony application as the key interface between the end-users and others power-users such as the National Grid (NG). The Cimphony software application is network-model data visualization and analysis tool that can validate different network models present in the CIM format, merge these data models and transform them from one format to another.

The network models uploaded by the DNOs/TSOs are stored into individual EBS volumes which are tagged with the name of the DSO/TSO and a time-stamp of the upload. Each DNO/TSO has read/write access only to its volumes, while an NG would have read-only access to all volumes managed by the DNOs/TSOs.

In this use case, it is assumed that:

- Each user (DNS/TSO/NG) has a private public key pair.
- The NG and DNO/TSO exchange their public keys
- NG and DNO/TSO have read-only access to a pre-seeded IDS database for verifying the applications, data and configuration of the instances.

- NG and DNO/TSO have read-only access to pre-seeded manifest/database of measurements for the TC verification (openPTS) of the cloud-infrastructure.
- The Eucalyptus cloud infrastructure is integrated with the Trusted Computing technology described in Section 3.2. In this way, NGs (and DNO/TSO) can run/interrogate openPTS to obtain the TC measurements of the cloud-infrastructure.

Each SaaS provider workflow is as follows:

- Instantiate a VM from the stored image
- Verify that the cloud infrastructure can be trusted using openPTS
- Verify that the VM can be trusted using the Advanced Intrusion Detection Environment (AIDE) command.
- Mount the necessary Elastic Block Store volumes
- Install and configure the end-user application (Cimphony).
- Create the necessary access credential for the DNOs/TSOs
- Inform DNOs/TSOs about access credentials
- Wait for DNOs/TSOs to complete workflow
- Decrypt and verify the signature of the models/data uploaded by DNOs/TSOs
- Merge all the DNO models
- Convert and process the resulting models
- Elaborate or download the model over a secure channel
- Destroy the VM

The DNS/TSO (cloud end-users) workflow is as follows:

- Obtain the access credentials from NG
- Verify that the access credentials work correctly
- Verify that the application can be trusted
- Verify that the cloud-infrastructure can be trusted
- Verify that the instance can be trusted
- Upload the data/network model to the correct EBS storage device.
- Validate the network model using the Cimphony application
- Encrypt/sign the uploaded data using the NG public-key and DSO/TSO private-key.
- Inform NG on completion of task.

The integrity of the instance itself is established by the Advanced Intrusion Detection Environment (AIDE) which is an intrusion detection programme specifically referred to as file and directory integrity detection checker, while the integrity of the cloud-infrastructure is based on TC. The independent security verification of both instances and cloud-infrastructure ensures an overall stronger security when both instance and infrastructure have not been altered or modified.

Figure 3.1 shows a prototype implementation of a trusted cloud computing deployment on the Brunel University computer network where both TC integrity measurement and verification based on the TPM [22] and intrusion detection using AIDE [63] were integrated into a Eucalyptus based cloud platform. The prototype demonstrates the levels of containment used to provide infrastructure security, while also providing a form of integrity and trust for the clients using the platform.

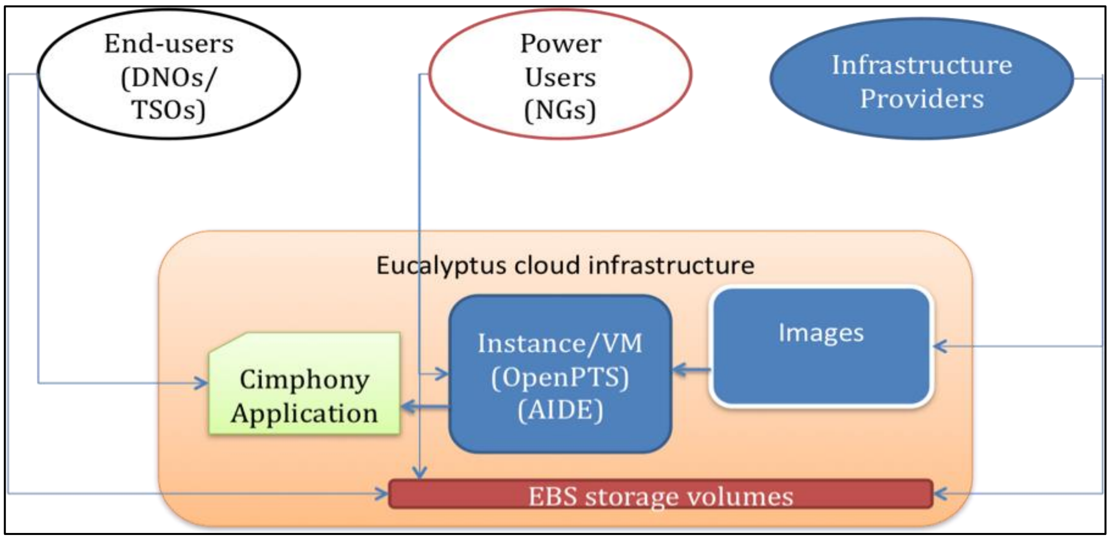


Figure 3.1: Deployed system on Brunel University Network

This research is in collaboration with industry partners Open Grid Systems [10] and NG [6]. The prototype was made available to the industry partners for application deployment and testing. The result suggests little or no overhead from the added security mechanisms.

3.2 Integration of Trust and Security

As reflected in [64] and [65], security in cloud computing is a well-researched area although most work do not adequately address security across the layered nature of cloud infrastructure (physical and the virtual machines running on the "cloud") and in many cases, also do not extend integrity verifications to the applications or services running on the cloud platform.

In the community cloud model (see Figure 3.1), individual partners may be provided with a diversified access model based on resource sharing needs as this creates a secured environment that guarantees trusted ownership and integrity of the cloud resources. Individual actors are responsible for the upload of owned data into dedicated storage locations that is subsequently shared (with diverse privileges) to a restricted set of partners, with the ability to verify the integrity and secure-storage of all accessible resources (owned or coming from partners) and the integrity of the cloud-infrastructure including hardware, boot-process, middleware and instances that process their own data.

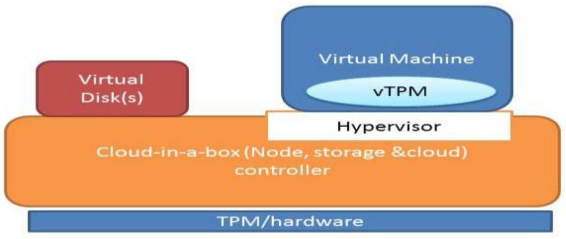


Figure 3.2: Logical overview of cloud environment secured by TC

Figure 3.2 shows the unique chain of trust that is used to secure the physical, virtual and IaaS layers. At the physical layer, a TPM [22] chain of trust extends from the hardware device through the firmware (BIOS), boot loader (kernel), operating system of the cloud middleware and virtualization layer (Hypervisor). This chain may also be extended to cover IaaS storage devices (virtual disks) as shown in Figure 3.2 However, for the use-case application, this particular chain is not automatically extended to cover instances running above the Hypervisor (Virtualization layer) because a virtual machine or instance, once created, is treated as completely different/independent computer. For these instances, the inclusion of a software based virtual-TPM (vTPM) device within them allows a different and unique chain of trust to be built for the kernel, operating system and software components of the instance.

Attestation of the Cloud infrastructure: The overall integrity of an instance or VM would also depend on that of the underlying cloud infrastructure, that is, it is important for a VM to verify that it was created by the right process (cloud-controller), it is running on the correct hardware (node-controller) and accessing/using the proper data storage devices (storage-controller). Such an attestation process thus implies some interaction between a VM and one or more component parts of its host cloud platform and infrastructure. It is noted that in cloud platform implementations such as Eucalyptus cloud-in-a-box[66], the cloud-middleware may be protected by additional security measures such as firewalls that limit/minimize exposure of the critical internal cloud structure to the hosted VMs, services and/or end-users.

In our implementation of a secure trusted community cloud (see Figure. 3.3), the TC attestation of the cloud-infrastructure (based on the TPM hardware chain-of-trust) from a hosted/running VM is carried out as a transaction (TC operation) targeted at its local controller over a secure-shell connection with reduced privileges. The use of a local/reduced-privilege secure-shell connection reduces the possibilities for network based man-in-the-middle attacks and also limits exposure of internal cloud structure.

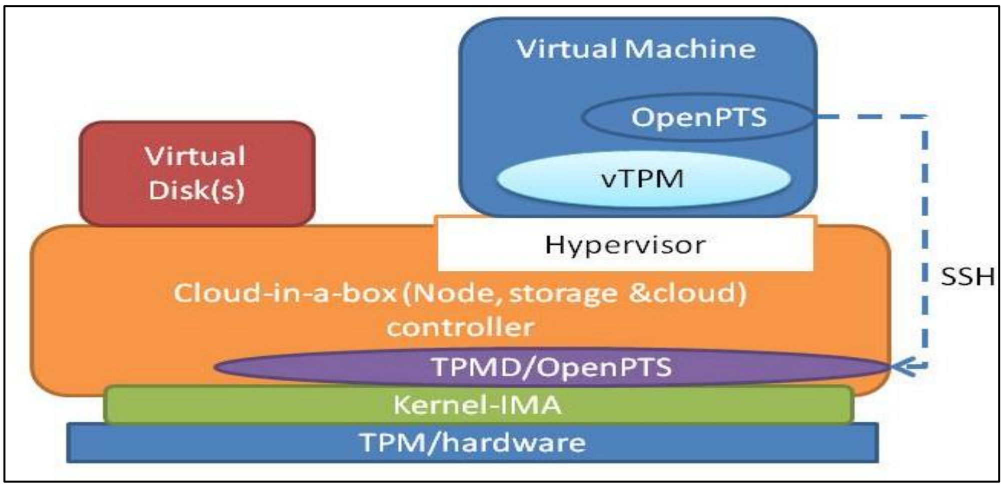


Figure 3.3: Architecture of the secure trusted cloud

The process of securing the physical and virtual resources that combine to make up the IaaS layer using TC was implemented thus, TPM support is enabled in the BIOS to measure the initial state of the physical system. In prior work by other researchers [52], trusted grub or grub-ima was used to secure the bootstrap layer. However, these are no longer supported on modern versions of operating systems such as Ubuntu 14.04 and have been superseded by the newer UEFI secure-boot process. Secure-boot is also enabled from BIOS and is already supported by modern operating systems including Microsoft Windows and Ubuntu 14.04. During secure boot, another option for enhanced security is to use TPM for measured boot. The Eucalyptus cloud platform is based on the CentOS Linux, where secure-boot is expected to be fully supported in a future release. The standard Linux kernel already includes the IBM Integrity Measurement Architecture for measuring all the applications, kernel modules and configuration files loaded at boot time. Enabling this requires passing the option “ima” to the booting kernel and mounting a corresponding securityfs virtual file-system during boot.

The above steps were used to build a chain-of-trust from the hardware/BIOS where the TC measurements of critical operating system and software components of the cloud-platform are stored and used for verification. The process of measuring, storing and verifying TC integrity of operating-systems and software components is covered extensively in other work including [52] and [67]. In the Linux operating system, a Trusted Core Service Daemon (TCSD) is used to export trusted services from the physical TPM into the operating system of the cloud-infrastructure.

A software application, the Open Platform Trust Services (OpenPTS) [67] application is used to implement the TC attestation procedure. OpenPTS runs in either collector or verification modes and can carry out remote attestations using the standard secure-shell application for transportation and protection against man-in-the-middle attacks.

In the collector mode, openPTS collects and maintains a private manifest/database of trusted-measurements to be examined during the verification process.

In our implementation, the private manifest/database of trusted-measurements is integrated into the immutable images (see Figure 3.1) from which VMs are created. When running, each VM/instance uses this private manifest/database for the on-demand TC attestation of the cloud-infrastructure.

Attestation of Instance/Virtual Machine: The inclusion of a software based virtual TPM (vTPM) device within the instances allows a new and unique chain of trust to be built for its kernel, operating system and software components of the instance. However, for our purpose, an instance/VM is always a TC verifier and the virtual TPM (vTPM) installed within the VM is only required to satisfy a dependency requirement of the openPTS application (installed in the VM) used for the attestation of the cloud infrastructure.

In our implementation, end-users of services provided by the secured/trusted community cloud can immediately verify service integrity based on a mandatory use of authenticated Secure Sockets Layers (SSL) certificates for all connections. However, this does not guarantee internal integrity of an instance (VM) providing the service, that is, the service is being provided by the expected software application using the expected configuration files and data. This additional assurance is provided during the attestation process by a file/directory level intrusion detection system (IDS), for which the hash of its pre-seeded database may be protected using a suitable TPM based chain-of-trust such as that of the host cloud platform/infrastructure.

3.2.1 Securing Cloud Platforms with TC and Intrusion Detection

Currently, security in most Cloud computing platforms is limited to the ability to partition/group instances by owners, or hierarchical levels of administration. They do not include or enable support for Trusted Computing integrity measurements/verifications based on TPM, an industry standard for computing hardware integrity measurement and testing, that depends on special hardware storage of cryptographic keys. Cloud platforms also do not include adequate intrusion detection mechanisms for certifying that the instances/virtual machines have not been tampered with.

TC provides an added level of trust related to the cloud-infrastructure as it allows measuring the collective integrity of the hardware platform, firmware and operating systems components responsible for booting.

Integrating TC into cloud implementations would allow cloud administrators to measure and verify infrastructure integrity; however, the end-user integrity measurement/testing of the underlying infrastructure require some additional mechanism such as a secure channel for end-users to access the cloud platform.

TC based on TPM is useful for measuring system integrity; it typically cannot detect exactly what was changed or locate the changes to exact file, the addition of a file and directory intrusion detection system allows fine-grain examination/characterisation of the changes that occurred.

Intrusion detection software such as AIDE provides file level integrity verifications (including permissions and attributes) based on stored hashes. A relatively small Linux system could easily have hundreds of thousands of files. A full scan and verification tend to be both processing-intensive and time-consuming as the hash must be recalculated for each file and compared to the stored value. The ability to limit the scan to certain selected files is quite useful and can significantly reduce the amount of time needed during verification.

The methodology used in securing the underlying cloud-infrastructure and instances for end-user verification involves the following steps:

- Enabling TPM hardware from BIOS/firmware: The cloud hardware is expected to include suitable TPM hardware.
- Enabling TC measurement related options on the Kernel layer. For Linux, this is by modifying the boot-loader options (adding IMA) and configuring a virtual file system (securityfs).
- Installing and configuring the operating system TPM related applications (trousers and openpts).
- Creating an SSH-only account for remote attestation by instances.
- Integrating TPM tools into the virtualised image.
- Pre-seeding the OpenPTS verification database in the image with the keys from the Cloud-infrastructure.

3.3 Prototype Design

3.3.1 Eucalyptus Cloud Software

Eucalyptus which is an acronym for Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems is a Linux based cloud software. This is the software that was used for the experimental platform. It enables the cloud service know as IaaS to be deployed on existing IT infrastructure either as a private, public, community or hybrid cloud. It is portable, modular and simple software which was initially developed to support High Performance Computing (HPC). The API offered by Eucalyptus is compatible with Amazon's EC2, S3, IAM, ELB, Auto Scaling and CloudWatch services thereby offering the capability of a hybrid cloud [66].

Most exiting cloud computing software are either proprietary or depend on software that cannot be extended for experimental purposes and studies.

Eucalyptus was chosen for this research based on the following features: it is open source which allows for further customisation and configurations, it provides a user/management web interface with custom credentials, allows for re-write of the scheduling controller and provides strong internal security settings[2], [63], [67], [68].

Eucalyptus allows for easy installation and as nonintrusive as possible unlike most cloud software that locks up or exposes the interfaces and platform to proprietary and closed software and resources. The framework implementation was carried out according to industry standards and thus allows and encourages third party extensions.

The Eucalyptus Cloud Software also allows the entire cloud infrastructure to be configured and deployed on a single computer (box). This “Cloud-in-a-box” technology is basically a CIaaS that provides basic functions like provision storage, network and security resources. Furthermore, it also includes auto scaling, cloud-watch and elastic load balancing, which are typically found in a fully functional “cloud”.

Eucalyptus provides some basic cloud computing security on its platform such as ssl certificate, user password management among others that were activated on the platform. It provides a network overlay that isolates the network traffic of different users.

The Eucalyptus framework comprises of various components but there are four high level components among them and each with its own web-service interface. These components are the cloud controller, the node controller, the cluster controller and storage controller [68], [69]. These are represented in Figure 3.4.

Cloud Controller: This is the entry point for all users (administrators, developers and end-users) of any eucalyptus platform installation. Cloud controller (CLC) queries the node manager for information about resources, The CLC manages all the underlying virtualised resources of any Eucalyptus cloud and makes visible to the users.

Node Controller: On each node or host is a node controller (NC) which in response to queries and control request from the cluster controller, the node controller queries and controls among others the system software, the host operating system and hypervisor. It is the node controller that discovers the node’s (or host’s) physical resources, size of memory, available disk space and even the state of other VMs or instances running on the node/host. Even though the node controller seems to be maintaining records of the instances or VMs on the host, there may be a situation where instances are started and stopped with mechanisms that are beyond the node controller. In respect of this and for best resources management and resource availability, it’s important that the node controller sends its information to the cluster controller.

Cluster Controller: The cluster controller (CC) acts the link between the node controller and the cloud controller. The CC has three main functions which are

- Schedules incoming instance run requests to the specific NC : when there is a request to run an instance, the CC contacts each NC of a host and sends the instance run request to the first available NC that meets and has the required resources to host the instance.
- Controls the instance virtual network overlay which is further discussed in a later section
- Gathers and reports information about a set of NCs : the CC can receive a list of resource characterises describing a resource requirement needed by an instance like memory, disk and can use these information to calculate how many instance of a specific type can be executed on the its NCs, this information is sent to the CLC.

Storage Controller: The storage controller is a data storage service that supports standard web service technologies and its interface is compatible with Amazon’s S3. SC acts as storage service for VM images and users can also use SC to stream data from instances that have been started on the nodes. It supports concurrent and serial data transfer and to aid scalability on the eucalyptus platform it does not provide locking for object writes.

Other components of the Eucalyptus software like the User Facing Services (UFS) serves as the endpoint for any Amazon web compatible services offered by Eucalyptus (e.g S3). The management console allows the user to manage the eucalyptus cloud.

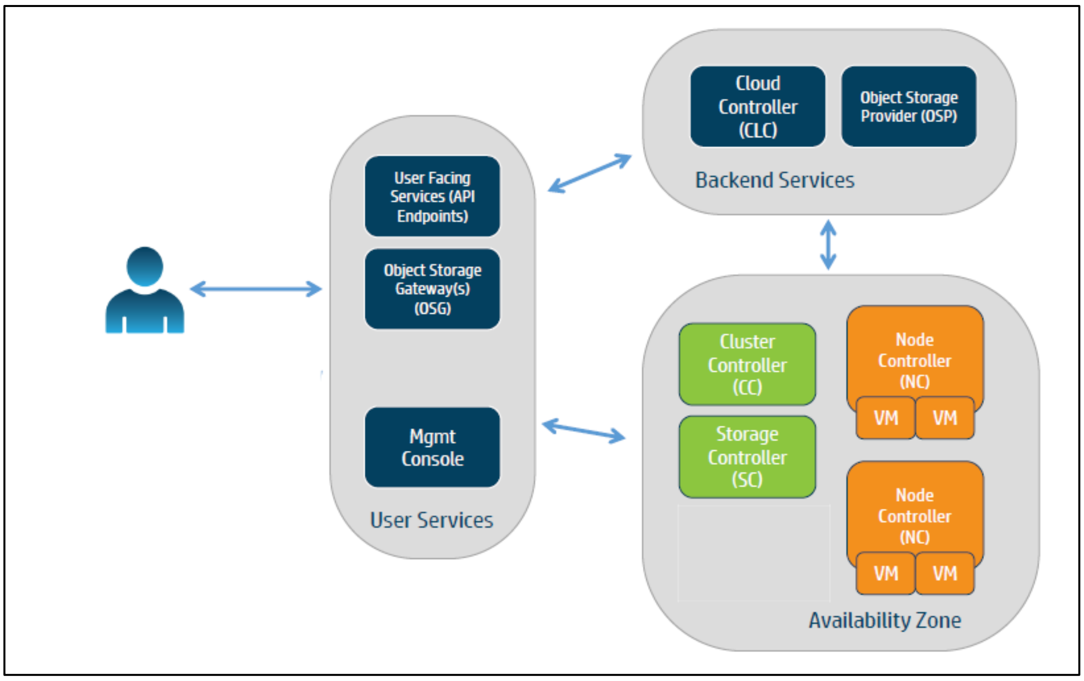


Figure 3.4: Major components of a Eucalyptus software(source [70])

Eucalyptus has been designed in such a way that it allows easy deployment into existing enterprise network topology. The instance network solution has also been designed to provide connectivity between various instances and interfaces, isolation from other instances or groups and also provides high performance.

In other to create and enforce security across the platform, Eucalyptus has what is referred to as security groups. A security group allows specific network rules to be applied on an instance or group of instances running on the cloud platform. It allows you control the network access to the instances. This specifies the kinds of traffic that are allowed in or out of the instances. Basically the security groups are set of firewalls that are applied to any VM associated with the group. The eucalyptus cloud software has a default security group and any instance created takes up that security though an administrator can change the rules for the default security, the administrator can also delete, create, authorise and revoke security groups[71].

In other to assign the resources of a node appropriately, the virtual machine capacity is limited to the size and number of node controllers available. Eucalyptus relies on the xen hypervisor [72] to create and run the virtual machines. Eucalyptus allows you to bundle describe, run, terminate and reboot a variety of Linux and Windows VMs [71].

TC measurements/verification based on the OpenPTS (Open Platform Trust Services) implementation and Intrusion detection based on AIDE (Advanced Intrusion Detection Environment) software were installed / configured on the images to further enhance security and trust of the images that would subsequently be used to deploy the instances.

3.3.2 Infrastructure Setup and Configuration

The cloud platform was deployed on a high performance Intel server with 4 Intel processors running at 2.27 GHz with 64 GB of memory/RAM. Each processor had 10 cores.

The specific hardware and software details are displayed in Table 3.1.

Table 3.1: Configuration of the deployed Cloud Platform

Hardware	HDD	1 TB
	Processor	40 Cores
	CPU	2.27 GHz
	LAN Connectivity	Gigabit Ethernet
Software	Cloud Software	Eucalyptus
	CI Operating System	CentOS
	Images Operating System	Ubuntu 14.01 (Trusty)

Installation of Eucalyptus was performed using a CD image of Cloud-in-a-Box version 3.4 downloaded from Eucalyptus website. In cloud in a box, all components (shown in Figure 3.4) are installed to a single physical host. It represents an optimal choice for research as the hardware requirements are limited even in production mode and it is also possible to install additional software packages on the cloud server using the RedHat yum package manager.

The Eucalyptus installer installed a customized version of RedHat 6.2 to the entire harddisk. During the installation process, the server network card (eth0) was assigned a public IP address of 193.62.138.225, netmask of 255.255.255.0. The IP gateway was set to 193.62.138.254 and DNS was set to 134.83.127.81. The following IP address range was assigned as “the public ip for use by the virtual machines“:   193.62.138.226 – 193.62.138.249.

In a Eucalyptus cloud-in-a-box installation, all instances (virtual machines) are attached to an internal network bridge named br0 that exists only inside the server machine, however all instance are usually isolated from each other on this internal bridge. The network connectivity over br0 is used to push configuration information to running instances (virtual machines).

The server uses 1 public IP address for itself. The end-users including those outside Brunel University and connect to the server using a web browser (on port 8443) in order to start their applications on demand. This is the cloud self-service model. All 15 other public addresses are placed in a pool to be used by instances (virtual machines).

Once an instance (virtual machine) is started and running, it gets an external IP address assigned to it from the pool. The physical server will use iptables/proxy-arp to also place the IP address of the instance on eth0: X (physical NIC eth0) alongside its own public IP. Once the instance (virtual machine) is destroyed, the iptables/proxy-arp/eth0: X entries are removed and the IP is returned to the pool.

With the self-service cloud model, the web interface of the server informs the end-user what exact IP address is assigned to their instance (virtual machine) from the pool once it is created. The end user now connects directly with his instance through the external IP using one of the approved protocols (ssh or http/httpd) and can use his power application.

Overall the installation was fairly straightforward and during the process, the system clock was set to UTC and a secure password was set on the root account. The installer reported that over 1000 packages were installed before requesting a reboot.

The first boot process was lengthy and additional configurations were automatically performed before showing the welcome wizard GUI, which aided with creating the normal user account and performing NTP settings. The GUI then reported the automatic creation of two Eucalyptus accounts. The first was a demo account, with a user named admin and password of password, while the second was the eucalyptus account with a user named admin and password of admin before requesting a new reboot.

Activating the TPM hardware started from the server BIOS, once the Eucalyptus hypervisor was started The bootloader configuration file (/boot/grub/enu.lst) modified to pass the option ima to the kernel on boot and subsequently an entry for the virtual file system (securityfs) was added to the file /etc/fstab. Finally, the   operating system TPM related applications (trousers and openpts) were installed and configured as specified in the OpenPTS quick start manual.

The installation comes with a basic (default) CentOS image that can be used to create an instance (virtual machine) for testing. It is also possible to install new images from the public Eucaytus store (eustore). However, such images contain limited functionality, old versions of software and are difficult to update.

An alternative method is obtaining generic pre-built cloud images of an O.S. from on-line site such as the Ubuntu Cloud service (http://cloud-images.ubuntu.com). Given the difference in boot kernels, it is necessary to use a generic boot-loader (known as kexec) for starting the specific O.S kernel contained within the image. The kexec utility for Eucalyptus is available from https://github.com/eucalyptus/eucalyptus/wiki/Kexec-Images/. A step-by-step procedure for adding and running pre-built Ubuntu images in Eucalyptus is documented in Appendix A1.

As earlier stated, a more secure approach is to create a cloud image from an installation disk. This process requires considerable more effort but provides the benefit of ensuring that only relevant applications are included within the image. The process of creating and testing a cloud image from an Ubuntu 13.10 CD-ROM/ISO disk-image is documented in Appendix A2. Similar to the use of pre-built images, these images may be started using the kexec boot-loader or alternative the installed kernel and ram-disk combination may be extracted from the image and used directly as an Eucalyptus boot-kernel. This procedure was used to build a highly trusted cloud image which also included the installation of additional software such as vTPM and OpenPTS which are required for TC attestation.

For other less mission critical applications, it is also possible to create a reasonably secure cloud image by modifying an existing one. Here, the process requires a review/audit of all installed components followed by removal of unwanted applications/files and the addition of required applications. For example, creating a custom image that includes a specific version of the hadoop map-reduce java application started with expanding an existing pre-built cloud image. First, a raw disk-image of adequate size was created and the contents of an existing cloud image transferred into it. The added space is required for installing the hadoop application. Next, the newly created image was mounted at a temporary location and its contents modified by installing a java development kit from the Ubuntu repository. Then a version of hadoop was downloaded as a TaR file and installed within the image. Finally, the installed hadoop was then configured to start-up at boot-time automatically by adding suitable entries into the system start-up file located at /etc/rc.local. Subsequently the kernel and ramdisk were extracted and bundled along with the image for use in Eucalyptus. Appendix A4 contains the procedure used to modify a pre-built image with the installation of hadoop and subsequent testing on the Eucalyptus server.

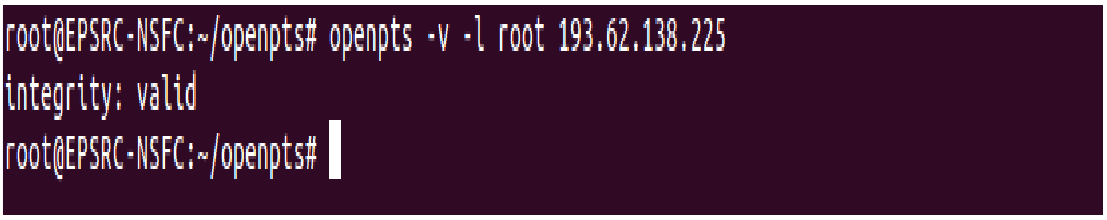
By default, all instances are created with a temporary file-system that is destroyed as soon as the instance is deleted. This implies that all changes made within an instance are lost as soon as it is closed. In Eucalyptus, the alternative EBS-backed images allow the creation of instances with addition of an external storage location that can be used for storing persistent or changed data. The procedure to create an EBS backed image is documented in Appendix A3.

Depending on the VM’s configuration, the platform could run up to 80 VMs. The images running on the deployed system are based on 64-bit Ubuntu Linux operating system version 14.01/trusty, which have been further customised in the current research.

3.3.3 TC Verification Process

The OpenPTS technology operates with one of the systems, that is, the cloud-in-a-box host system taking the role of the collector and the other system(s) (the instances) taking the role of the verifier

The integrity measurements/verification provided by OpenPTS allows an end-user to verify that the cloud-in-a-box server, where the image is hosted (called the collector) is actually the right and secured one (Figure 3.5). The verification process is performed over a secure shell (SSH) connection based on pre-seeded keys at the verifier side.



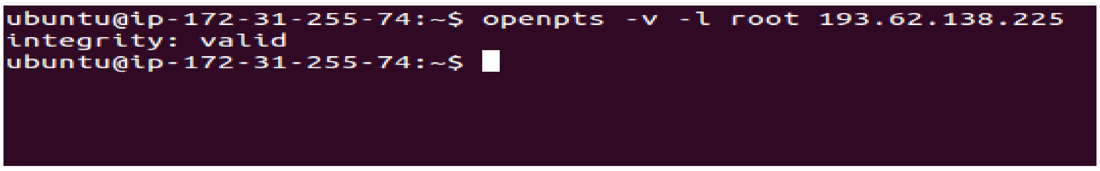
```
root@EPSRC-NSFC:~/openpts# openpts -v -l root 193.62.138.225
integrity: valid
root@EPSRC-NSFC:~/openpts#
```

Figure 3.5: OpenPTS verification on the collector

The cloud-in-a-box server performs a trusted boot based on data contained within its TPM hardware. TPM is a microprocessor chip built into a machine to enhance its security by integrating cryptographic keys into the machine; these keys provide authentication (proves the platform is what it claims to be) and attestation (platform is trustworthy and has not been breached). These keys are checked for consistency by the verifier.

With a suitable OpenPTS command syntax (–v option), the PaaS layer user can compare the current values against a pre-seeded set (Figure 3.5). The result, which means the keys tally (success) or does not (failure), is reported immediately. On failure, it is also possible to automate more drastic action such as the immediate shut down of the instance/virtual machine.

A verification failure (reported by OpenPTS) is an indication of a variety of issues, including the possibility that the verifier is communicating with a different or rogue machine, booting from the wrong machine or even that the image has been tampered with or modified. TC verification can be used to verify the security and trust of a cloud-computing infrastructure.



```
ubuntu@ip-172-31-255-74:~$ openpts -v -l root 193.62.138.225
integrity: valid
ubuntu@ip-172-31-255-74:~$ 
```

Figure 3.6: OpenPTS verification on the verifier

Within an instance, file level Intrusion detection is provided by AIDE, an integrity checker for files and directories, based on a variety of digest algorithms (md5, sha1, rmd160, tiger, whirlpool, sha512, etc). AIDE has the ability to check for inconsistencies in the file/directory attributes. Based on the required speed/performance, the scan/verification may be limited to specific files and directories.

3.3.4 Hierarchical Security Model

Apart from the built-in security groups features in Eucalyptus, a hierarchical security model was also used to provide secure Cloud computing services to a group of researchers both within and outside the University. These researchers are categorized according to the following access levels:

a. Power Systems Cloud Infrastructure administrators – This category of users is responsible for providing secured images to be used by other users for their application /software. They have control of the underlying cloud infrastructure and can access the system on both ports 8443 and 8888 using the address https://powersystemscloud.brunel.ac.uk:8888/ or https://powersystemscloud.brunel.ac.uk:8443/

b. Power Systems Cloud Application Developer: The second category of users on this platform is responsible for creating instances (VMs) from the provided images and subsequently installs desired applications; these applications are in turn used by the end-users. They have no access to or control over the underlying cloud infrastructure and can only access the system using the address https://powersystemscloud.brunel.ac.uk:8888/

c. Power Systems Cloud Application Users: The third category of users on this platform is the end-users. Users in this category have only HTTP/HTTPS access to instances / applications running on the platform.

3.4 Evaluation and Results

The integration of either TC or intrusion detection with Cloud computing introduces performance overhead especially during boot up or in the process of setting up attestation or verification [10]. The goal of the research experiment was to determine the overhead that could result from the introduction of either TC integrity measurement/verification or AIDE based intrusion detection/verification into an instance.

Cloud images which are preinstalled disk images with various operating systems which include CentOS and Ubuntu were configured on the platform. The different images had different security configurations and were thus classified as “low security” if the images had only the operating system basic security configuration, “medium security” if trusted computing using TPM was configured and “high security” if both TPM and AIDE were configured. Instances were then deployed from these images.

The improved security appears to have limited and negligible overhead for single instances and it can therefore be concluded that it is not going to be noticeable until thousands of instances are started. Table 3.2 shows the average boot-time recorded for each security level and Figure 3.7 shows a graphical representation of the 21 instances deployed from each of the 3 categories of images during the research experiment. These instances were deployed from the images classified into three broad security levels: low, medium and high as explained above. Where the low security group do not include TPM or AIDE and serve as an experimental control or baseline group. The medium group include only TPM based integrity measurements/verification. The instances at start-up typically started the TPM-emulator device and setup the user-environment for remote verification of the CI platform. While the high security group include both integrity verification and AIDE intrusion detection, the instances start up the TPM emulator device and setup the user-environment for remote verification of the CI platform. The AIDE database is already pre-seeded, but the end-user verification process needs to be set-up as well.

From Table 3.2, the control group had an average boot-time of 54.16 seconds, while the medium security group had an average boot-time of 55.38 seconds; the high security group had an average booth time of 56.81 seconds. The results show that the overhead incurred including TPM based integrity measurements/verification into a cloud platform for this research is 1.22 seconds while the overhead incurred including TPM and AIDE based intrusion detection into a cloud platform is 2.65 seconds. This is further represented in Figure 3.7, which shows the results obtained from the deployment of 21 instances at each security level. The maximum overhead between low and high security levels is between 3 and 4 seconds. The results suggest that the improved security overhead on the cloud platform is limited and capable of scaling to large numbers (thousands) of instances.

Table 3.2: Instances Boot-time

	Low Security	Med Security	High Security
	Min:Sec.CenSec	Min:Sec.CenSec	Min:Sec.CenSec
1	00:53.32	00:55.58	00:56.81
2	00:54.11	00:55.32	00:57.01
3	00:53.78	00:55.35	00:56.66
4	00:54.21	00:55.14	00:57.18
5	00:53.81	00:55.28	00:56.59
6	00:53.63	00:55.54	00:57.09
7	00:53.54	00:55.13	00:56.54
8	00:53.78	00:55.20	00:57.01
9	00:53.92	00:55.22	00:56.61
10	00:53.63	00:55.01	00:56.67
11	00:54.88	00:54.91	00:56.90
12	00:54.84	00:55.51	00:57.21
13	00:54.39	00:55.49	00:56.69
14	00:54.51	00:55.44	00:56.90
15	00:54.10	00:55.78	00:56.72
16	00:54.14	00:55.00	00:56.66
17	00:54.50	00:55.26	00:56.87

18	00:54.41	00:55.70	00:56.83
19	00:54.77	00:56.22	00:57.02
20	00:54.80	00:55.78	00:56.61
21	00:54.23	00:55.03	00:56.51
Av Sp.	00:54.16	00:55.38	00:56.81

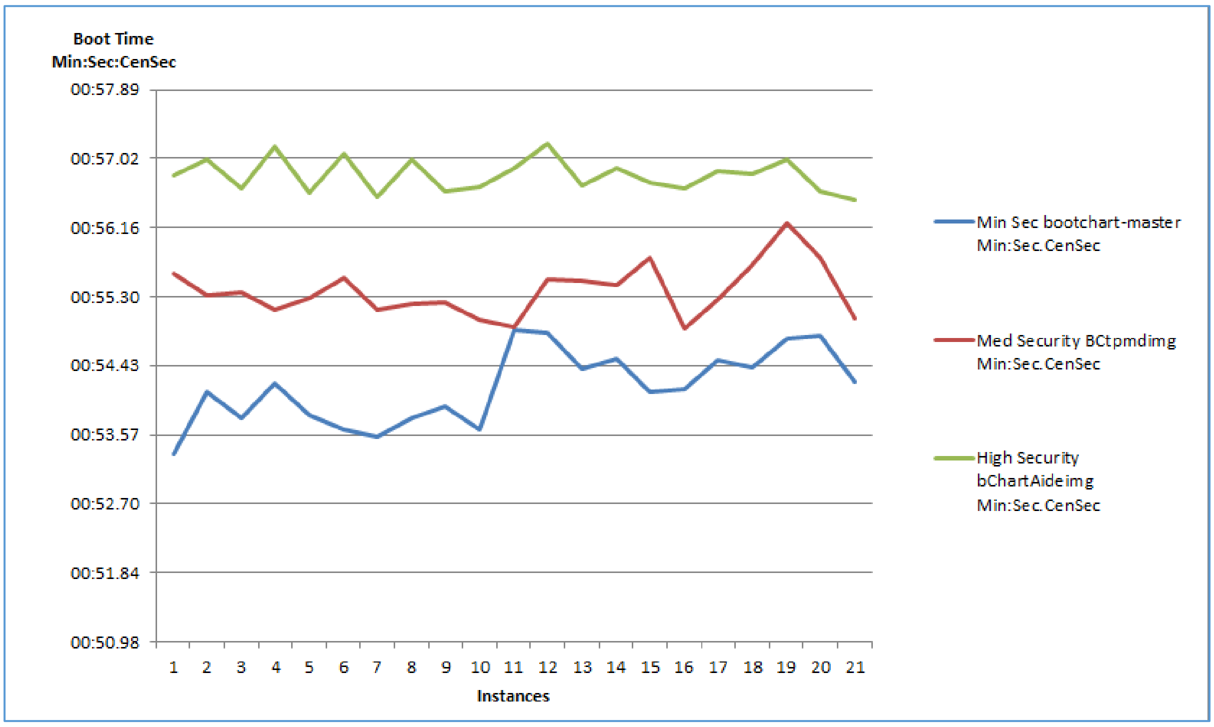


Figure 3.7: Boot time representation

3.5 Summary

A trusted cloud deployment especially suited for mission critical applications in the energy sector was presented and discussed in this chapter. The approach involves the integration of end-user accessible TPM integrity measurement/verification into the cloud platform/infrastructure without the need for “custom” software or patches. Furthermore, security is enhanced by the inclusion of an instance-level file and directory integrity checker for selected files and directories. Using a trusted cloud computing infrastructure can guarantee trusted ownership and integrity of the uploaded data as it would ensure that each operator is provided with a diversified access model based on data sharing needs. In a cloud deployment approach, individual organisations/users share a common cloud platform and sometimes not necessarily retaining control over their sensitive data or applications deployed on external infrastructure. The research presented here provides additional levels of trust for cloud infrastructure which allows individual organisations/users to retain control of their sensitive data/processes. A working prototype of the secure trusted cloud deployed on the Brunel University London network has been made available to industry partners OGS and NG to deploy and test power system applications like Cimphony.

Chapter 4

Securing Resources in the Cloud with Data Colouring

Given the nature of the cloud platform and its present limited ways to detect unauthorized access or modifications to data; cloud users and data owners do not trust that their data is adequately secure on the cloud platforms. This chapter presents a technique of data colouring for securing data on cloud platforms based on establishing and using concatenated fingerprints for watermarking. In the prototype implementation presented here, cloud users and data-owners secure their data by first colouring it offline before uploading onto any cloud platform. The colours may be used to detect unauthorized modifications and also suggest the path of data loss or theft. A basic shell-script implementation of the technique based on steganography is presented along with some evaluation results from its use and evaluation on an experimental cloud platform deployed during the research work.

4.1 Introduction

Due to the anonymous nature of cloud, data owners hardly trust the cloud providers and the cloud platforms to deploy their sensitive data on the cloud; this in turn has adversely affected the rapid deployment of cloud computing infrastructures[26], [28]. Data confidentiality, integrity and availability remain the major security concerns for users with different security considerations. By virtue of its multi-tenancy configuration, sensitive data can be comprised or tampered with by an unauthorized user and the cloud provider may not be able to provide a record of which client accessed which data or in the case the data is tampered with, the user or cloud provider cannot trace on which cloud platform it occurred [73].

Trust and security would greatly be enhanced in cloud computing when cloud users and data owners are able to secure their data before uploading onto the cloud platform and are still able to trace and confirm any distortion to the data and the exact path of distortion. This means cloud security challenges must be addressed from the provider’s end, on the server and resources side and also from the client or user’s side [12]. Measures have been put in place to protect data from been tampered with, limit unauthorised access or illegal usage using a suitable mechanism such as hashing, data-colouring or encryption or obfuscation. Encryption is one common measure put in place to render data unreadable to unauthorised users through scrambling of data, although this process could attract undue attention to the data [74].

Another measure frequently used is watermarking as various digital formats especially those for images including the portable document format (PDF) already support the easy embedding (and removal) of visible watermarks. These watermarks are usually located in well-defined sections thereby making identification and unauthorized removal easy.

Table 4.1 shows a comparison of possible self-defence techniques for securing data used by end-users. It is not exhaustive but offers a quick overview. As may be inferred from the Table, data-colouring could be an optimal technique for cloud platforms/applications as coloured data may still be processed without overhead (the colouring is transparent) while also providing the ability to detect tampering and/or identifying and reporting data-loss . That is, coloured-data (output of a data-colouring process) may be viewed as being able to maintain integrity and availability of data without a processing overhead. Therefore, considering security related metrics of Confidentiality, Integrity and Availability in a cloud context, hashing as a data-protection technique can only provide integrity, while obfuscation provides only limited confidentiality; data-colouring provides integrity and availability; while encryption provides confidentiality and integrity but not availability.

This chapter presents a way of securing data using a technique of data colouring [53]. The technique allows data owners to first secure their data offline (off the cloud platform) by colouring it before subsequently uploading or processing the colour data on any cloud platform. Based on steganography, the implemented shell-script allows the user to detect if the data has been tampered with and identify the path through which it was tampered; this in turn ensures data integrity and confidentiality. The technique has been evaluated on the deployed experimental cloud platform at Brunel.

Table 4.1: Comparison of various data security techniques

	Hashing	**Data-colouring**	**Obfuscation**	**Encryption**
Example	MD5, SHA	Watermarks, fingerprinting	Minimization, compression	Code-table/cipher
Association/ Technique	mathematical	Mathematical/ embedding	Entropy reduction/ transformation	Cryptographic transformation
Creation overhead	Low	Low	Medium	High
Resulting data can be directly processed on clouds	Yes	Yes	Yes (if process is reversed and recreated after processing)	Yes (if decrypted and encrypted after processing)
Security /features	Data integrity	Data ownership	Making data/content illegible	Making data/content inaccessible to unauthorised access.
Notes	Storage of hash is external to data.	Embedding/ distribution of watermark/ fingerprint inside data. may be difficult to detect and remove if steganography is used	Relatively easy to undo and redo	Difficult to undo without original code-table

4.2 Watermarking and Data Colouring

Watermarking is a security feature that prevents and discourages counterfeiting through the addition of an identification image/pattern with varying visibility. In digital watermarking, a digital mark (pattern) is embedded in a digital file; the digital watermark, which may sometimes be hidden, serves to identify ownership (and copyrights) thereby verifying the authenticity and integrity of the digital file.

An extension of the watermarking concept known as fingerprinting ensures that different watermarks are embedded in every copy of the distributed data-sets (digital files), this aids the detection and tracking of both perpetrators and the path of data distortion [75], [76]. Digital fingerprinting (watermarking) may also include information that is useful for identifying unauthorised modifications to the content.

Most data owners are still not comfortable with the idea of having a faceless entity host their data where there is still no existing solution that allows the user to secure its data before uploading on to the cloud platform or while processing. Most existing networked environment security measures can be extended to the cloud environment [53].Since the cloud is a multilayer entity, enforcing security has to be done across the different layers. Security measures in cloud computing has to be a build-up of all the layers. These security measures once enforced across the layers would make users confident of the platform and assured that their data's integrity has not been tampered with while still having unlimited availability to computing resources.

A cloud platform is only secured if and when both the cloud user and cloud providers are able to participate fully depending on one another to perform certain task [12], [65]. While IaaS involves the sharing of common hardware, the cloud provider needs to enforce security across the network and cloud platform through firewalls, intrusion detection, DNSSec and encryption among others. Copyright protection needs to be put in place for the applications that would run on the platform while measures like data colouring, watermarking, and monitoring needs to be enforced on SaaS layer. Though watermarking provides a form of security and ownership, it doesn't stop unauthorized users from locating and removing the image.

Data Colouring (DC) may be considered as a special form of digital watermarking, where fragments of the digital mark known as colour drops are distributed or spread out within the data. That is, the fragments of the digital mark are not co-located or limited to a specific location or segment. Data colouring allows users to secure their data using colour drops without the drops being visible[53].

Figure 4.1 shows the data colouring process, where according to [53] and [65] the colour drops are a combination of an *"expected"* value - Ex known only to the data owner, the "*entropy*" value – En known only to the users in a particular group and the *hyperentropy* value He known to all the users of the cloud infrastructure. Ex, En and

He are combined together to generate a collection of colour drops that forms a unique colour that neither the cloud providers nor other cloud users can detect.

It is argued in [77] that the computational complexity in obtaining Ex, En and He is lower than that in conventional encryption and decryption process. It can be observed from Figure 4.1 that $En + He$ represents information that is agreed and exchanged between a cloud provider and a data-owner such as the public-key component of a cryptographic key-pair. Specifically Ex is the information that is only known to the data owner such as the private component of personal cryptographic key-pair and an encryption password. En is the value known to only the users of a specific group on the platform such as the public-key component of the cryptographic key-pair for that group and He is the value know to all users of the platform, such as the public key component of the cryptographic key-pair of the cloud platform.

The forward colour generator is composed of two distinct operations, these are the colour drops generator and the data colouring process. The colour drops generator is responsible for producing a sequence of bits from the combination of $(En + He) + Ex.$ In traditional watermarking, the colour drops would be the unique watermark.

The second operation inserts the generated bit sequence (watermark) into the user-data to obtain the coloured data. The coloured (watermarked) data may be subsequently stored or processed on the cloud platform or a copy maybe delivered to a recipient.

In data colouring, the coloured (or watermarked) data retains all the functionality of the original data but contains additional identification bits that is included within the data in a manner that does not permit easy detection or removal of the colour drops (unique watermark).

The backward colour generator verifies the inserted colour drops (watermark). It consist of 3 separate operations; they are - the extraction of colour drops from the coloured data, the generation of the colour drops based on the same input parameters initially passed to the forward colour generation and an operation to compare the generated colour drops to the colour drops extracted from the coloured data.

The colouring and verification of the colour drops is carried out by the data-owner as they would require knowledge of Ex.

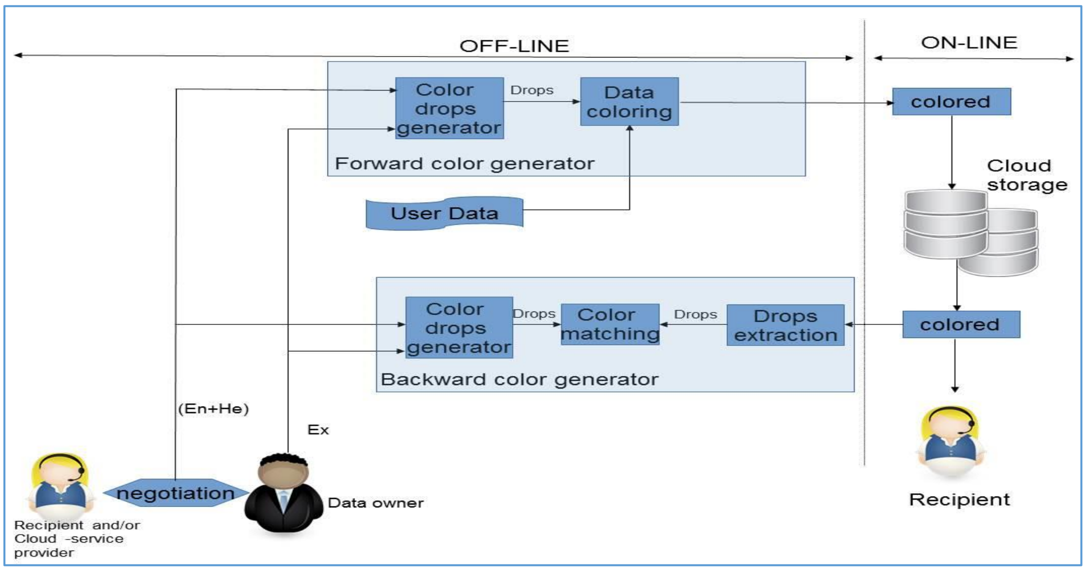


Figure 4.1: Data Colouring Process

In data colouring, the colour drops (or watermarks) are embedded within the data (or data-set) to provide integrity and identification without impacting the functionality of the data. The presence of colour drops should be invisible (or transparent) during regular use of the data-set. The process of colouring or embedding the colour drops within the data sets should also be resilient against unauthorized reversal while reliably supporting the authorized location and extraction of colour drops.

Steganography, the art of hidden writing [78] is used as the primary technique for embedding colour drops into the data sets in the data colouring implementation discussed in Section 4.3. The aim in steganography is to embed and hide the existence of a message within another carrier message from a third party.

Steganography requires the presence of empty (unused) locations where data may be inserted within a data-file. In Information Theory, the entropy (in bits) of a unit-length multiplied by the total-length of that message is a measure of how much information the message contains. It is important to note that unit-length is domain dependent, that is, the unit-length for a spoken message is different from the unit-length of the same message in written form even when both types convey exactly the same information. The implications are that the entropy is related to both type and length of a message. In digital water-marking, where the preservation of the information-content of a message is important, the modification of a message (by noise) is domain specific; that is an audio message requires modification in the audio-domain (by audio noise) while a visual image is affected in the visual domain (by visual noise). That is, visual-noise would not affect information content of an audio-messages or vice-versa. While it is possible to simply combine two digital files into a single one using a concatenation technique. For example, a “PNG” may be combined with a “ZIP” file into a new file where the upper part is the original “PNG” image and the lower (bottom) part is the original “ZIP” file. The resulting (combined) file may be treated either as a “PNG” file or a ZIP file and its size is a sum of both original files, however, it is quite trivial to extract the individual files. Steganography requires format specific methods and techniques for inserting hidden messages into a digital file.

Steganography is different from Cryptography in the sense that it does not make the message unreadable from third party but just embeds and hides a message (secret communication) within it. An advantage of steganography is that it doesn’t attract undue attention [78], [79] as the original message continues to function as normal (the hidden message is invisible or transparent).

Sometimes, the hidden message may be pre-encrypted, compressed or encoded before embedding in the carrier message. Also, sometimes, the hidden message may be split among a set of files but then all files must be available, unmodified and processed in the right order in other to retrieve the hidden data/message. In steganography, the security of the hidden message is cryptographically enhanced when the secret messages are first encrypted before embedding into the carrier. The hidden message is usually embedded as bit-level in the redundant space of the carrier message most times, in a statistical manner to avoid possible detection or modifications.

4.3 Implementation

The implementation presented here expects that colouring is carried out completely offline to enhance security; therefore, this means only coloured data should be uploaded to cloud platform(s). The colouring of data-files before uploading to various cloud-service models is expected to improve the integrity of the cloud-based resources as it enables data-owners to detect, trace, report and document unauthorized access and use of uploaded data/data-files to respective cloud providers or users.

The shell-scripts discussed in this section depend on the free and open-source steganography tool OutGuess[80] for colouring data files (embedding hidden data into redundant bits of a carrier file) or extracting colour drops from already coloured files (i.e. extracting the hidden data from redundant bits). OutGuess relies on specific data handlers that would identify and modify redundant bits to carry the secret message. OutGuess is able to handle different data formats as long as a suitable handler is available. Table I presents the sources of colour drops used in the data-colouring implementation.

The cryptographic hash of a Public Key Infrastructure (PKI) private key of the data owner guarantees the colour drops contain information that ascertains ownership, while the hash of the PKI public key of data recipient or cloud service is useful to trace and highlight path of data loss or theft and the hash of the data content itself is useful for detecting unauthorised modifications.

Table 4.2: Data Sources for Colour Drops Generation

Item	**Contribution**
Data-file to be coloured	Fingerprint to detect unauthorised modifications to content
Private-key of data-owner	Fingerprint to identify data owner
Public key of recipient or cloud-service	Fingerprint to trace path of data-loss/theft

Furthermore, a password is used during the embedding process to encrypt the colour drops thereby securing them against unauthorized modification and removal.

The use of the original data-file as well as suitable PKI keys such as *pretty good privacy* (PGP) keys for creating the digital-fingerprint (watermark) guarantees uniqueness (and *entropy*) while also satisfying other defining conditions of Ex, En and He such as, knowledge limited to data-owner and association to defined group of users (or cloud-platform).

4.3.1 Forward Colour Generator

Figure 4.2 shows the source code of the forward colour generator (fcg.sh) command-line shell-script that generates the colour drops and uses them for colouring the original data.

```
#!/bin/sh
echo "Enter file-name with data to be coloured [Enter]: "
read name
echo "Enter password key [Enter]: "
read key1
echo "Enter private key file of data-owner [Enter]: "
read key2
echo "Enter public key file of recipient/could-service [Enter]: "
read key3
type=$(file -b --mime-type "$name")
drops=$(openssl md5 "$name" | cut -f 1 -d' ')$(openssl md5 "$key2" | cut -f 1 -d' ')$(openssl md5 "$key3" | cut -f 1 -d' ')
dropsfile="/tmp/$$"
echo "$drops" > "$dropsfile"
coloured=$(dirname "$name")/coloured-$(basename "$name")
if [ "$type" = "image/jpeg" -o "$type" = "image/pnm" -o "$type" = "application/pnm" ]
then
      outguess -k "$key1" -d "$dropsfile" "$name" "$coloured"
else
      echo "Not supported file format"
fi
if [ -s "$coloured" ]
then
      echo "Data was successfully coloured and saved to file \"$coloured\""
else
      echo "Colouring was NOT successful: Something went wrong."
fi
```

Figure 4.2: Forward Colour Generator Scripts (fcg.sh)

The script (fcg.sh) concatenates the md5 hashes of the three input-files to obtain a unique digital fingerprint (384 bits) that forms the colour drops for colouring. For portability, the md5 sums are generated using the “openssl” software application (a command-line executable). The colour-drops are embedded in the original data using the OutGuess steganography tool where the drops are treated as a "hidden" message to be embedded in the original data.

The fcg.sh scripts can be briefly described thus. In this case, *brunel_letter.jpg* is the data file to be coloured, *id_dsa* is the private key of the data-owner and *9FBB231E.asc* is the public key of the cloud provider. At the prompt of the scripts, the user provides the file name, a password, the user’s private key and the provider’s public key – these are the input values. The “file” command then determines the Multi-Purpose Internet Mail Extensions (MIME) type of the original data file. The implementation depends on the "file" command tool to determine MIME type of the data to be coloured. The colour drops are then created by concatenating the respective md5-hashes of the three input files, subsequently a temporary file is created containing the drops and its file-name is created by prefixing the name of the original data-file with the word “coloured”. Now the “outguess” utility is called to embed the colour drops into the original data-file and saves to the pre-set output filename for supported file types otherwise prints an error message (Line 19). On successful completion of colouring a message is relayed to the user (Line 23) and an alternative message (Line 25) in the case of failures.

In the resulting script (Line 10), the discovered MIME-type is used to ensure only supported types are passed to the “outguess” tool, however, this idea may also be used to also select alternative steganography tools that are capable of colouring data types not supported by the "outguess" software application.

4.3.2 Backward Colour Generator

Figure 4.3 shows the source code of the backward colour generator script (bcg.sh). The script extracts colour drops from a coloured file and compares with colour drops generated directly from the input parameters. The script (bcg.sh) uses the OutGuess steganography tool for extracting the colour drops (the hidden message) from the coloured file with the supplied pass-phrase for decryption.

```
 1 #!/bin/sh
 2 echo "Enter file containing COLOURED data [Enter] :"
 3 read coloured
 4 echo "Enter file containing ORGINAL (UNCOLOURED) data [Enter] :"
 5 read name
 6 echo "Enter password key [Enter]:"
 7 read key1
 8 echo "Enter file with PKI private-key of data-owner [Enter]: "
 9 read key2
10 echo "Enter file with PKI public-key of recipient user/cloud-service [Enter]: "
11 read key3
12 type=$(file -b --mime-type "$coloured")
13 drops=$(openssl md5 "$name" | cut -f2 -d' ')$(openssl md5 "$key2" | cut -f2 -d' ')$(openssl md5 "$key3" | cut -f2 -d' ')
14 dropsfile="drops-$$"
15 echo "$drops" > "$dropsfile"
16 name2="${name}.txt"
17 if [ "$type" = "image/jpeg" -o "$type" = "image/pnm" -o "$type" = "application/pnm" ]
18 then
19       outguess -k "$key1" -r "$coloured" "$name2"
20 else
21       echo " Not supported file format"
22 fi
23 if [ -s "$name2" -a -s "$dropsfile" ]
24 then
25       echo "Extracted drops=\"$name2\""
26       echo "Generated drops=\"$dropsfile\""
27       diff  "$name2" "$dropsfile"
28       if [ $? -eq 0 ] # the diff command returns 0 if there is a match and 1 if otherwise
29       then
30             echo "Extracted and generated colour drops  match - DIGITAL FINGERPRINT VERIFIED"
31       else
32             echo "Extracted drops do NOT match generated drops - DIGITAL FINGERPRINT NOT VERIFIED"
33       fi
34 else
35       echo "Something went wrong: Unable to locate either the extracted or generated colour drops "
36 fi
```

Figure 4.3: Backward Colour Generator Scripts (bcg.sh)

The script then generates a new set of colour drops based on the input parameters it is then compared with the extracted set and a match or mismatch reported. Comparison is carried out using the "diff" command-line tool.

In the case of the bcg.sh scripts, the input parameters would be the data that had been coloured initially in this case *coloured-brunel_letter.jpg*, the original data file which is *brunel_letter.jpg*, same password used in fcg.sh, the private key (of the data owner) which is *id_dsa*. The “file” command determines the Multi-Purpose Internet Mail Extensions (MIME) type of the original data file, the colour drops are created and stored in a temporary file and the suffix ‘.txt’ is added to the filename.

The “outguess” utility is then called to extract the colour drops from the coloured data file and saves to a pre-set output filename. If the coloured file’s MIME type is not supported by “outguess”, an error message (line 21) is printed. The files containing the extracted and generated colour drops are then compared and a message on successful verification (line 30) or an error message if the drops don't match (line 32). The bcg.sh also depends on an additional tool the "diff" command to determine if the extracted colour drops match the generated colour drops.

4.3.3 Theft and Loss Responsibilities

An important feature of this data colouring implementation is its ability to highlight a path of data-loss/theft based on fingerprinting.

Table 4.2 presents a simple matrix showing how the theft/loss responsibilities (path) may be determined from the corresponding inputs used during the data colouring process. In Table 4.2, row 1 represents the classical watermarking process as only the identity of the owner is verifiable from the colour drops (watermark).

Rows 2, 3 and 5 suggest that colour drops based on the corresponding combinations would not carry owner information and in such cases, the drops cannot be used to prove ownership of the data. Row 4 and 6 suggests combinations for which the drops may also be used to identify either a Cloud Service Infrastructure (CSI) or a Cloud Service Provider (CSP) or single-recipient. Row 7 highlights the combination for which drops are capable of also identifying individual CSP, CSI and recipient.

From Figure 4.1, the verification of colour drops is expected to be carried out by the data-owner.

Table 4.3: Theft / Loss Responsibilities

	Private Key of data-owner	**Public key of cloud-service**	**Public key of data recipient**	**INFORMATION OBTAINED FROM DROPS**
1	YES	NO	NO	Identity of data-owner
2	NO	YES	NO	Identity of CSI
3	NO	NO	YES	Identity of recipient (CSP)
4	YES	YES	NO	Identity of both owner and CSI
5	NO	YES	YES	Identity of both CSI and recipient (CSP)
6	YES	NO	YES	Identity of both owner and recipient (CSP)
7	YES	YES	YES	Identity of owner, CSI and recipient (CSP)

4.3.4 Mathematical Representation

The cryptographic hash used in generating the colour drops is based on the Message Digest checksum algorithm popularly called MD5, which is usually used to verify data integrity and authenticity [81], [82].

According to Hwang and Li in [53], the expression for colour drops can be represented by a combination of $Ex + En + He$ where Ex is known only to the data-owner, En and He are the values or public key component of the cryptographic key-pair that may be shared by users on the cloud platform. Specifically, En is known or shared only among the users of a common group on the platform and He is known or shared by all the users on a particular platform irrespective of the group they belong on the platform.

In this implementation, the generated colour drops (before embedding/steganography) are represented mathematically by the expression of $f(OF) + f(PrK) + f(PuK)$ where $f(\mathrm{x})$ is the md5checksum operation, OF is the original file, $f(OF)$ is the md5checksum of the original file, PrK is the private key of the file, $f(PrK)$ is the md5checksum of the private key, PuK is the public key of the recipient or CSP and $f(PuK)$ is the md5checksum of the public key.

In the data colouring implementation, Ex may be represented by the Equation (4.1).

$$Ex = f(OF) + f(PrK) \tag{4.1}$$

As these two items are known only to the data-owner, which implies that

$$En + He = f(PuK) \tag{4.2}$$

It follows that the embedded colour drops is then given by Equation (4.3).

$$Ex + En + He = RC4(f(OF) + f(PrK\ ) + f(PuK), K) \tag{4.3}$$

where $RC4(x, K)$ is the encryption function as it is implemented in “outguess” and K is the password supplied by the data-owner.

Mathematically, the simple watermark technique may be described as the embedding of a watermark (W) in a data set (D) such that:

- W can be reliably located and extracted from D
- W is large (the embedding has a high data rate).
- Embedding W into D does not adversely affect the functionality of D
- Embedding W into D does not change any statistical properties of D.
- W has a mathematical property that allows us to argue that its presence in D is the result of deliberate actions.

In data colouring, the watermark is defined as:

$$W = Ex + En + He \qquad (4.4)$$

that is,

$$W = RC4(f(OF) + f(PrK) + f(PuK), K) \qquad (4.5)$$

Generally, the colouring process represented a transformation of D by W that is:

$$D' = T(D, W) \qquad (4.6)$$

where D' is the resulting coloured data, D is the original data, W is the colour drops/watermark and $T()$ is the data-type specific transformation function used during colouring to embed the drops into the original data such that substituting the above we have

$$D' = T(OF, RC4(f(OF) + F(PrK) + f(PuK), K)) \qquad (4.7)$$

Equation (4.7) is the mathematical representation of the coloured data set. Where D' is the coloured data, OF is the original file or data, $T()$ represents the embedding of W into D (data-type specific transformation during steganography), $RC4(x, K)$ is a function that encrypts x based on the password $K, f()$ is the MD5 checksum function, PrK is the private key of data-owner and PuK is the public key of the recipient /CSP or CSI.

The above equation suggests that in this implementation the colour drops (represented by $Ex + En + He$) is protected by an encryption function and subsequent embedding in the original data in such a way that the functionality of the original data is preserved while making its unauthorised extraction or removal difficult.

4.4 Cloud Platform and Testing

Simulation is already a widely used research technique in Science and Engineering, for example, simulation is heavily used in areas such as climate-modeling, drug-design, material-science, supply/logistics and protein- analysis. However, simulation in Cloud Computing requires Cloud specific solutions due to the service oriented nature of clouds combined with other features such as elasticity of resources, multi-tenancy of resources, multiple layers and components. The virtualization of resources is fundamental to Cloud Computing and so it is vital that the simulation process can present virtualized resources for improved simulation of resource elasticity. In Cloud Computing, the selection, scheduling, allocation and consumption of resources is typically governed by algorithms that may be influenced by both internal and external factors such as user requirements, current consumption levels, availability requirements and other legal requirements (for example SLA documents) [83] which may also affect direct or in-direct Trust. Thus, the accuracy of simulations in Cloud Computing platforms is enhanced by the ability to simulating virtualized resources and complex scheduling/allocation of elastic resources. Researchers in [84] and [85] provided a comprehensive list of Cloud simulators that satisfy these requirements, however, none appear to provide objects that may be used to direct study Trust in Cloud Computing. Researchers in [86] carried out a research on Simulation of Security on computational GRIDs with a focus on improving scheduling by including security considerations. Researchers in [87] focused on simulation of Trust relationships and consider both direct and reputation based Trust in computational grids using both discrete and Fuzzy Logic algorithms for selection. Researchers in [88] examined Trust in distributed and peer-to-peer networks and considered discrete algorithms.

Shaikh and Sasikumar [84] provided a comparison of several different Cloud simulators including Eucalyptus and the CloudSim toolkit (this well-known Cloud Simulator written in the Java programming language) based on some selected features. [85] compared a wider range of Cloud Computing simulation tools including CloudSim toolkit, CloudAnalyst (which appears to be graphical and similar to CloudSim), GreenCloud (based on Network Simulator NS2), iCanCloud, MDCSim (from the University of Pennsyvania), NetworkCloudSim and VirtualCloud. Both

[84] and [85] rated CloudSim within the top four tools for Cloud Simulation. Other works including [89] and [90] present aspects of CloudSim including portions of its rich API for simulating various cloud objects including large scale IaaS, PaaS or SaaS cloud implementations. Figure 4.4 shows the basic architecture of CloudSim 2.0.

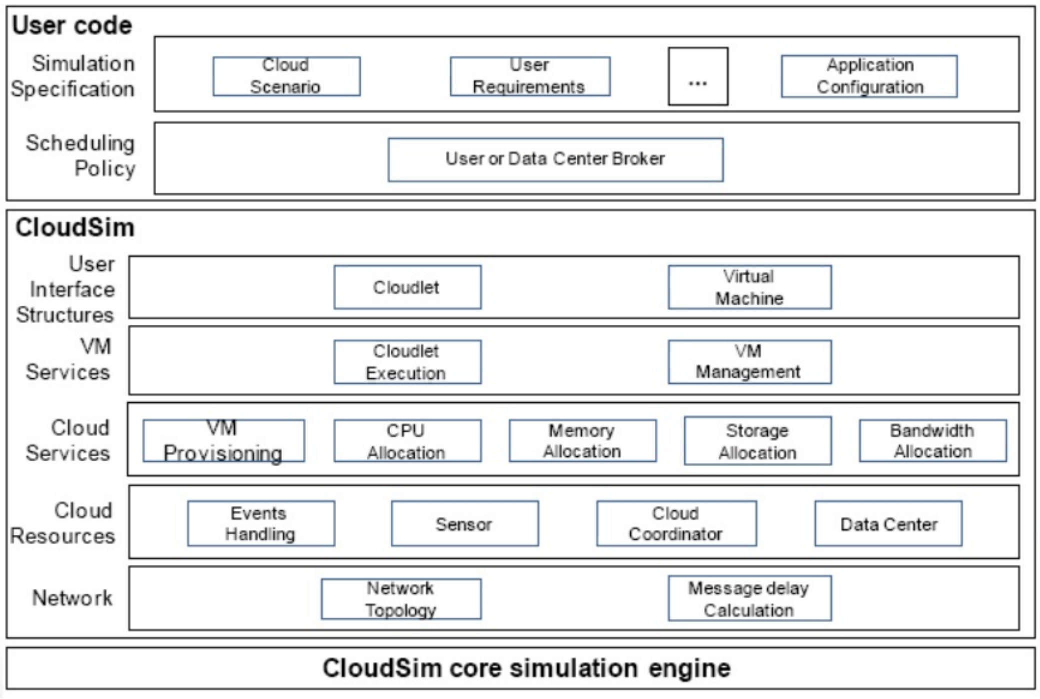


Figure 4.4: CloudSim Architecture (source [89])

[91] described a basic scenario of using CloudSim where a datacenter (object) has one or many Host (object), and each host has one or many VMs. Each VM deals with many cloudlets (or units of a cloud service). In CloudSim, each VM is assigned several cloudlets which are processed using a selected scheduling policy such as time-sharing and space-sharing. Simulation of Cloud IaaS layer involves extending the Datacentre object, which manages a number of host objects which in turn manage VM during their life cycle. A host represents a physical computing server and is defined with a pre-configured processing capability (MIPS - millions of instructions per second), memory (RAM) size, storage capacity and a provisioning policy for the allocation of processing cores to virtual machines. The hosts are assigned to one or more VMs based on a VM allocation policy that should be defined by the IaaS service provider. CloudSim hosts may be single-core or multi-core hosts. Simulation of Cloud PaaS layer involves the allocation/provisioning of virtual

machines on hosts that satisfy some characteristics which include storage, memory, software environments and availability or zone requirements. In CloudSim, cloudlets may be used to define custom software environments/ applications that can be deployed within a VM instance using a virtual machine allocation policy (VmAllocationPolicy). The default allocation policy available in CloudSim assigns VM to hosts on a First-Come-First-Serve (FCFS) basis. Implementing a Trust based policy would require extending the default policy or extending the VmAllocationPolicy class.

For Cloud SaaS layer, CloudSim uses cloudlets to model individual cloud-based application services (such as content delivery, social networking, and business workflow). However, each application is defined in terms of its computational requirements/complexity and requires specification of program instruction length, program size, data transfer overheads, output data size.

As shown in Figure 4.5, simulating complex large scale Clouds with CloudSim does not require knowledge of the underlying core simulation engine and the results from the simulation engine include the result of processes, the time consumption of each cloudlet. [92] and [93] show the extensibility of the CloudSim toolkit. In [93], the toolkit is complemented with a completely new set of objects were created to provide fine-grain simulation of network components and behaviour as shown in Figure 4.5.

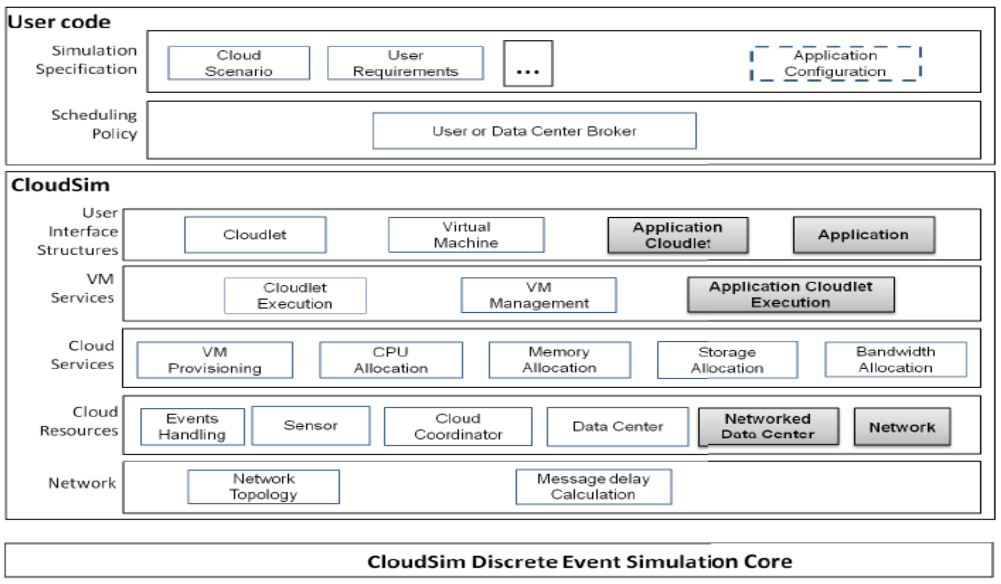


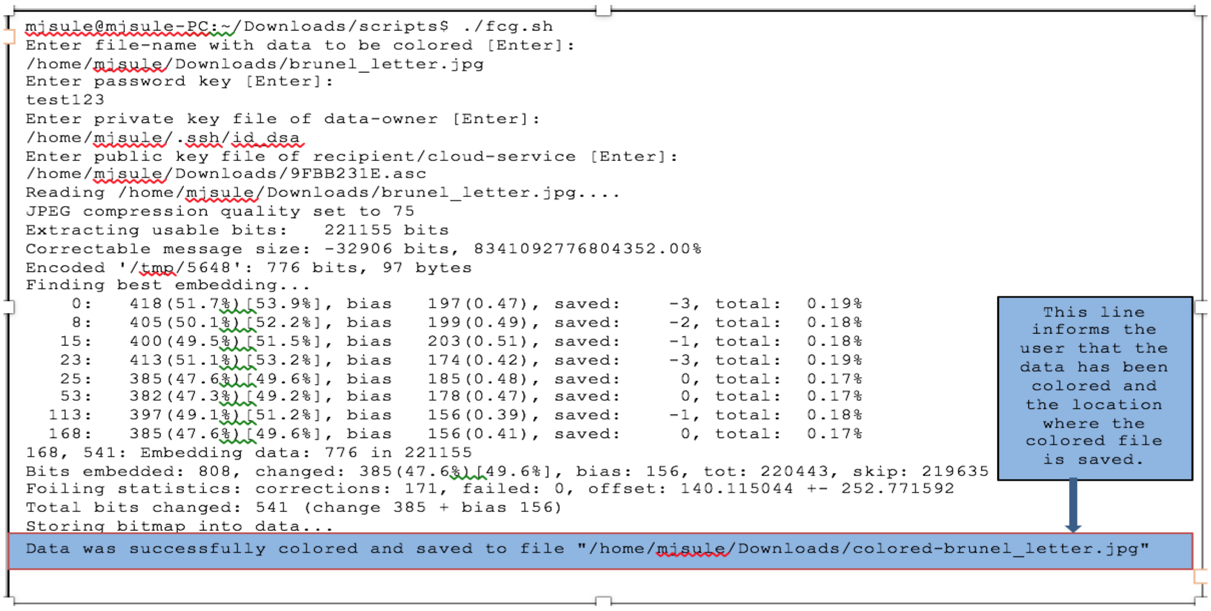
Figure 4.5: NetworkCloudSim architecture (source [93])

However, in this research work, an experimental approach using the Eucalyptus cloud platform was used rather than simulation using CloudSim. The experimental approach allows studying addition real-world situations that is possible with simulations.

A trusted cloud computing platform was deployed using Eucalyptus [94] enabled by the TPM [95]. The cloud platform integrates end-user accessible TPM integrity measurement/verification without the need for “custom” software or patches. Furthermore, on the platform, security is enhanced by the inclusion of an instance-level file and directory integrity checker for selected files and directories. In this cloud deployment approach, individual organisations or users share a common cloud platform and sometimes not necessarily retaining control over their sensitive data or applications deployed on (foreign) infrastructure.

The data-colouring implementation reported in this work is aimed at providing integrity/protection of uploaded data as it would ensure that each operator/user can retain and verify ownership of sensitive data with a flexible access model based on data sharing needs.

The implemented scripts may be obtained from the url https://powersystemscloud.brunel.ac.uk:8888/brunelece. Figure 4.6 shows the fcg.sh script running during the colouring of a jpg image. The first input item to the script is the jpg image to be coloured followed by the encryption password.

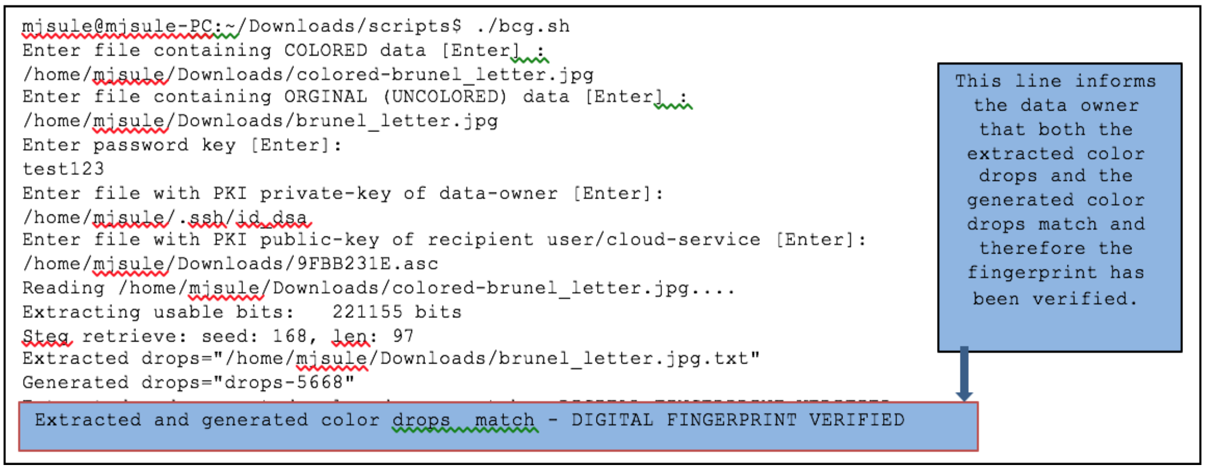


```
mjsule@mjsule-PC:~/Downloads/scripts$ ./fcg.sh
Enter file-name with data to be colored [Enter]:
/home/mjsule/Downloads/brunel_letter.jpg
Enter password key [Enter]:
test123
Enter private key file of data-owner [Enter]:
/home/mjsule/.ssh/id_dsa
Enter public key file of recipient/cloud-service [Enter]:
/home/mjsule/Downloads/9FBB231E.asc
Reading /home/mjsule/Downloads/brunel_letter.jpg....
JPEG compression quality set to 75
Extracting usable bits:   221155 bits
Correctable message size: -32906 bits, 8341092776804352.00%
Encoded '/tmp/5648': 776 bits, 97 bytes
Finding best embedding...
    0:   418(51.7%)[53.9%], bias   197(0.47), saved:    -3, total:  0.19%
    8:   405(50.1%)[52.2%], bias   199(0.49), saved:    -2, total:  0.18%
   15:   400(49.5%)[51.5%], bias   203(0.51), saved:    -1, total:  0.18%
   23:   413(51.1%)[53.2%], bias   174(0.42), saved:    -3, total:  0.19%
   25:   385(47.6%)[49.6%], bias   185(0.48), saved:     0, total:  0.17%
   53:   382(47.3%)[49.2%], bias   178(0.47), saved:     0, total:  0.17%
  113:   397(49.1%)[51.2%], bias   156(0.39), saved:    -1, total:  0.18%
  168:   385(47.6%)[49.6%], bias   156(0.41), saved:     0, total:  0.17%
168, 541: Embedding data: 776 in 221155
Bits embedded: 808, changed: 385(47.6%)[49.6%], bias: 156, tot: 220443, skip: 219635
Foiling statistics: corrections: 171, failed: 0, offset: 140.115044 +- 252.771592
Total bits changed: 541 (change 385 + bias 156)
Storing bitmap into data...
Data was successfully colored and saved to file "/home/mjsule/Downloads/colored-brunel_letter.jpg"
```

This line informs the user that the data has been colored and the location where the colored file is saved.

Figure 4.6: Generating colour drops (fcg.sh running)

As shown the data-owner is identified by a DSA private-key taken from the secure shell (SSH) application, while a PGP public-key belonging to the cloud-service provider (Brunel University London) is used to identify the cloud-platform/service. Figure 4.7 shows the output of the bcg.sh script during the successful verification of the coloured file. As shown both the original file as well as the coloured version is required for successful verification. The steganography tool (outguess) can extract the colour drops from the coloured file once the right password is provided. The original file is needed for the generation of a new set of colour-drops.

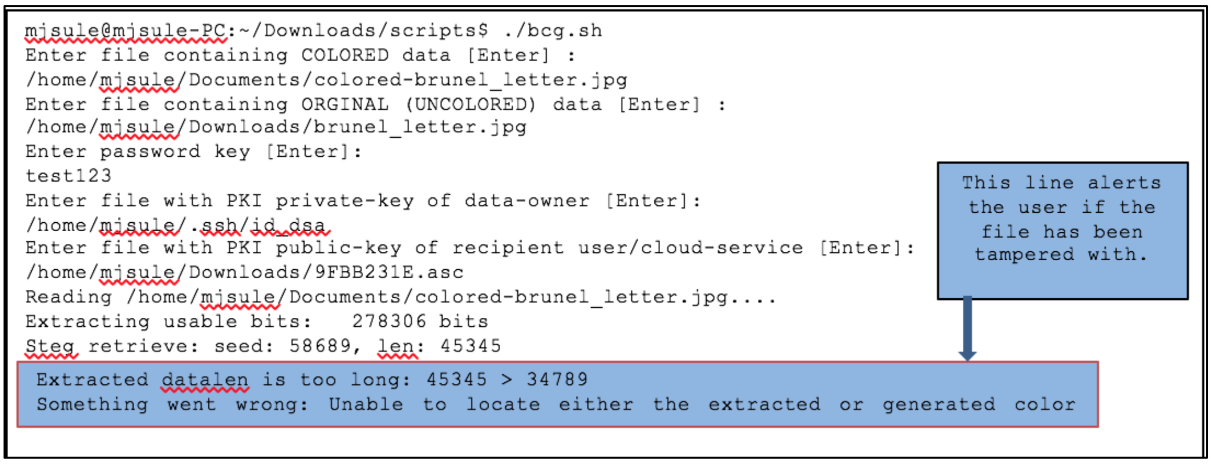


```
mjsule@mjsule-PC:~/Downloads/scripts$ ./bcg.sh
Enter file containing COLORED data [Enter] :
/home/mjsule/Downloads/colored-brunel_letter.jpg
Enter file containing ORGINAL (UNCOLORED) data [Enter] :
/home/mjsule/Downloads/brunel_letter.jpg
Enter password key [Enter]:
test123
Enter file with PKI private-key of data-owner [Enter]:
/home/mjsule/.ssh/id_dsa
Enter file with PKI public-key of recipient user/cloud-service [Enter]:
/home/mjsule/Downloads/9FBB231E.asc
Reading /home/mjsule/Downloads/colored-brunel_letter.jpg....
Extracting usable bits:   221155 bits
Steg retrieve: seed: 168, len: 97
Extracted drops="/home/mjsule/Downloads/brunel_letter.jpg.txt"
Generated drops="drops-5668"
 Extracted and generated color drops  match - DIGITAL FINGERPRINT VERIFIED
```

This line informs the data owner that both the extracted color drops and the generated color drops match and therefore the fingerprint has been verified.

Figure 4.7: Extracting colour drops with bcg.sh

Figure 4.8 shows the output of the bcg.sh script when detecting modifications to the coloured data file where a word was changed in the file. As shown, the steganography tool would fail to reliably extract the colour drops from the tampered file and produces an error message accordingly.



```
mjsule@mjsule-PC:~/Downloads/scripts$ ./bcg.sh
Enter file containing COLORED data [Enter] :
/home/mjsule/Documents/colored-brunel_letter.jpg
Enter file containing ORGINAL (UNCOLORED) data [Enter] :
/home/mjsule/Downloads/brunel_letter.jpg
Enter password key [Enter]:
test123
Enter file with PKI private-key of data-owner [Enter]:
/home/mjsule/.ssh/id_dsa
Enter file with PKI public-key of recipient user/cloud-service [Enter]:
/home/mjsule/Downloads/9FBB231E.asc
Reading /home/mjsule/Documents/colored-brunel_letter.jpg....
Extracting usable bits:   278306 bits
Steg retrieve: seed: 58689, len: 45345
Extracted datalen is too long: 45345 > 34789
Something went wrong: Unable to locate either the extracted or generated color
```

This line alerts the user if the file has been tampered with.

Figure 4.8: Verifying colour drops with bcg.sh

4.5 Summary

In this chapter we have presented a technique of data colouring for securing user data resources in cloud platforms.

The implementation creates colour drops from concatenated fingerprints that allow the verification of data owner, cloud service provider or recipient while also protecting against unauthorised modifications.

The concept of data colouring presented in this chapter can be applicable to all data formats; though as it is for now based on the OutGuess steganography tool, the present implementation discussed in this chapter can only be used on the following digital image formats: JPEG, PPM and PNM. Future work would investigate the support of additional data formats and other steganography tools.

Chapter 5

Modelling Trust in Cloud Computing based on Fuzzy Logic

As mission critical applications continue to be deployed on the cloud platform, a trusted distributed end-user attestable multilayer security tool to verify the trustworthy state of the platform is necessary. This is important as the end-user wants to be confident that the service is available and reliable, with accountability and non-repudiation. Tradition security and privacy controls continue to be implemented on cloud but due to its fluid and dynamic nature, research work in the area of end-user attestable trust evaluation of the cloud platform seem to be limited. With the nature of cloud, even though users are given and usually sign service level agreements with the providers[96] these are still not enough, the user would want a transparent system with a facility that allows the user to trace or determine the relationship between varying trust relationships across the cloud layers, components, algorithms and applications especially at large scale. A trust model using fuzzy logic was deployed; this model is useful in determining the trust values for a cloud platform or service. Using this model an end-user is also able to classify and compare various cloud platforms. The results obtained show that the model deployed in this research improves end-user's confidence when selecting or consuming cloud resources.

5.1 Introduction

Several security protocols and tools have been enhanced and adopted to fit cloud computing but few of these take into account specific issues as it relates to the cloud end-users [97], [98]. Cloud end-users are unable to fully adopt the cloud platform as it lacks transparency, accountability and governance unlike other computing technologies [97]. A cloud end-user wants to be able to have control of the services they are accessing, whether across the different cloud layers or physical locations.

While a lot of research has gone into making cloud computing more secure by enforcing security mechanisms like encryption, firewalls, and security groups, obstacles to trust still hinder potential users from completely adopting "cloud" (as it is sometimes called). End-users are concerned about the "faceless" provider taking control and having access to their resources during processing or storage in the cloud. The end-user wants to be more aware about what happens at the backend of the cloud platform.

An end-user's trust concern does not lie entirely on technology but also lack of transparency. While an end-user desires a secure platform, the end-user also wants measures that would promote transparency and accountability and this is lacking even among major providers like Amazon EC2 or S3, Microsoft Azure and Google. Insufficient information could lead an end-user to distrust even the most secure system [97], [99].

In general, it may be stated that the overall trustworthiness of any cloud resource may be derived from an algorithmic sum of trust levels (security measures) that are enforced or available across all layers or parts of a computing system [100].

In the case of cloud computing, the ability to probe and test on-demand the trustworthiness of all assigned or consumed computing resources would make end-users confident of the platform as it provides them with additional assurance of data-integrity both during storage and processing [101]. An objective aggregation or evaluation of various security mechanisms configured across the layers of a cloud platform would provide the end-user with useful information on the trustworthy state of the platform.

As stated by Andert et al in [102] the key principle of any security design and implementation is that security must be built into every layer of the solution. The researchers also defined trust modelling as the process performed by a security architect to define complementary threat profile on a use-case-driven flow analysis. The model identifies the mechanisms that necessary to respond to a specific threat. A trust model includes an explicit validation of an entity's identity.

Figure 5.1 shows the three popular layers of the cloud platform, these are also known as the delivery models – *IaaS, PaaS and SaaS*. From Figure 5.1, we also see that an end-user who subscribes to a SaaS provider or any of the services may not be concerned about the trustworthiness of the PaaS layer or other layers but maybe concerned with the trustworthiness of the SaaS layer as that is the resource the end-user wants; or the end user maybe subscribing to all the services and maybe concerned with the overall trustworthiness of the platform.

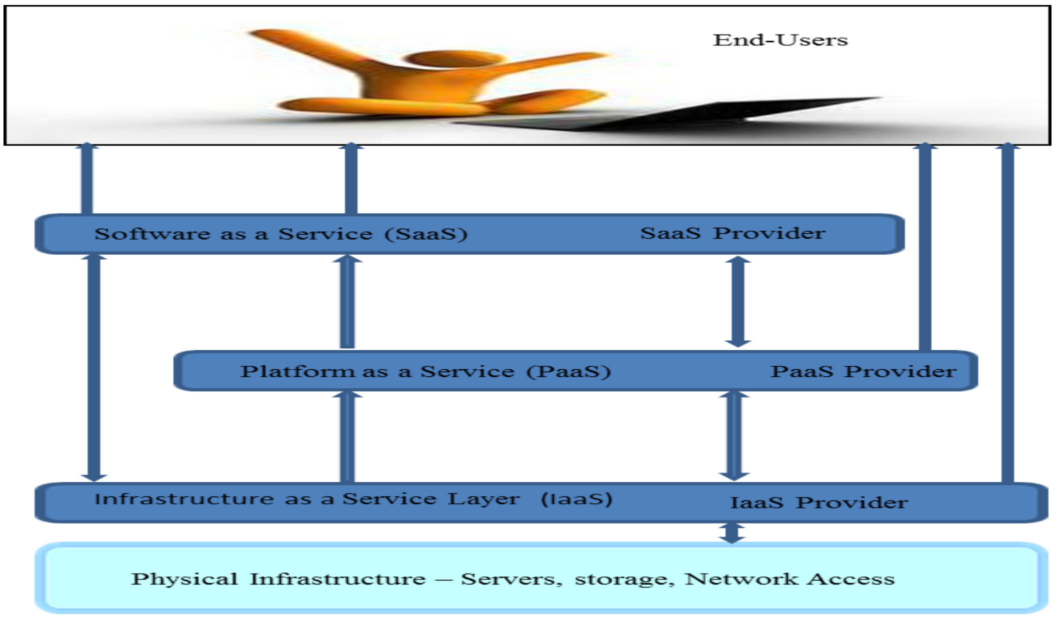

End-Users
Software as a Service (SaaS)
SaaS Provider
Platform as a Service (PaaS)
PaaS Provider
Infrastructure as a Service Layer (IaaS)
IaaS Provider
Physical Infrastructure – Servers, storage, Network Access

Figure 5.1: Cloud service delivery models and trust interaction

For the IaaS layer, the physical environment of the host (server) may be protected by a Data-Cetre-Policy (DCP), which ensures that the server host is protected against power-outages, loss of connectivity and even limits access to authorised personnel. Typically, SLA may be used to monitor and guarantee the enforcement of an adequate DCP. Furthermore, the server (host) computer which is part of the IaaS, may be secured using a suitable mechanism such as TC, which refers to the TPM dependent chain of trust (CoT) that is built from the cryptographic storage of measurements for the various component parts of a computing platform including BIOS, boot-loader, O.S. kernel, system libraries and/or virtualization middleware.

In the process of TC verification, the current set of measurements is compared against preset values using a verifier application such as openPTS, which can also act as a collector [103]. Finally, most implementations of the virtualization engine (part of the IaaS layer) include an intrusion detection security mechanism known as security zone or security groups that serve mainly to isolate or prevent unauthorized communications or interactions between instances that belong to different security groups or zones.

In the PaaS layer, the virtual machine may be protected by a suitable mechanism such as software based virtual TPM (vTPM) that provides TC like protection for the virtual machine or instance. That is, a chain of trust is built from the cryptographic storage of measurements for the various component parts of the instance including BIOS, boot-loader, O.S. kernel, system libraries and/or applications.

Similar to the TC verification process, the current set of measurements (from the vTPM) is compared against preset values using a verifier application such as openPTS. PaaS layer access to an instance is typically over some secure channel such as Secure Shell (SSH), which includes a direct (peer-to-peer) key-based verification process before securing the communication channel using cryptographic encryption [104].

In a high security context, an instance may also include an Intrusion Detection Engine (IDE) for ensuring that contents (applications, files and data) of the instance have not been tampered with or have only undergone authorised modification or changes.

At the SaaS layer, connections to services are commonly protected using the secure socket layer (SSL), which is based on the use of certificates for both identification and securing the communications channel cryptographically.

Self-signed SSL certificates may be self-signed for arbitrary (take-it-or-leave-it) trust, while certificates from external third party certification authorities provide in-direct trust. For high security, an additional security mechanism such as DC secures the data for cloud-based processing and storage. In the DC implementation (as discussed in Chapter 4), the original data is coloured (via steganographic techniques) using digital bits that can uniquely identify the data-owner, cloud-service and data-recipient which can help in tracing the path through which a data loss or theft happened in the event it happens.

For any cloud platform to be secure and trusted, the individual layers (IaaS, PaaS and SaaS) of the platform must be secure. As may be deduced from the above descriptions, there is no “one fit all solution” for securing all the layers [105]. This work derives a unified trust value for a cloud platform from the fuzzy combination of security states of eight different security mechanisms across all cloud layers. The approach is based on attribute / identity trust with elements of direct and in-direct trust. The choice of fuzzy logic is informed by the ability of fuzzy logic to allow representation of any information with some varying (non-crisp) degree of membership [106].

The results from this research show that an accurate trust value may be obtained in as little as 4 transactions for low security platforms to up to 8 transactions for high security ones, this appears to be faster than other comparable models.

The trust model presented in this research may be used to evaluate relative trust derivable from diverse security mechanisms configured on a cloud platform and may also be used as a reference or index tool for comparing the relative trust of cloud platforms. Section 5.2 presents an analysis of related work. Section 5.3 discusses trusted computing and fuzzy logic, the proposed model is discussed in section 5.4 and its implementation and testing is presented in section 5.5. The results are evaluated in section 5.6 and section 5.7 concludes the chapter.

5.2 Related Work

Security and trust related research though not new is still an emerging field in cloud computing. Considering that cloud computing itself is an evolving and unique technology, serving a variety of users with various needs and demands, a single security architecture may be impossible to achieve [5], [27]. Most cloud security related research appear to focus more on the IaaS or related layers with limited or no considerations for other layers that make up the cloud platform.

Researchers in [107] used a subjective logic approach to evaluate trust. While researchers in [108] provided a framework without adequate information about its implementation. Wu in [109] ascertained that a platform can only be secured when all players or stakeholders put their heads together and used a fuzzy reputation for trust management in cloud based on detection of malicious attacks with some set of metrics. Fan et al in [110] considered objective and subjective trustworthiness, with subjective trust based on SLA(s) and some quality of service (QoS) attributes. The approach involved users going through some third party trust providers. The researchers in [111] though considered a trust evaluation system using hierarchical fuzzy inference system for service selection, they only considered infrastructure as a service (IaaS), they didn’t consider the other services.

Researchers in [112] also considered evaluating trust in a cloud computing environment but the system was evaluated using past experiences of previous customers by assigning a reliability weight to customer’s feedback but a high reliability weight doesn't necessarily mean the platform is trusted and secure at later time.

My approach instead focuses on direct monitoring by end-users themselves as this eliminates the doubts that trust ratings are without bias and/or may not completely relevant to the end-user's desired scenarios.

Researchers in [113] presented a trust based approach using trusted computing which is applied only at the IaaS layer and as earlier mentioned a cloud end-user also wants security mechanisms configured across all the layers of a cloud platform to enable the end-user make a more informed decision. The researchers in [114] proposed data access control mechanism but the security considerations were not sufficient.

Researchers in [115] extended the trusted computing chain of trust from the physical infrastructure domain (or IaaS layer) to the PaaS Layer. It is clear that a trust value is only obtainable on cloud platforms that have implanted this extension. While researchers in [54] calculated trust based on historic, direct and recommended (in-direct) values, they do appear to consider attributed trust that may be derived from the identified behaviour of the platform. Also, their proposed model does not appear to cater for diverse end-users needs and requirements.

In general, it may be stated that the overall trustworthiness of any cloud resource may be derived from an algorithmic sum of trust levels (security measures) that are enforced or available across all layers or parts of a computing system [100].

In the case of cloud computing, the ability to probe and test on-demand the trustworthiness of all assigned or consumed computing resources would make end-users confident of the platform as it provides them with additional assurance of data-integrity both during storage and processing.

The research work on trust presented in this chapter involves the ability of a user to evaluate the trustworthiness of the resources on a cloud platform using fuzzy logic.

The model was evaluated and compared against two cloud trust models they are Dynamic multi-dimensional trust model (DMTC) as presented by [116] and Trust model based on fuzzy mathematics (TMFM) as presented by [54]. DMTC dynamically reflects a trust relationship in a cloud system by calculating direct trust based on time evaluation and space evaluation for recommended trust.

TMFM computes and evaluated the trust status of a platform based on fuzzy mathematics. The evaluation is based on direct observation between entities.

5.3 Trusted Computing and Fuzzy Logic Theory

Generally, trust in computing tend to be modelled after human relationships as users tend to exchange and share ideas based on experiences thereby indirectly or subconsciously building an impression of the services based on previous users' experiences and in turn any perceived service. Users would now tend to form an impression based on the previous user's experience after interacting with the old user so even if the new user hasn't had the experience he forms his impression based on the previous user [99], [100], [117]. The concept of trust may also be applied to the area of service delivery, for example, a party may "trust" another party to deliver quality service, in which case trust becomes a measure of the service-availability. In the case of security, if the party's trust is based on the availability of some security measures, then trust becomes a measure of the security available and attested to. Authors in [88] classified trust into objective and subjective trust as known as direct and in-direct trust. Objective trust is obtained from the direct interactions between two parties; while, subjective trust involves the impressions obtained from third parties.

As shown in Figure 5.1, trust in cloud should be examined over four different layers (Physical, IaaS, PaaS and Saas). In this work, trust at the physical layer is derived from the defence capability of individual physical devices. A single cloud host (physical server) which is an IaaS component may be secured using the TPM based Trusted Computing (TC) industry standard[103]. As shown in Figure 5.2 [115], a chain of transitive trust is established from hardware (core Root of Trust) to the software layers (virtualization) across the BIOS device, the boot-loader and operating system.

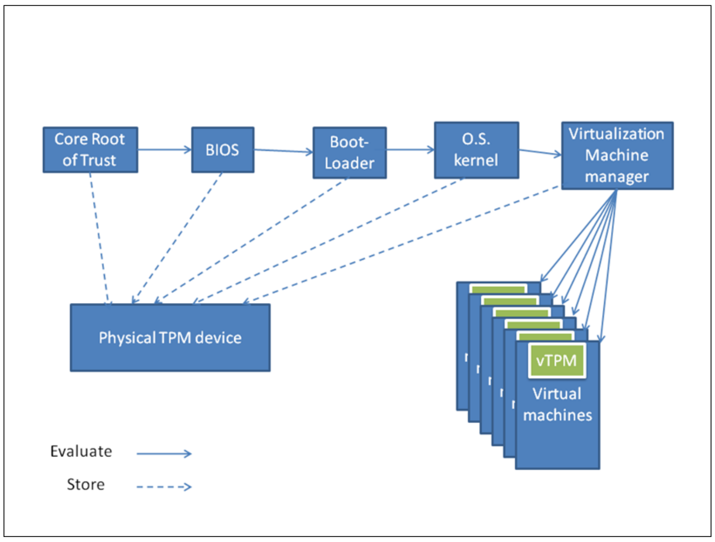


Figure 5.2: Linear and tree based on trusted computing of a single host

In Figure 5.2, the chain of trust that exists between the core root of trust to the virtualization machine manager layer may be modelled based on the linear transitive trust principle where the final trust is the minimum offered by each intermediary object that is part of the chain. Each stage of the TC chain (cRMT to VMM) shown in Figure 5.2 may be represented as a node in an N node graph system where the trust level between two adjacent nodes would be given by the expression [115]:

$$T(A,B) = \min\{T(A,B), T(B,C), T(C,D), \ldots, T(N-1,N)\} \qquad (5.1)$$

Similarly, the trust relationship between the virtualization machine manager and the virtual machines is a one-to-many relationship of direct trust and may be modelled by a tree graph where the trust relationship is the maximum trust level obtained from each directly connected object. That is, assuming a one-to-many node ($VMM\ to\ vm_i$) system where T denotes the Trust level between two adjacent levels.

$$T(VMM,\ vm) = \max\{T\ (VMM, vm_1), T(VMM, vm_2), \ldots, T(VMM, vm_i)\} \qquad (5.2)$$

While, the virtual machines may include a virtual TPM device as shown in Figure 5.2, their virtual nature precludes their consideration in the overall security and trust of the IaaS layer. Typically, the IaaS layer is a collection of physical resources (such as hosts) that may be co-located in a single data-centre or distributed across geographically separated data-centres and the net trust expected from the IaaS service layer is obtained from the multiplicative sum of the trust levels of individual physical hosts and resources. It is clear that the presence of a single compromised or unsecured host would negatively affect the trust of all hosts in a data-centre or the IaaS layer.

$$T_{IaaS} \text{ layer } = min\ (T_{R1},\ T_{R2}, \dots, T_{Rn}) \tag{5.3}$$

In many cloud implementations, the cloud PaaS layer is mainly secured using either zones or firewalls and in some cases both are implemented at the Cloud middle-ware and the trust replications may thus be modelled by the equation with the number of virtual machines limited to the finite set of vm(s) sharing the same zone. In addition, trust at the PaaS layer may be complemented by a defence capability such as the inclusion of an IDS engine directly into the PaaS software-environment. In which case, the IDS is used to provide a baseline control against which the contents (files and data) of the PaaS system may be compared to detect unauthorised modifications or updates to the PaaS software-environment caused by trojan horses, viruses or other malicious activities. The IDS engine is typically a separate application that runs within a vm, consumes resources and provides a measure of the one-to-many (tree) direct trust of other applications within the vm. That is, the IDS trust may be modelled by:

$$T(IDS, app\ ) \ = \max\ \{\ T(IDS, app_1), T(IDS, app_2), .., T(IDS, app_n)\} \tag{5.4}$$

where $T(IDS, app_n)$,represents the trust measure provided by the IDS engine for a particular application. The PaaS environment is typically created from a software hierarchy consisting of a boot-loader, operating system kernel, shell and libraries. The final security of individual applications therefore depends on the security of this underlying software stack. That is:

$$T\ (IDS, app_x) = \min\{T(IDS, bootloader), T(IDS, O.S - Kernel), T(IDS, O.S - shell), T(IDS, O.S - libraries)\} \tag{5.5}$$

PaaS layers are typically provided by virtual machines from the underlying IaaS layers and the trust relationship between the IaaS and PaaS layers may be modelled as a trust fusion or average process.

$$T\ (IaaS, PaaS) = \max\{\ T(IaaS, PaaS_1), T(IaaS, PaaS_2), .., T(IaaS, PaaS_n\} \tag{5.6}$$

And for each PaaS

$$T\ (PaaS) = T\ (IaaS_x, PaaS_x) + T\ (IDS_x, app_x) \tag{5.7}$$

Where the "+" sign represents the trust fusion process and $T\ (IDS_x, app_x)$represents the added component provided by the internal IDS engine.

In cloud implementations, the SaaS layer is typically protected by securing the communication layer against man-in-the-middle attacks using a suitable mechanism such as secure socket layer (SSL) connections or secure shell (SSH) connections. Both SSL or SSH represent identity trust relationships based on possession of matched certificates or credentials, however, while SSL may require a 3rd party certification authority, SSH connections represent direct-trust between two parties. SSH connection may be modelled as simple vector, where an unknown host is not trusted while a known or matching host is fully trusted. It is noted that a known but not-matching host is completely untrusted.

$$\text{T (SSH)} = \begin{cases} 0 & if \text{identity} - \text{trust exists but is not matched} \\ 1 & \text{if identity} - \text{trust is matched} \end{cases} \tag{5.8}$$

Although trust in SSL is different from SSH, the simplest form of SSL connections may similarly be modelled by:

$$\text{T (SSL)} = \begin{cases} 0 \text{ if identity} - \text{ trust is NOT verified} \\ 0 \quad \text{if identity} - \text{trust is revoked} \\ 1 \quad \text{if identity} - \text{trust is verified} \end{cases} \tag{5.9}$$

However, the ability to revoke certificates suggest that SSL connections may be better modelled by more complex algorithms such as EigenTrust or PeerTrust where the group of peers is limited to a well-known set of peers or community.

At the application level, data at the SaaS layer may be protected against theft or loss using a suitable mechanism such as hashing, data-colouring, encryption or obfuscation. Data-colouring could be an optimal technique for cloud platforms or applications as coloured data may still be processed without overhead (the colouring is transparent) while also providing the ability to detect tampering, identifying and reporting data-loss. Considering security related metrics of confidentiality, integrity and availability in a cloud context, the coloured-data (output of a data-colouring process) is able to maintain integrity and availability of data during processing. While, hashing as a data-protection technique can only provide integrity, obfuscation provides only limited confidentiality; data-colouring provides integrity and availability; while encryption provides confidentiality and integrity but not availability. In situations where all multiple protection mechanisms are in use, it is possible to assign weights to each based on the relative contribution of the individual mechanism.

The SaaS layer trust may be modelled as a fusion of trusts derived from the security values of its connections and data-protection mechanisms. Typically, a SaaS service requires an underlying PaaS. The trust relationship between a PaaS and a SaaS application is typically a 1-to-1 nature which may be modelled as a discount trust process.

The simulation for this researcher would be implemented to accommodate different scenarios. These scenarios would be deployed to accommodate different levels of trust, this is achieved by deploying different security techniques across different layers and this in turn can be evaluated to reflect the different levels of trust in the platform.

Since no computing system can be completely trusted as every system is “only as secure as its weakest link”, the same can be said of the cloud system; therefore, categorically stating a system is completely trusted or not may not be a true reflection of the system.

Fuzzy logic which is a computing approach that is based on “degrees of certainty” includes cases of trust not just based on 0 and 1 (low or high) but it allows the inclusion of various cases in between such as low to medium, medium to high, etc.

Fuzzy logic makes an aggregate of various parameters and based on certain thresholds makes a decision, it is suitable for modelling uncertainties. Type 1 fuzzy set or type 2 fuzzy set can be applied to the different cases for decision making[118], [119].

In this chapter, fuzzy logic evaluation is used to calculate cloud trustworthiness based on the user’s needs and satisfaction. The user may only want to know the trust state of the IaaS and doesn’t mind about PaaS and SaaS as the user doesn’t need that service or maybe only concerned with the trust state of IaaS and PaaS and not SaaS or vice versa. Using fuzzy logic approach, trust values are combined to enable the user decide the trustworthiness of the platform or services.

5.4 Proposed Model

The security assessment of a cloud platform should be of paramount importance to any user regardless of the service required. Researchers in [108] listed some parameters necessary for measuring the overall security of a cloud platform and the deployed service.

The cumulative crisp sum of security values is used to evaluate the trustworthiness of the cloud platform or the deployed service. In a dynamic world, security is not static but ever evolving, crisp value representation (a 1 which is “present” or a 0 which means “absence”) would only provide a theoretical or expected (or possibly inaccurate) state of the platform. In this section, we propose a multi-layer security trust model (MLSTM) that is based on FLC system with the following characteristics [120]:

- State changes are based on Gaussian fuzzy numbers
- Various operators are used to represent the rules
- The overall control action of the system is computed to reflect the accumulated security strength.

As shown in Figure 5.3, the proposed Multi-Layer Security trust model (MLSTM) cuts across the well-known layers of a cloud platform – IaaS, PaaS and SaaS.

Trust evaluation at any layer is derived from the identified behaviour of individual security mechanisms at distinct layers of the platform. For the IaaS layer, the trust value is obtained from a FLC system that combines verifications of the TC, ID security mechanisms and an adequate DCP. Similarly, at the PaaS layer, trust value is obtained from the FLC combination of verifications of the SSH, IDE and vTPM security mechanisms, while the SaaS layer trust value is obtained from the FLC combination of SSL and DC security mechanisms.

In our model, the end-user is presented with a single trust value derived from the FLC combination of trust values from IaaS, PaaS and SaaS layers which may be used to decide if platform is trustworthy or not.

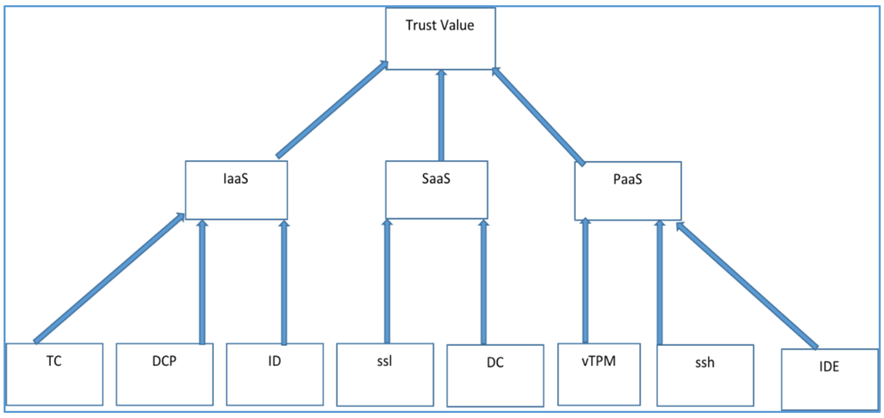


Figure 5.3: Multi Layer Security Trust Modell (MLSTM) Concept

Fuzzy sets are different from simple every day probabilities as with probability the even would happen or it would not happen leading to a crisp value of true or false – 1 or 0, therefore the operators of the classic set theory need to be redefined to fit membership functions for values between 0 and 1. Crisp representation imposes a sharp boundary on a set where each member of a set is assigned 1 while a member outside the set is assigned a value of 0, that is in crisp representation an element belongs to a set or not.

$$x \in \mathrm{X}\ is\ true\ for\ 1 \tag{5.10}$$

or

$$x \notin X \text{ is true for } 0 \tag{5.11}$$

In the crisp interpretation, a given cloud (x) is secure if it is a member of the set of secure clouds (X) as presented by equation (5.10), while the converse is true that is a given cloud platform is NOT secure when it is NOT a member of the set of secure clouds (X) as shown in Equation (5.11).

However, practically, it is possible that a sub-set part of a cloud platform to be secure (member of set X) while some other part of the same cloud service is not secure (not a member of X).

In Set Theory, the intersection of two crisp sets is composed only of the elements that are present in both sets while the union of the sets is derived from elements that are present in either of the sets. For example, in digital electronics, binary logic circuits can assume distinct (crisp) states of 1 or 0, the intersection of the two states in the AND logic gate results in the lower state.

While the union of the two sets which may be illustrated by a logical OR gate operation in digital electronics would take the higher value of the sets as shown in Table 5.1

Table 5.1: Binary Logic Representation of Intersection and Union

Intersection **(Logical AND)**	**Union** **(Logical OR)**
1 AND 1 = 1	1 OR 1 = 1
1 AND 0 = 0	1 OR 0 = 1
0 AND 1 = 0	0 OR 1 = 1

In fuzzy logic, membership of a given element in a set is determined as a fractional value between 0 and 1 known as the degree of membership, which conveys an idea of much of that element is contained within a given set.

It is possible to define an arbitrary minimum membership value say 0.5 that should be increasing for a cloud platform to be indeed trusted. The degree of membership function would be thus:

$$2\,[\,\mu_A(s)]^2 \qquad if\ 0 \leq \mu_A \leq 0.5$$

$$1 - 2\,[1 - \mu_A(s)]^2 \qquad if\ 0.5 \leq \mu_A \leq 1$$

(5.12)

With the Gaussian membership function:

$$\mu_{Ai}(x) = \exp(-\frac{(c_i - x)^2}{2\sigma_i^2}) \qquad (5.13)$$

In Equation 5.13, σ is the standard deviation and c is the centre of the ith fuzzy set of A_i . the membership function always returns values in the range of 0 and 1.

The degree of membership from Fuzzy Logic can be used to support vague concepts and model real world situations including the dynamic evolution/changing nature of security of a cloud platform with much higher accuracy compared with a crisp representation. That is, with fuzzy representation, it becomes possible to say a given cloud platform is x% secure or y% unsecure.

Alternatively, a cloud platform cannot be said to be completely secure, it may be secure to a certain degree or level even when the components that make up the system are assumed to be fully secured or completely unsecure, this also means a cloud platform must be able to offer real-time security [121].

Based on fuzzy logic, the degree is usually a real number between the range of 0 and 1. While, the crisp representation of cloud security can provide a binary ("True" or "False"; 1 or 0) answer to the question of "Can I trust the cloud platform?", the fuzzy representation goes further and can provide an answer to the question of "how trustworthy is the platform?" even when presented with diverse or varying requirements.

With fuzzy logic, any given cloud platform would have a varying degree of membership in two distinct universal sets of secure-clouds and unsecure-clouds. For an element with varying degree of membership in two different sets, the membership value in the resulting intersection (fuzzy AND) of both sets would be the lower of both membership values, while the membership value in a union (fuzzy OR) of both sets would be the higher value.

That is, the fuzzy OR operation is given by:

$$\mu_{A\cup B}(x) = \max\{\mu_A(x), \mu_B(x)\} \tag{5.14}$$

While, the fuzzy AND operation is given by:

$$\mu_{A\cap B}(x) = \min\{\mu_A(x), \mu_B(x)\} \tag{5.15}$$

Figure 5.4 shows the four major components of a FLC system, which according to [19] are:

- The fuzzification interface acquires the values of input variables and performs a scale mapping that transfers the range of values of inputs variables into corresponding universes of discourse, this is the range of all possible values for an input to a fuzzy system and converts input data into suitable linguistic values which may be viewed as label of fuzzy sets.
- The "linguistic (fuzzy) control rule base", provides necessary definitions which are used to define linguistic control rules and fuzzy data manipulation, the rule base characterizes the control goals and the control policy of the domain experts by means of a set of linguistic control rules.
- The fuzzy inference engine is the kernel of a FLC; it simulates a process similar to that of of human decision making based on both fuzzy concepts and inferring fuzzy control actions from the rules of inference in fuzzy logic. The inference type used in this research is the Mamdani-type inference.
- The defuzzification interface performs a scale mapping of the range of values of fuzzy output variables into corresponding universes of discourse, defuzzification to obtain a non-fuzzy control action from an inferred fuzzy control action and operates to transform fuzzy sets into crisp data sets.

The defuzzification process may be represented by the expression:

out = defuzz(x,mf, type) (5.16)

where defuzz returns a defuzzified value based on the membership function “mf” at an associated variable of value “x” and according to an argument “type” which for this research the centroid type was used.

In summary, the processing of rules (fuzzy conditional statements) in an FLC system is based on fuzzy sets, and any crisp inputs need to be “fuzzified” for correct processing and produces a fuzzy output which is then “defuzzified” to obtain a crisp value. The FLC system presented here is composed of a set of rules (conditional statements and algorithms) characterised to represent simple and complex relations in the form:

$$IF\ x_1\ is\ A\ and\ x_2\ is\ B\ and\ x_n is\ C\ \ldots THEN\ y\ is\ D \quad (5.17)$$

The fuzzy conditional statement in Equation (5.17), may be interpreted generally as IF a set of conditions is satisfied THEN a set of consequences can be inferred.

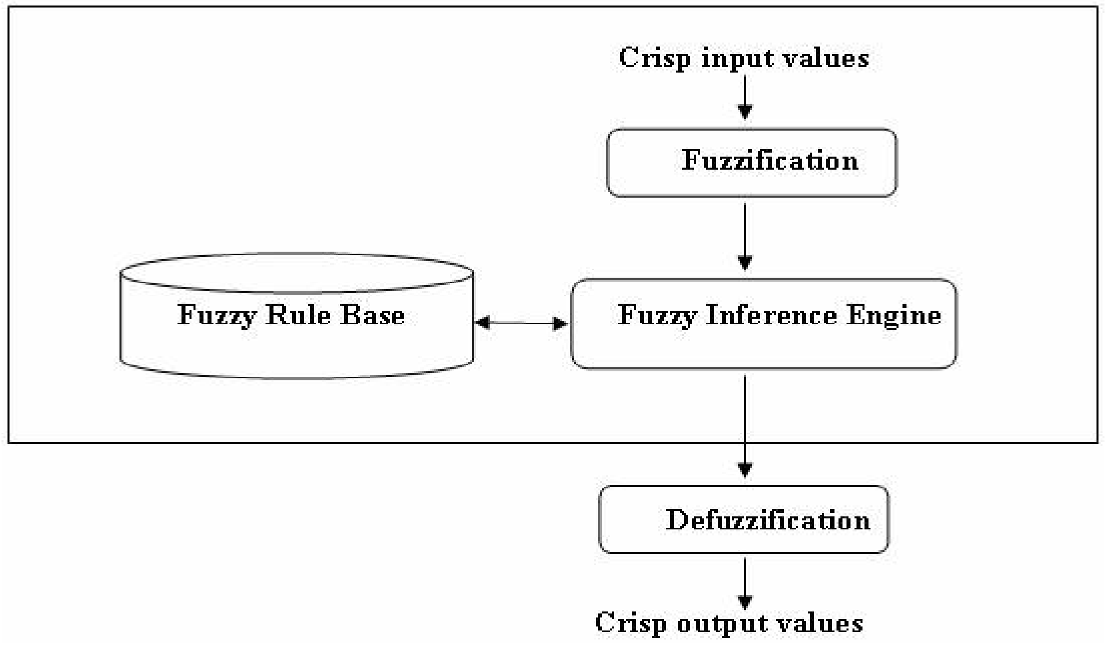


Figure 5.4: Functional / Block Diagram of a Fuzzy Logic Control (FLC) System (Source: [122])

A fuzzy rule system has multiple parts which is unlike classical rule based system where the result is always true or false (1 or 0). In a fuzzy system, all parts of the antecedent are calculated simultaneously and results in a single number using fuzzy set operations. The fuzzy operators AND or OR are applied to obtain a single number depending on the type of evaluation.

As mentioned earlier, due to the multilayer nature of the cloud infrastructure, each layer of the cloud platform may implement different security measures each of which should to be individually secure and aggregated at each layer even for complex configurations such as a user who is interested in accessing a cloud platform with the following specific configurations: trusted computing (TC), intrusion detection (ID), data centre policy (DCP); ssh, intrusion detection environment (IDE) and vTPM but not bothered about the presence or absence of data colouring.

Based on fuzzy intersections, the corresponding linguistic values of the input assigned a degree, which is a product of all its antecedents and consequent memberships. The fuzzy set provides a continuous transition across the input range based on the order weighted averaging of the associated set of weights $W = (w_1, w_2, \dots w_n)$ of each security configuration is such that $w_i \in [\,0{,}1\,]$ and $\sum_{i=1}^{n} w_i = 1$ its computed and represented thus:

$$T = \sum_{j=1}^{n} w_j s_j \tag{5.18}$$

So for the cloud platform let the reference set be S $\{s_1, s_{2,} s_{3,} s_{4,} s_{5,} s_6\}$, A is the crisp set that can only take two values 0 or 1, which would represent each of the configurations in IaaS, PaaS and PaaS and each can be represented thus:

$$A = \{(s_x, \mu_A(s_x))\} \tag{5.19}$$

The multi-layer security trust model (MLSTM) is based on the use of fuzzy logic combination of a controlled sequencing of specific transactions that examine various security mechanisms specific to the cloud layers of IaaS, PaaS and SaaS. The MLSTM is a tool for assessing and evaluating the diverse security concerns related to cloud services and can provide users with the ability to evaluate the security of a chosen cloud platform as part of the process of establishing trust.

The input variables for the system as noted above are IaaS, PaaS and SaaS. The value of IaaS is calculated from the following security configurations: *trusted computing, intrusion detection, data policy;* while that of PaaS is calculated from *ssh, intrusion detection environment, virtual TPM* and SaaS is calculated from *ssl* and *data colouring* and the output is the security value which the user is able to make decision as to “how secure or trustworthy is the system?”

Furthermore based on Equation 5.17, the rule base for the overall system consists of 27 rules which are represented in Table 5.2. There are three linguistic values for the input variables (IaaS, PaaS and SaaS) which are *low, medium and high* and six linguistic values for the output variable (security): *extremely low, very low, low, medium, high and very high.*

Table 5.2: Fuzzy Rule base Table for all the Layers

	IF			***THEN***
Rule No	*IaaS*	*PaaS*	*SaaS*	*Security*
1.	low	low	low	ex.Low
2.	low	low	med	v.low
3.	low	low	high	v.low
4.	low	Med	Low	low
5.	low	med	med	low
6.	low	med	high	low
7.	low	high	low	low
8.	low	high	med	low
9.	low	high	high	low
10.	med	low	low	low
11.	med	low	high	low
12.	med	low	med	low
13.	med	high	low	med

14.	med	high	high	high
15.	med	high	med	med
16.	med	med	low	med
17.	med	med	high	low
18.	med	med	med	med
19.	high	high	high	v.high
20.	high	high	med	high
21.	high	high	low	med
22.	high	low	med	med
23.	high	low	high	med
24.	high	med	med	med
25.	high	med	high	med
26.	high	med	low	med
27.	high	low	low	med

Furthermore, the rule base for the individual layers – IaaS, PaaS and SaaS are represented in Tables 5.3, 5.4 and 5.5 where the input variables can only have the linguistic values of *low* or *high* and the output variable has the linguistic value of *low, medium* and *high.*

Table 5.3: Fuzzy Rule base Table for the IaaS Layer

IF				***THEN***
Rule No	TC	ID	DCP	IaaS
1	low	low	low	low
2	low	low	high	low
3	low	high	low	low
4	high	low	low	low
5	high	low	high	med
6	high	high	low	med

7	low	high	high	med
8	high	high	high	high

Table 5.4: Fuzzy Rule base Table for the PaaS Layer

IF				***THEN***
Rule No	SSH	IDE	vTPM	PaaS
1	low	low	low	low
2	low	low	high	low
3	low	high	low	low
4	high	low	low	low
5	high	low	high	med
6	high	high	low	med
7	low	high	high	med
8	high	high	high	high

Table 5.5: Fuzzy Rule base Table for the SaaS Layer

IF			***THEN***
Rule No	SSL	Dcol	SaaS
1	low	low	low
2	high	low	med
3	low	high	med
4	high	high	high

For a user who is interested in accessing a cloud platform with the following configurations: TC, ID, DCP; *ssh,* IDE and *vTPM* but is not bothered about *data coloring*, the FLC system is analysed in Table 5.6, where the corresponding linguistic values of the inputs (IaaS, PaaS and SaaS), are combined using fuzzy (fired) rules into a trust value and their corresponding fuzzy levels computed with t-norm product to obtain the corresponding crisp values. The computed crisp trust value is 0.7323.

Table 5.6: FLC System Analysis Table

Cloud layer	**Membership (μ) range**	**Fuzzy level**	**Crisp value**
IaaS	0.5 – 1.0	Medium	0.5927
PaaS	0.5 – 1.0	Medium	0.5935
SaaS	0.0 – 0.5	Medium	0.6324
Trust		High	0.7323

Figures 5.5, 5.6 and 5.7 show the membership function graphs for the individual variable input of the security mechanisms configured on the IaaS layer and Figure 5.8 show the membership function for the output variable of IaaS after the rules have been applied.

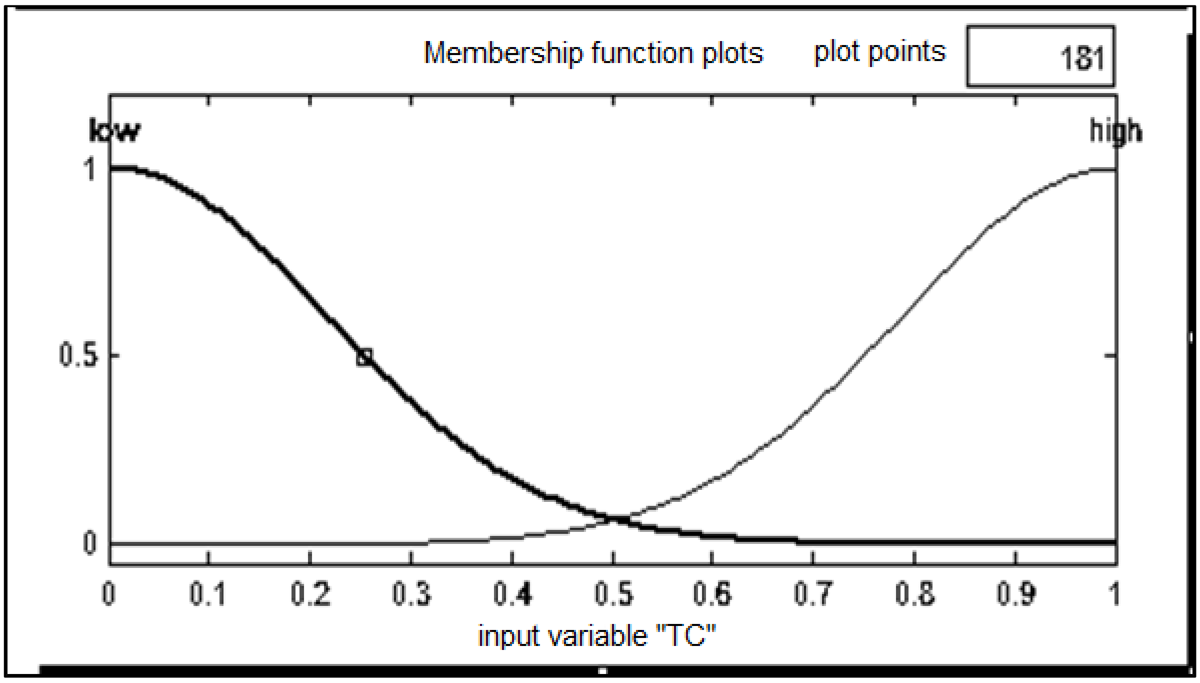


Figure 5.5: Membership Function graph for the input variable TC

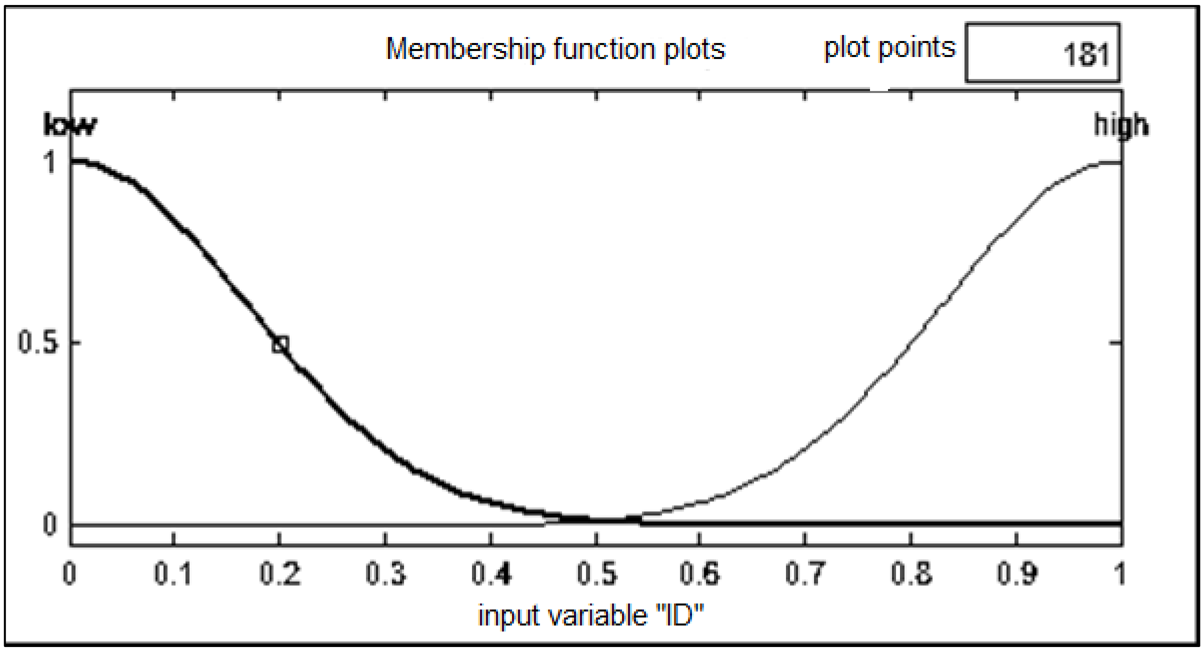


Figure 5.6: Membership Function graph for the input variable ID

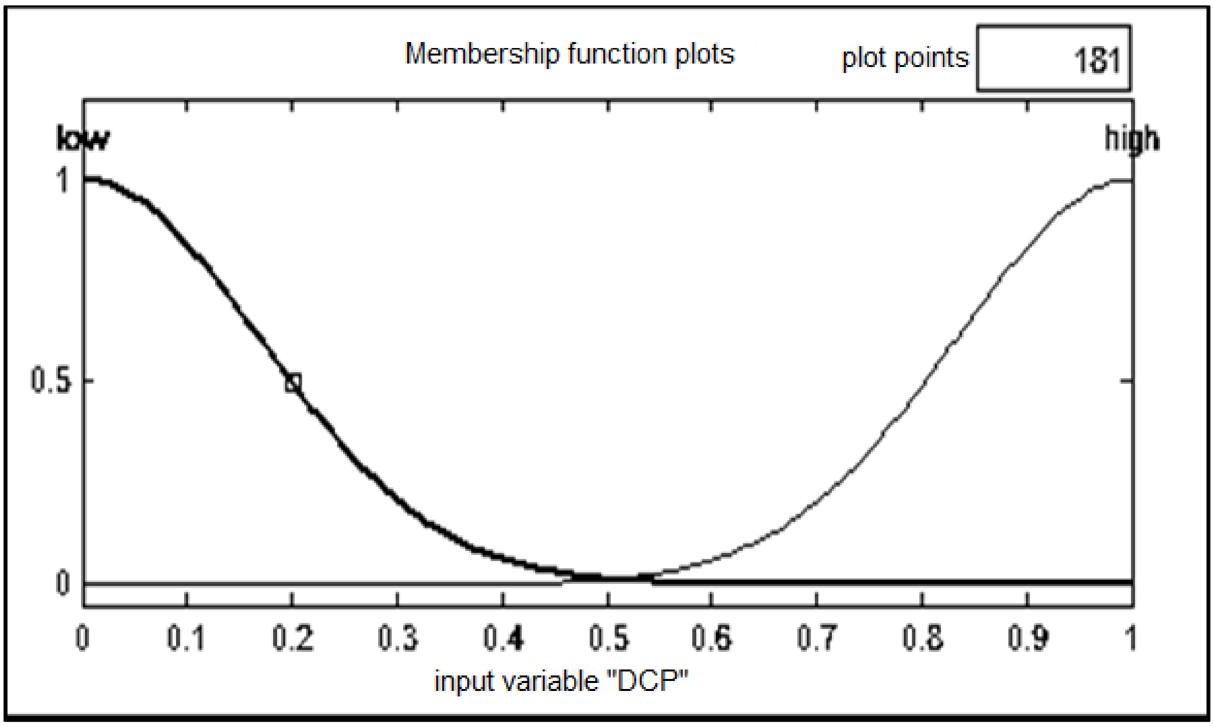


Figure 5.7: Membership Function graph for the input variable DCP

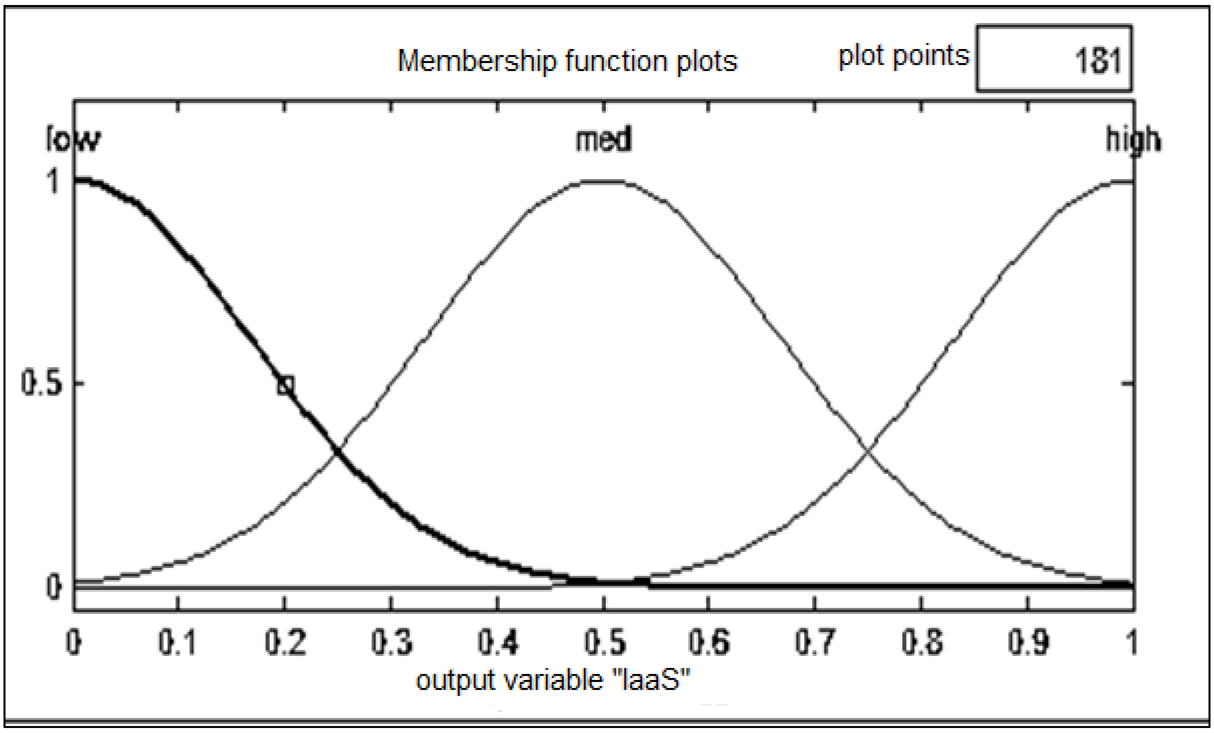


Figure 5.8: Membership function graph for the output variable IaaS

Figure 5.9 show the membership function graph of the output variable PaaS after the rules have been applied on the input variables of the security mechanisms on PaaS. The individual membership function graphs of the input variables are shown in Figures 5.10, 5.11 and 5.12.

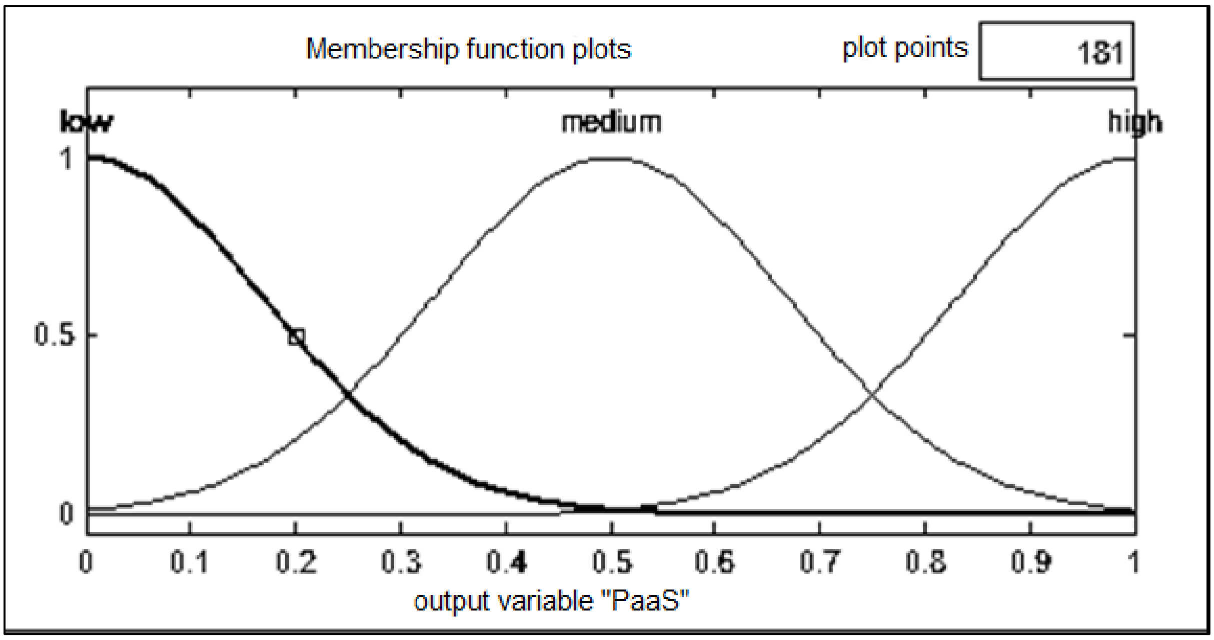


Figure 5.9: Membership function graph for the output variable PaaS

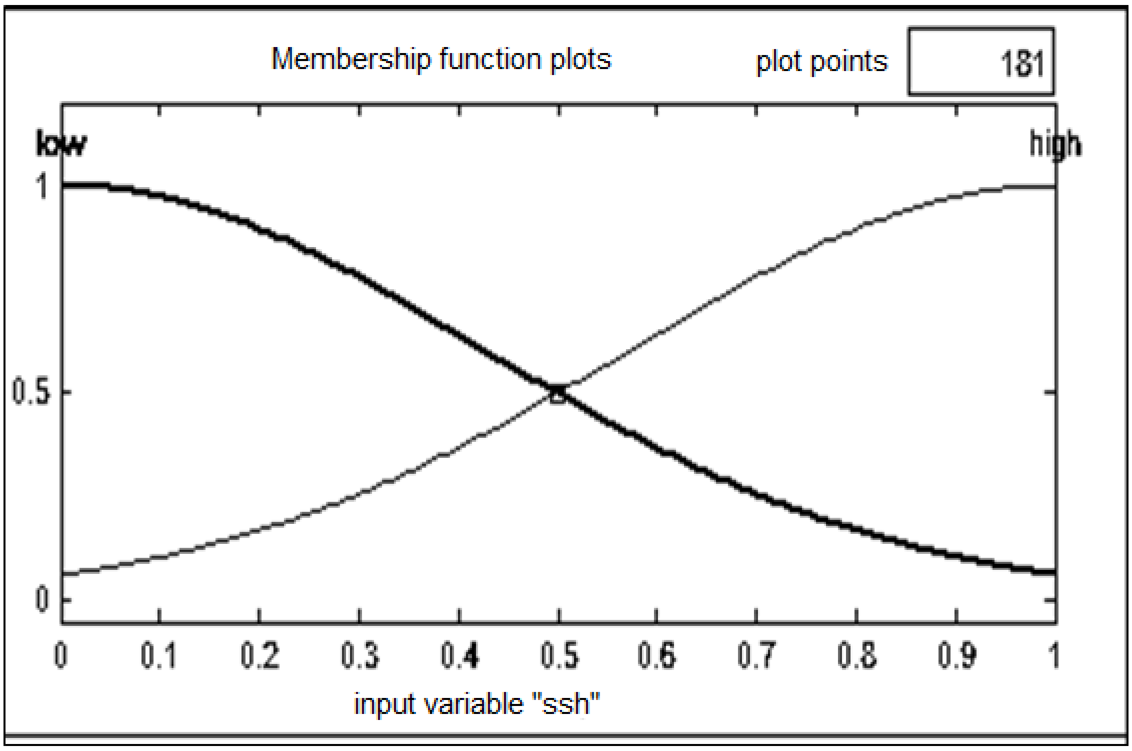


Figure 5.10: Membership function graph for the input variable ssh

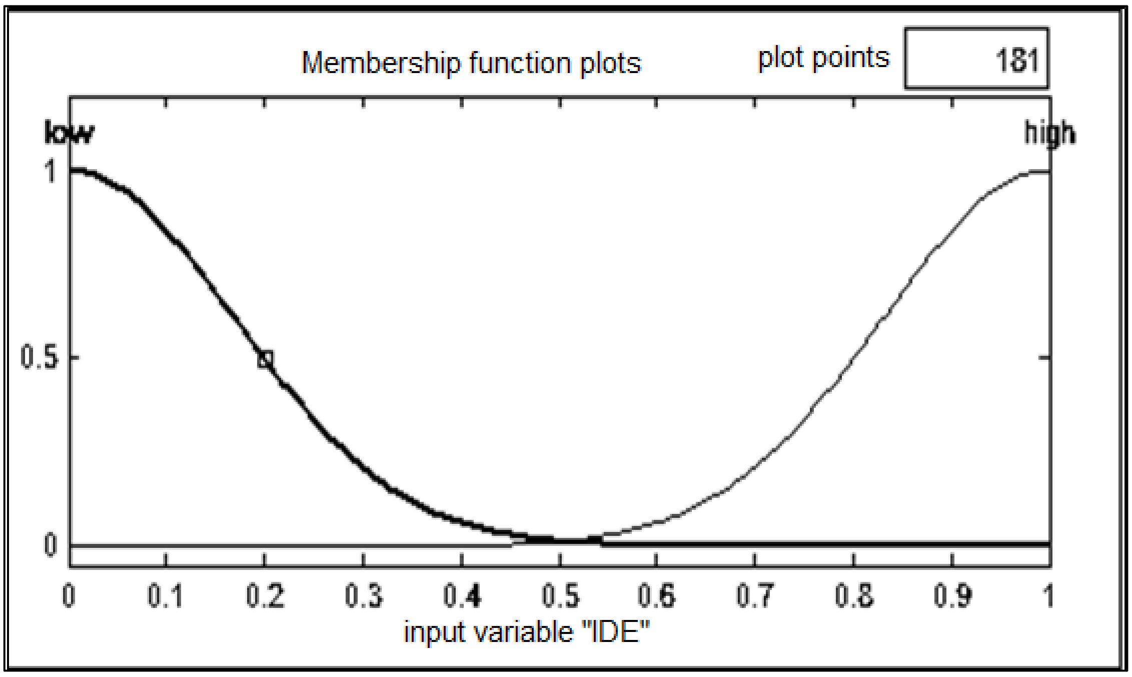


Figure 5.11: Membership function graph for the input variable IDE

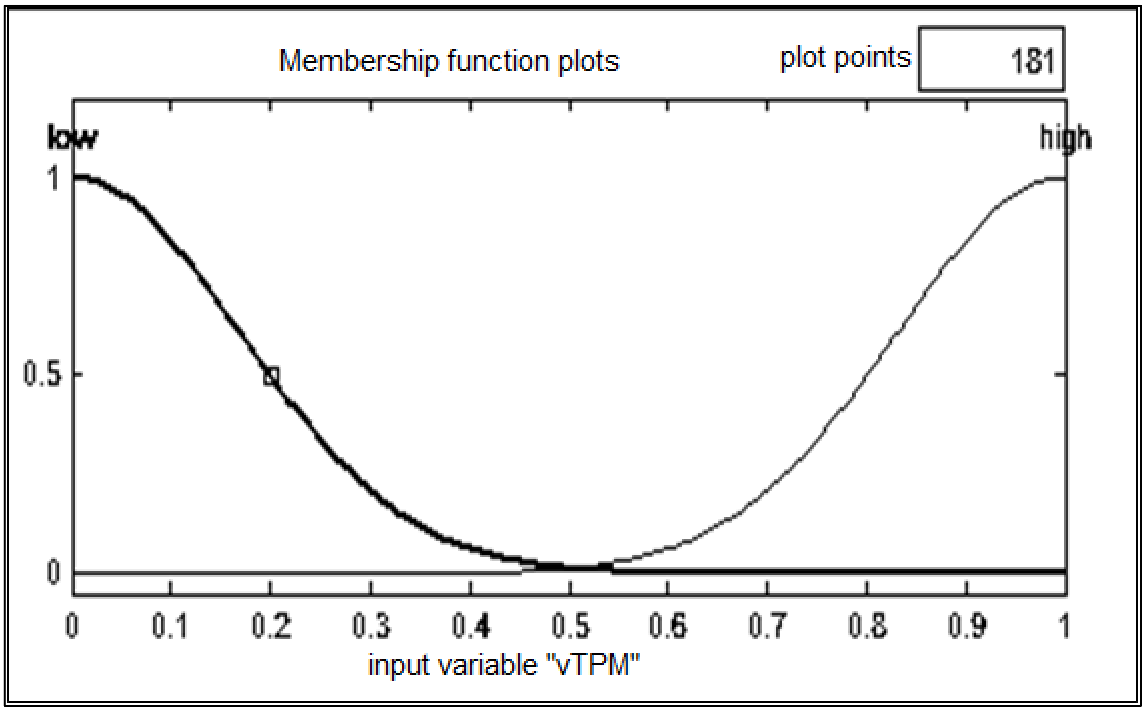


Figure 5.12: Membership Function graph for the input variable vTPM

The membership function graph in Figure 5.13 is that of the output SaaS and the graphs in Figures 5.14 and 5.15 are for the individual security mechanisms configured on the SaaS layer.

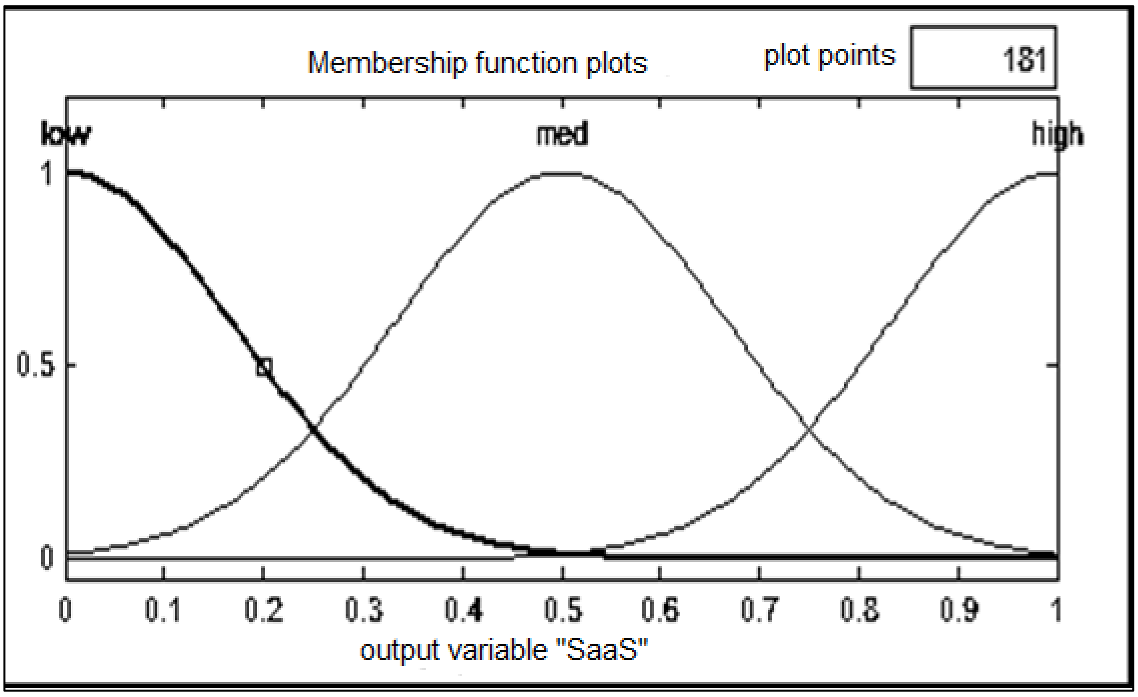


Figure 5.13: Membership Function graph for the output variable SaaS

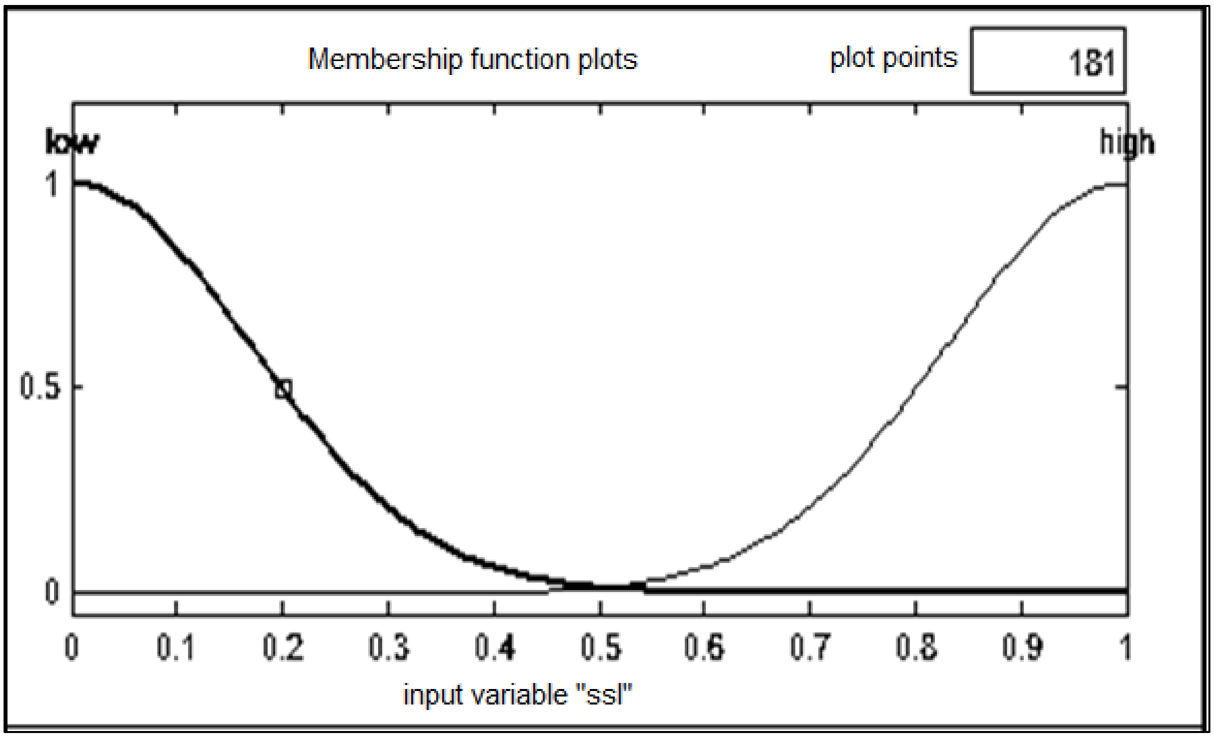


Figure 5.14: Membership Function graph for the input variable ssl

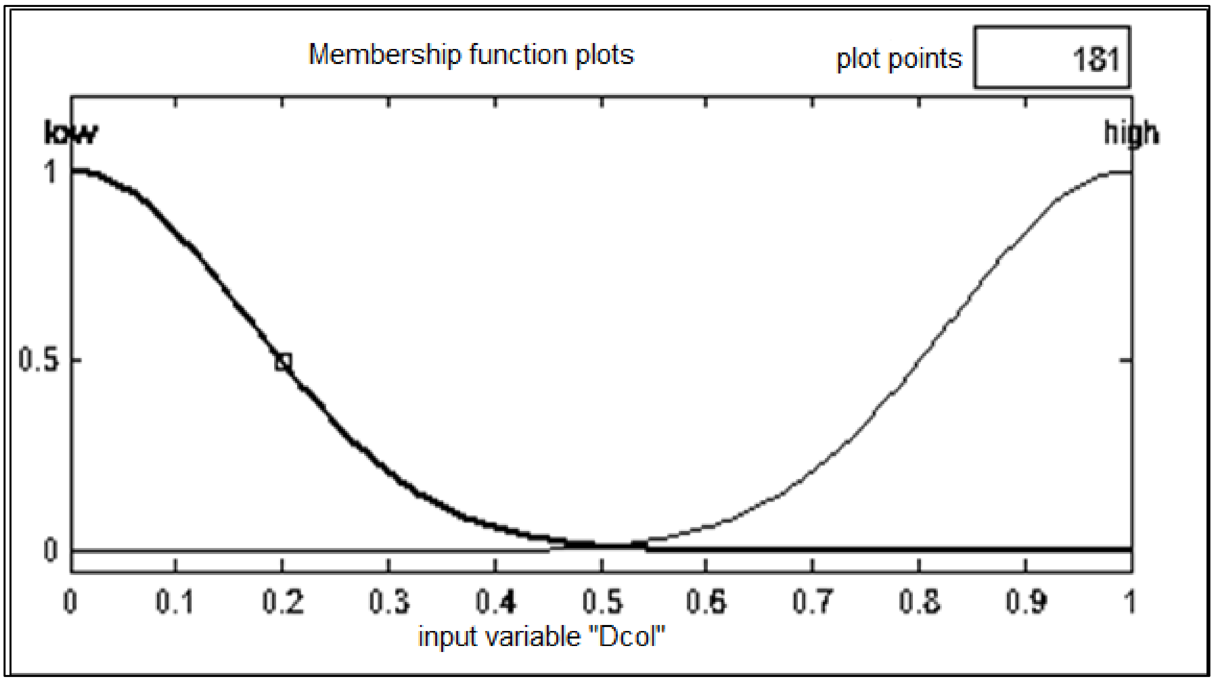


Figure 5.15: Membership Function graph for the input variable Dcol

Figures 5.16, 5.17 and 5.18 are membership function graph of the IaaS, PaaS and SaaS as input variables while Figure 5.19 show the membership function for the output variable security, which what provides the final crisp overall value of the system.

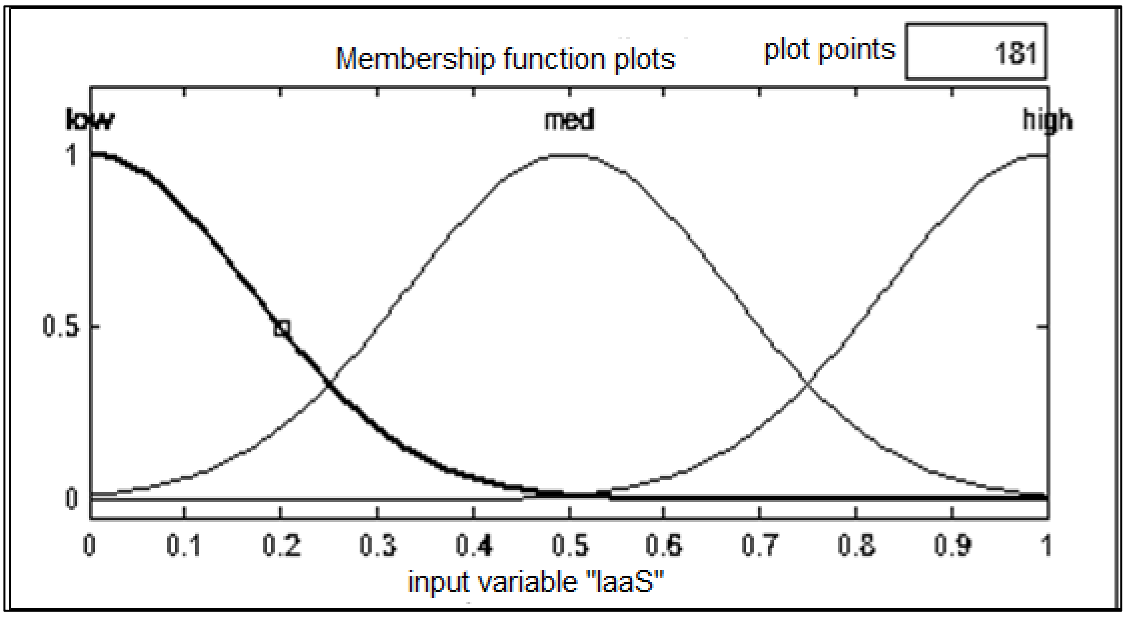


Figure 5.16: Membership Function graph for the input variable IaaS

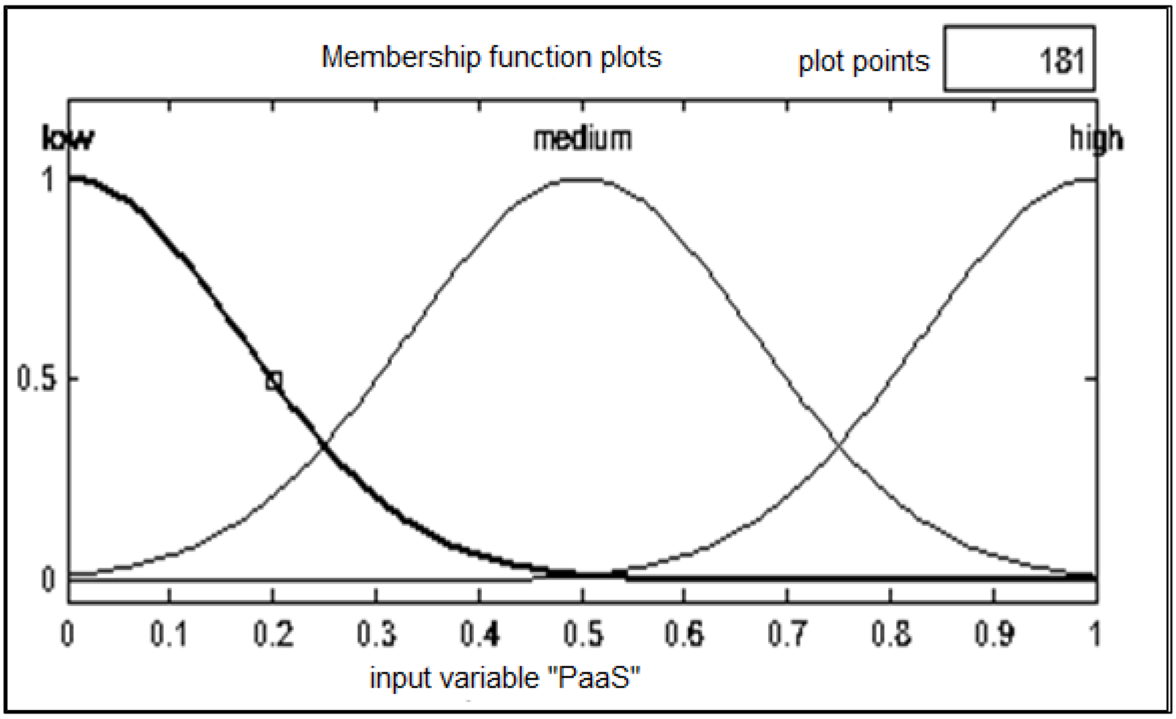


Figure 5.17: Membership Function graph for the input variable PaaS

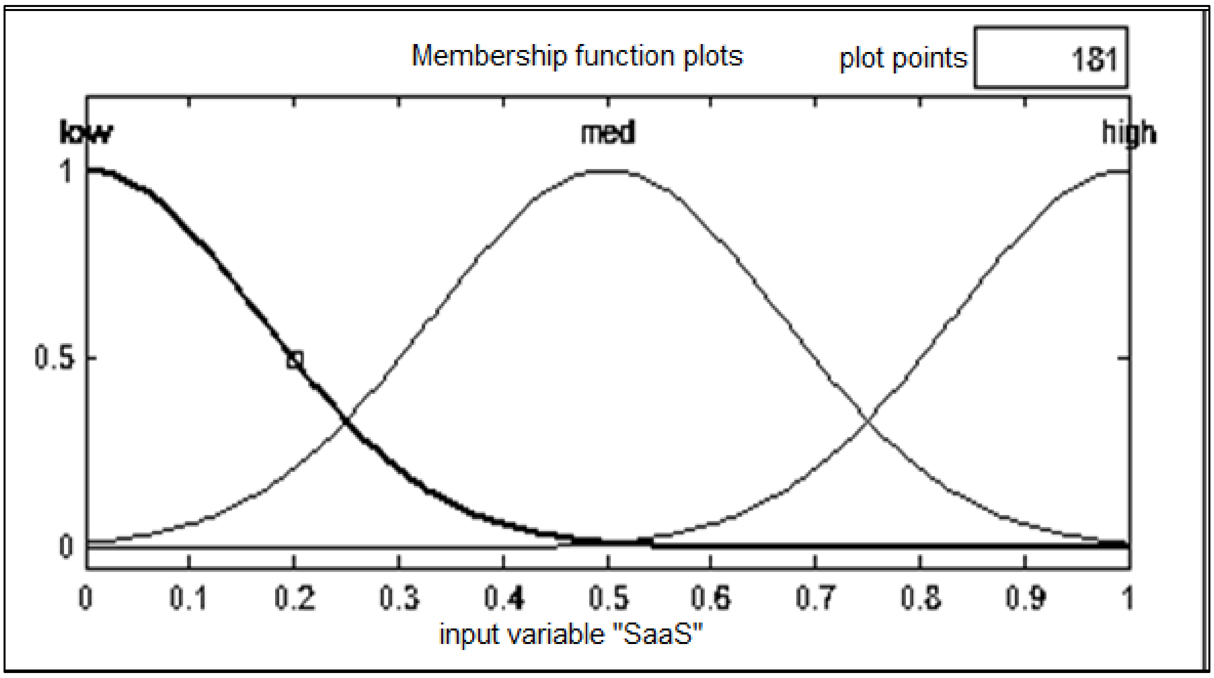


Figure 5.18: Membership Function graph for the input variable SaaS

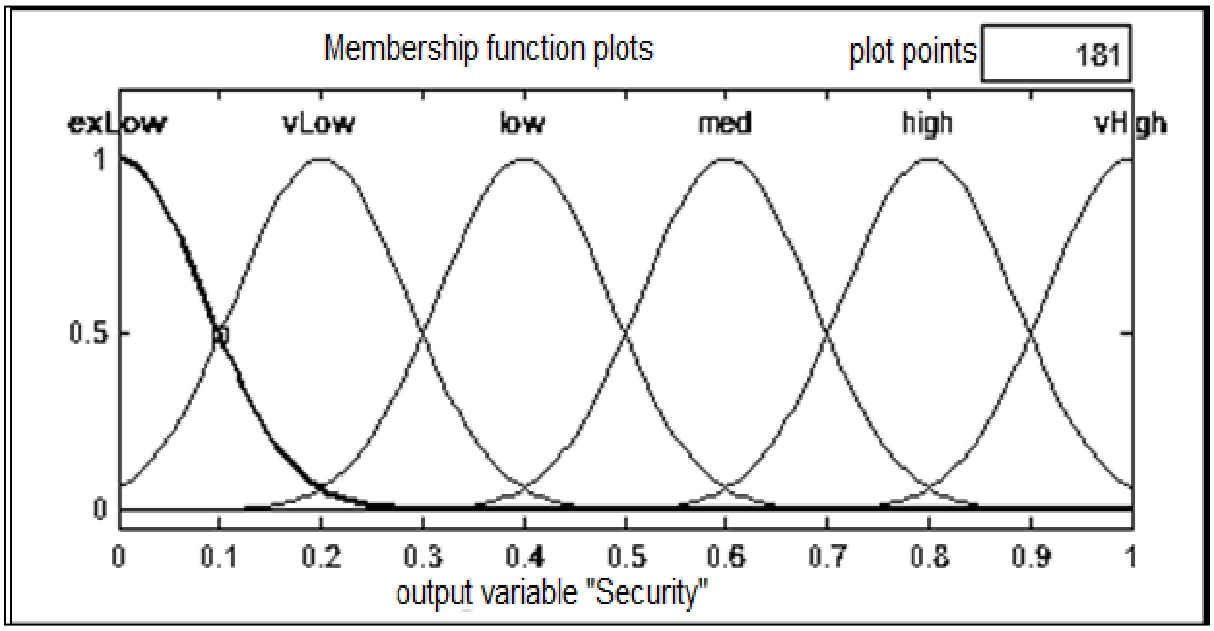


Figure 5.19: Membership Function graph for the output variable Security

5.5 Implementation and Testing

The proposed multi-layer security trust model MLSTM which derives a trust value from the fuzzy logic combinations of eight different security mechanisms (see section 5.1) from the three distinct cloud layers of IaaS, PaaS and SaaS was implemented using the matlab simulation software as shown in Figure 5.20

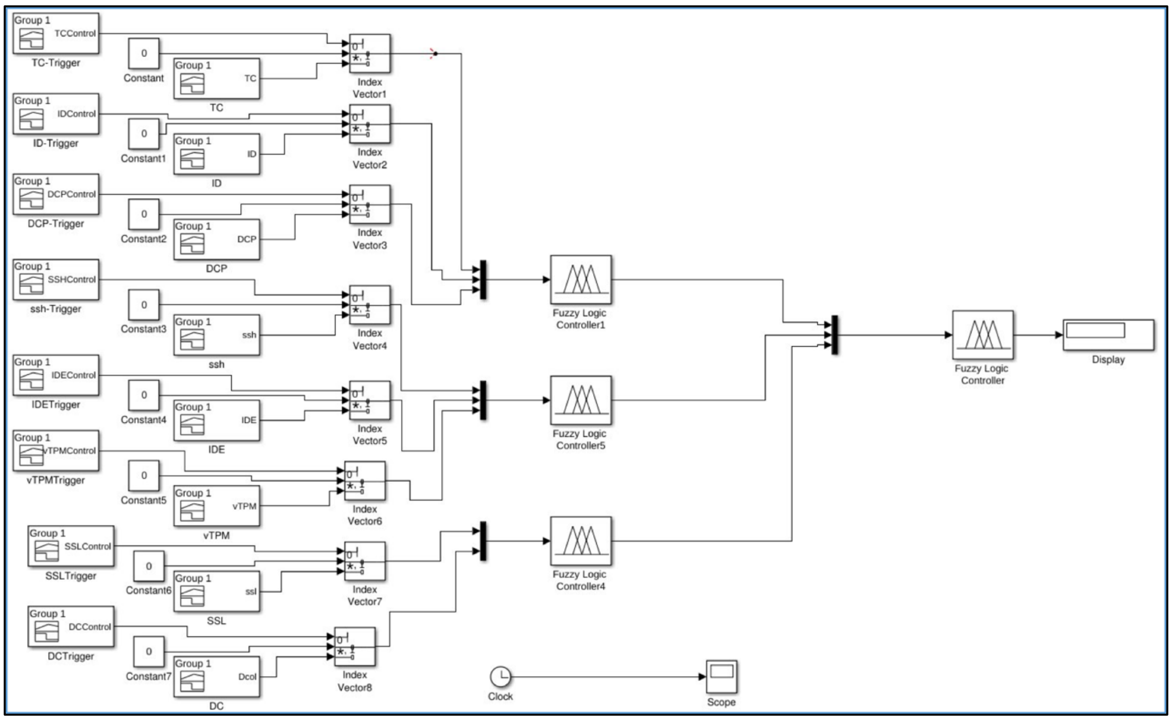


Figure 5.20: Matlab implementation of Multi-Layer Security Trust Modell (MLSTM)

Various simulations were conducted in both static and dynamic contexts. In the static context, the inputs (security mechanisms) could only take on two crisp or distinct states of 0 or 1 representing not-secure or fully-secured while in the dynamic context, the input values were changing with time.

For better understanding, the MLSTM identifies 4 distinct classification categories namely: High security, Normal security, Some-how secure and Low security that represent distinct combinations of the security mechanisms as shown in Table 5.7.

The trust value for a cloud platform is obtained through a controlled sequencing of the MLSTM transactions or probes as show in Table 5.8, where the "+" sign represents a fuzzy logic combination.

The MLSTM transactions results are obtained from specific probes (checks) that verify adequate response functionality expected from the security mechanism. A test-bed deployment of a cloud platform was also performed using the Eucalyptus cloud-in-a-box software and all security mechanisms listed in section 5.1 were implemented on the test-bed as well as MLSTM transactions or probes.

The results of the matlab simulation as well as other results from the test-bed are reported in the next section.

Table 5.7: MLSTM Categories

Category	**Combination of security mechanisms**							
	IaaS Layer			**PaaS Layer**			**SaaS Layer**	
	TC	**ID**	**DCP**	**SSH**	**IDE**	**vTPM**	**SSL**	**DC**
High Security	1	1	1	1	1	1	1	1
Normal Security	1	0	1	1	0	1	1	0
Some-how secure	0	0	1	1	1	0	1	0
Low Security	0	0	1	1	0	0	1	0

Table 5.8 : MLSTM Combinations

Transaction/sequence	**Combination of transactions**			
	High Security	**Normal Security**	**Some-how secure**	**Low security**
1	TC	DCP	DCP	DCP
2	TC+ID	DCP+SSH	DCP+SSH	DCP+SSH
3	TC+ID+DCP	DCP+SSH+SSL	DCP+SSL	DCP+SSH+SSL
4	TC+ID+DCP+ Ssh	DCP+TC	DCP +SSH+SSL	DCP+SSH+SSL
5	TC+ID+DCP+ SSH+IDE	DCP+SSH+TC	DCP+IDE	DCP+SSH+SSL
6	TC+ID+DCP+ SSH+IDE+vTPM	DCP+SSH+vTPM+ SSL	DCP+IDE+SSL	DCP+SSH+SSL

7	TC+ID+DCP+ SSH+IDE+vTPM+ SSL	DCP+TC+SSH+ vTPM	DCP+IDE+SSH	DCP+SSH+SSL
8	TC+ID+DCP+ SSH+IDE+vTPM+ SSL+DC	DCP+TC+SSH+ vTPM+SSL	DCP+IDE+SSH+ SSL	DCP+SSH+SSL

5.6 Results and Evaluation

In the matlab simulation, a static configuration where the transactions and probes are replaced by a constant value generator with an output value of 1 has the resulting trust values reported in Table 5.9 and Figure 5.21.

Table 5.9: MLSTM Trust Values by Sequence and Categories

Transaction/ sequence	**Trust value**			
	High security	**Normal security**	**Some-how secure**	**Low security**
1	0.235	0.235	0.235	0.235
2	0.4262	0.235	0.235	0.235
3	0.5708	0.2725	0.2725	0.2725
4	0.5708	0.4262	0.2745	0.2475
5	0.5974	0.4262	0.2974	0.2725
6	0.6234	0.4298	0.309	0.2725
7	0.7323	0.5928	0.4298	0.2725
8	0.7548	0.5935	0.4298	0.2725

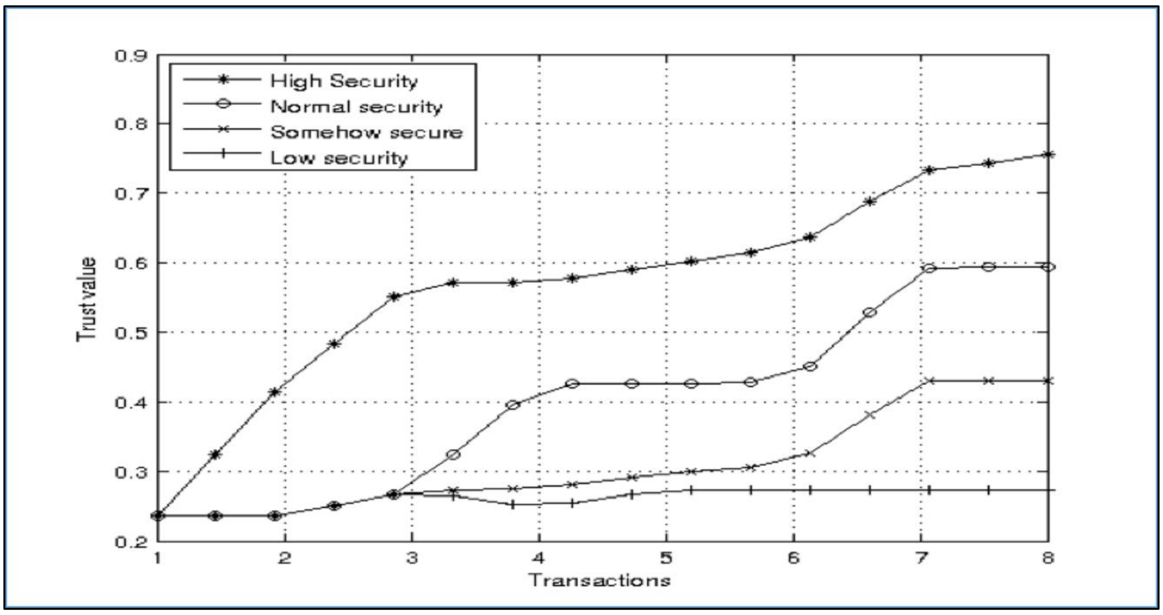


Figure 5.21: Multi-Layer Security Trust for 4 identified categories

Similarly, it is possible to obtain trust values for various combinations of security mechanisms as shown in Table 5.10.

The results presented in Tables 5.5 and 5.6 were derived from static considerations, where all security mechanisms are assumed to be fully secured. However, in a real-world context, a security mechanism may not always be in a fully secured state. For example, an un-patched SSH server is no longer in a fully secured state. Similarly, an IDE with an outdated database is no longer fully secured. In the matlab simulations, a real-world context was simulated by using suitable waveform generators for the transactions/probes.

Table 5.10: MLSTM Trust Values for Various Security Mechanisms Combinations

Scenarios	**Combinations**	**Trust Value**
1	TC	0.235
2	$TC \cap ID$	0.4262
3	$TC \cap DCP$	0.4262
4	$TC \cap ID \cap DCP$	0.5708
5	$TC \cap ID \cap DCP \cap ssh$	0.5708
6	$TC \cap ID \cap DCP \cap ssh \cap IDE$	0.5974

7	$TC \cap ID \cap DCP \cap ssh \cap IDE \cap vTPM$	0.6324
8	$TC \cap ID \cap DCP \cap ssh \cap IDE \cap vTPM \cap ssl$	0.7323
9	$TC \cap ID \cap DCP \cap ssh \cap IDE \cap vTPM \cap ssl \cap DC$	0.7547

The data in Table 5.11 is plotted in Figure 5.22 which shows the effect of the dynamic variations in the transaction results on the final trust value. As show, the final trust values appear to be lower by about 7%, however, the classification process is still successful. The peak trust graph shows the maximum trust values for a 10 seconds interval. It is really the same graph as Figure 5.21.

Table 5.11: MLSTM Trust Values for Real World Simulation

Transaction/ sequence	**Trust value**			
	High security	**Normal security**	**Some-how secure**	**Low security**
1	0.235	0.235	0.235	0.235
2	0.4259	0.3498	0.265	0.265
3	0.4712	0.2941	0.2941	0.2941
4	0.4720	0.432	0.2973	0.2974
5	0.5910	0.4355	0.4397	0.2974
6	0.6352	0.4524	0.3401	0.2956
7	0.6095	0.5906	0.4497	0.2941
8	0.595	0.593	0.369	0.2725
9	0.6772	0.5933	0.4513	0.2941
10	0.70804	0.5933	0.4298	0.2724

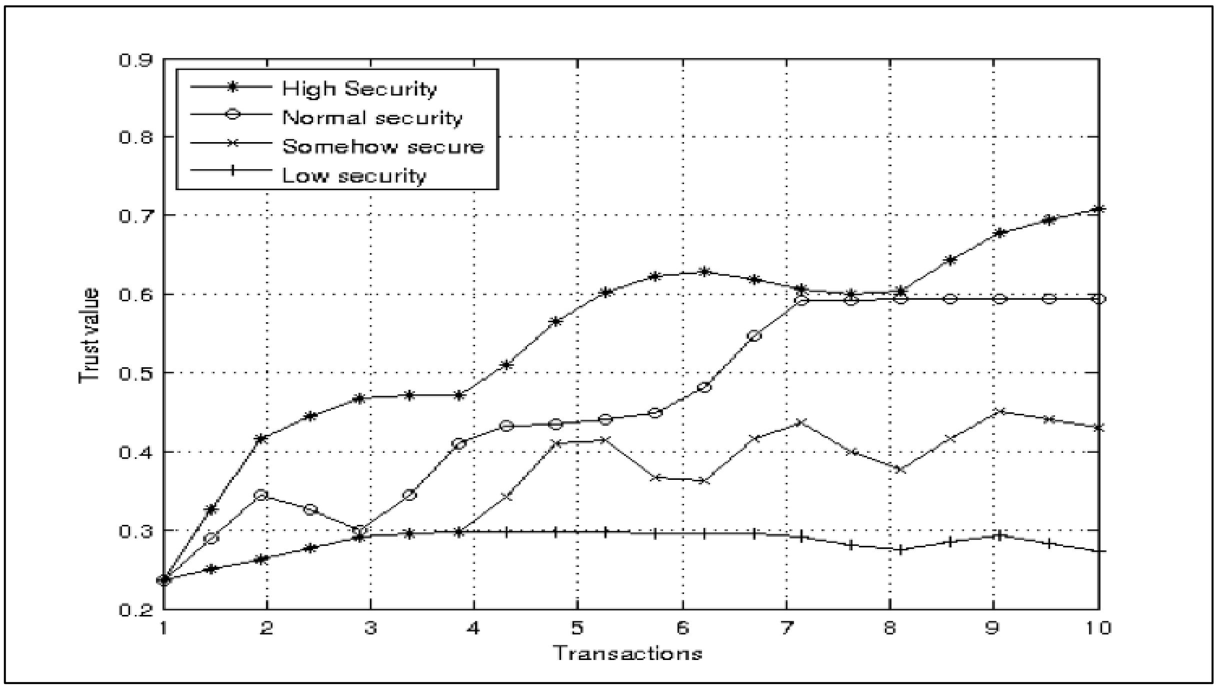


Figure 5.22: Multi-Layer Security for dynamic real world simulation

The MLSTM high success rate may be attributed to its use of special sequencing of 8 specific transactions devoted to security probes. In measuring the trust of a high security cloud, all 8 specific transactions are used and the results have been plotted in Figure 5.25 and also in comparison with other similar models – DMTC and TMFM [54] it reflects a higher success interaction rate. Figure 5.23 reflects the peak trust rate across the different security scenarios, Figure 5.24 reflects the dynamic input for the transactions in Figure 5.22 but in this instance it reflects the simulation carried out across 30 seconds. Figure 5.26 reflects the trust accuracy of MLSTM in comparison to other models. MLSTM already assumes a platform is untrusted and only begins to trust the platform when the probes and configure mechanisms are successful but the other models trust the platform and the trust level falls when the test carried out by users fail because the necessary security mechanisms are not in place.

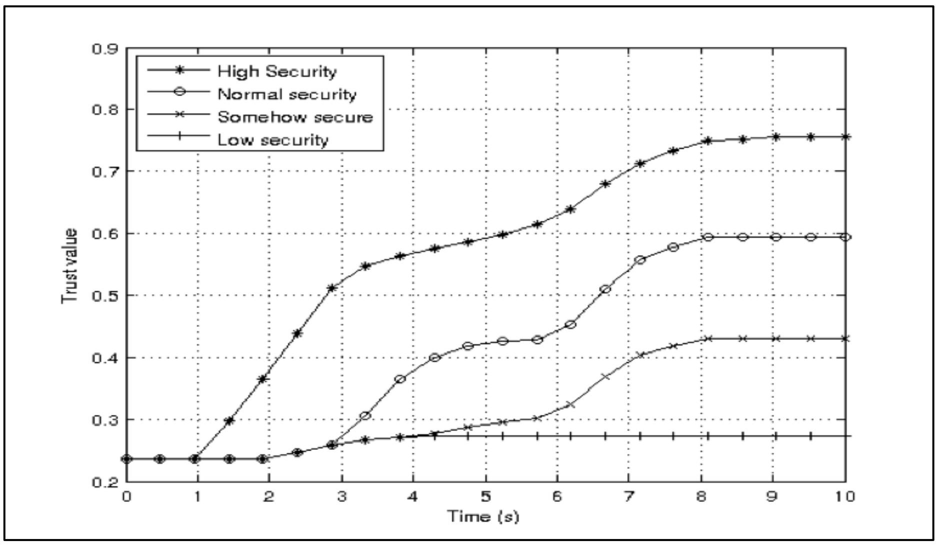


Figure 5.23: MLSTM Peak Trust Values

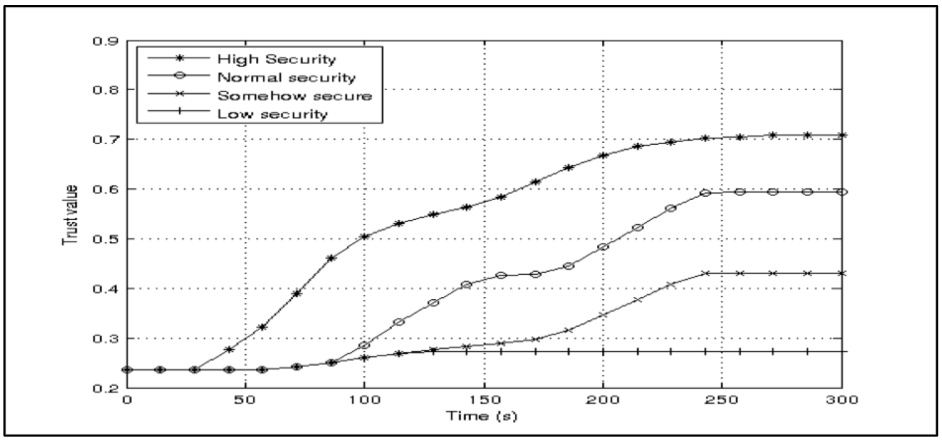


Figure 5.24: MLSTM matlab simulation with dynamic input transactions period of 30s

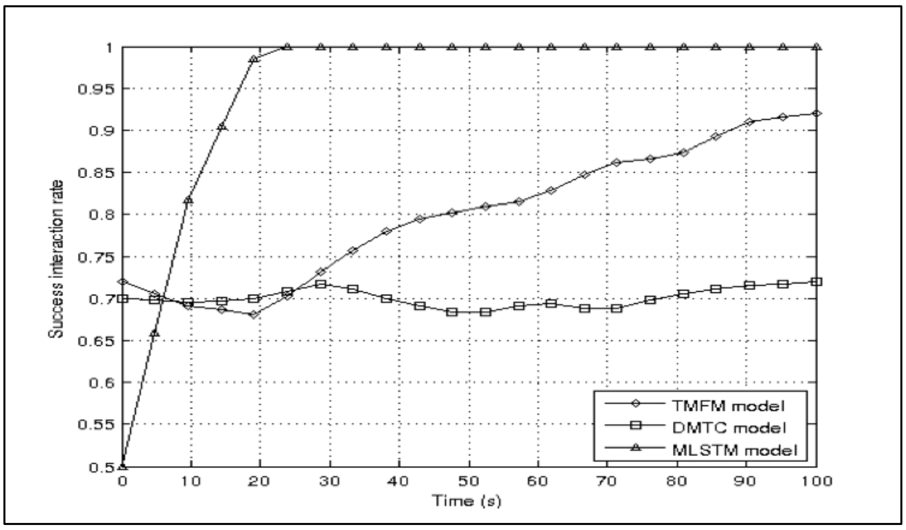


Figure 5.25: MLSTM success interaction rate in comparison with MDMTC and TMFM models

In detecting malicious hosts, the MLSTM model uses only 4 specific transactions and as Figure 5.26 shows, the MLSTM model can efficiently detect malicious hosts with a highly accurate rate. Here the MLSTM is benefitting from its initial assumption that all clouds may be malicious.

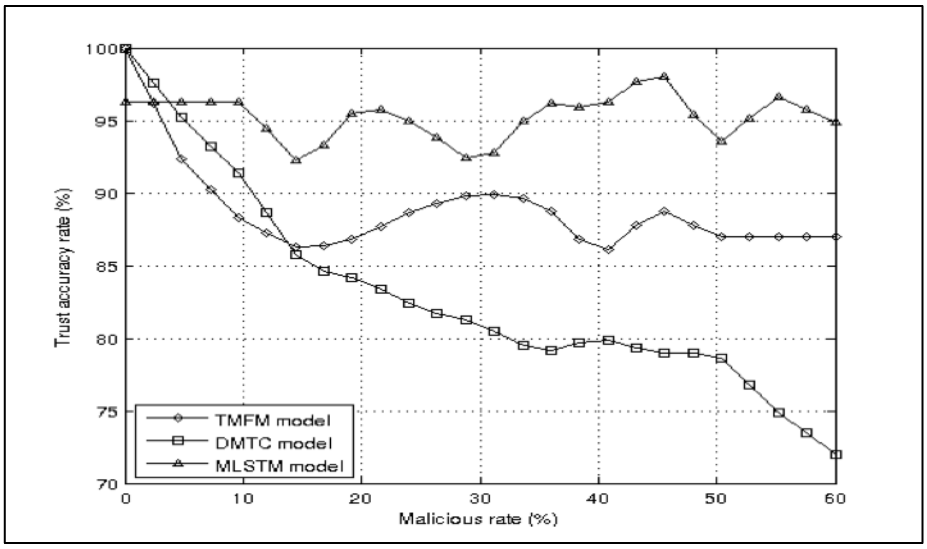


Figure 5.26: MLSTM trust accuracy rate comparison with DMTC and TMFM models

The data used for plotting Figure 5.27 reflects a large overhead which appears to be related to obtaining the TC (and vTPM) measurements. Probably this is due to the need for cryptographic encryption/decryption from the TPM chip/module. Figure 5.27 shows that it takes less than 1 second to identify a low security or some-how secure cloud, while it could take up to 8 seconds to identify a Normal security cloud and about 10 seconds to identify a high security cloud. This figure shows that it takes less than 1 second to identify a low security or some-how secure cloud, while it could take up to 8 seconds to identify a Normal security cloud and about 10 seconds to identify a high security cloud.

The results obtained from this research is compared to that obtained by [54] we see (Figure 5.26) that all have similar trust rate but as interactions continue they reflect different rates so my comparison would be along the following:

- Inputs : The results obtained in this research combines inputs from all layers of the platform to provide the trust value of the platform while the results in [54] have not considered the other layers and considered third party trust which could be biased

- Output : The model in this research meets it objectives better than the other two models as it efficiently detects malicious interactions and its trust accuracy remains consistent unlike the other models where the trust accuracy declines as reflected in Figure 5.26
- Impact : MLSTM impact is considered better as its successful interactions as seen in Figure 5.25 continues to increase as the success rate increases with successful interaction and would decline instead in the event of malicious interactions.

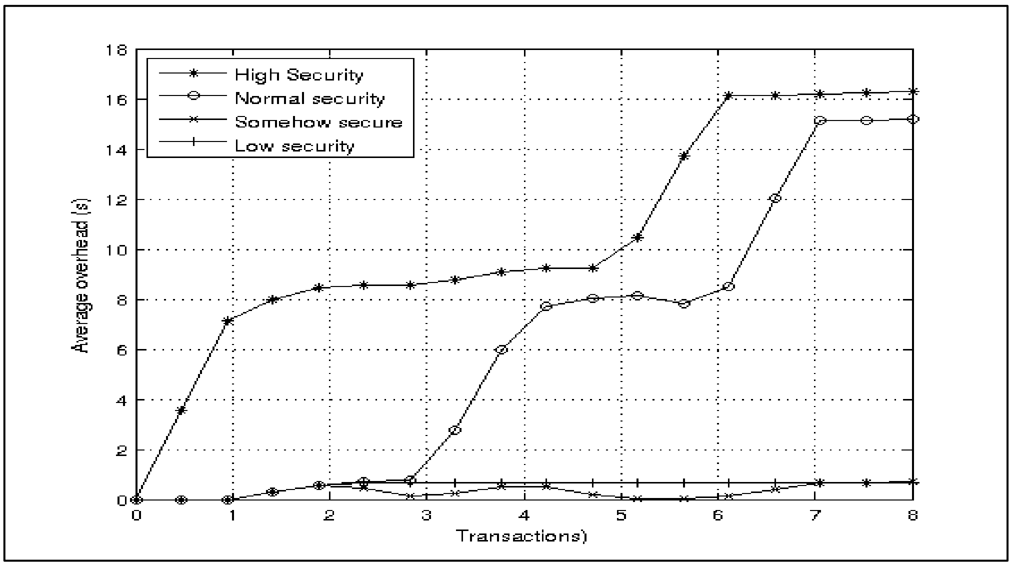


Figure 5.27: MLSTM average overhead for transactions

Figure 5.28 shows that at 50% of a transaction cycle, the MLSTM can provide a reasonable accurate classification of a cloud platform into high/normal security versus somehow secure/Low security.

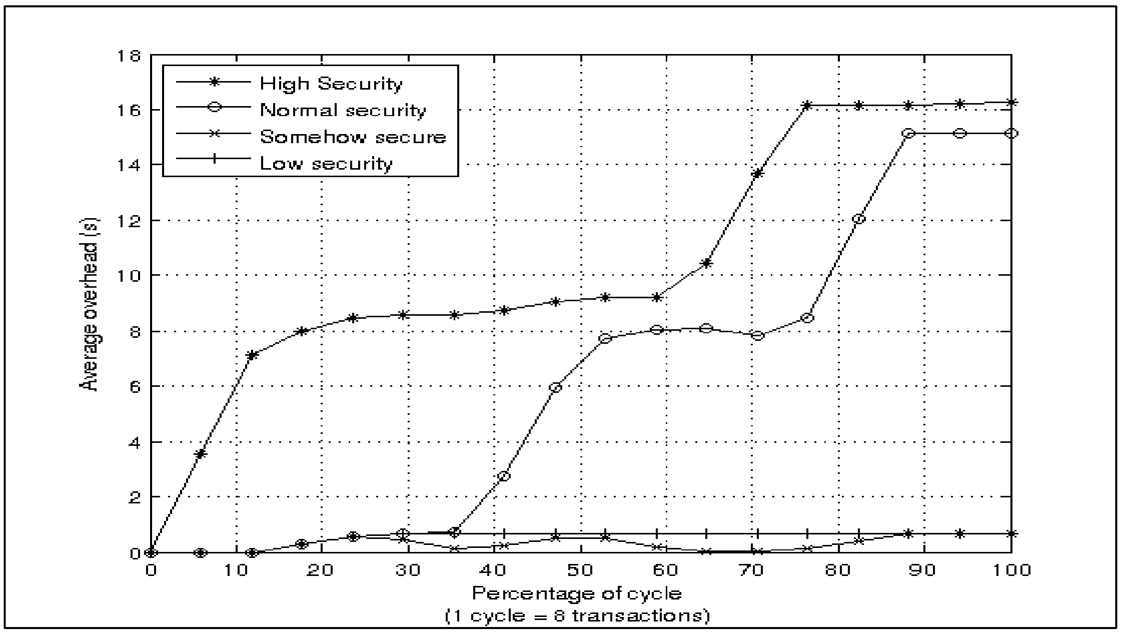


Figure 5.28: MLSTM average overhead for 1 transaction cycle

Figure 5.29 shows that the overhead required to complete the classification of a cloud is constant even after elevated number of transactions. This implies that the MLSTM model may be used for a continuous discrete sampling/classification of clouds.

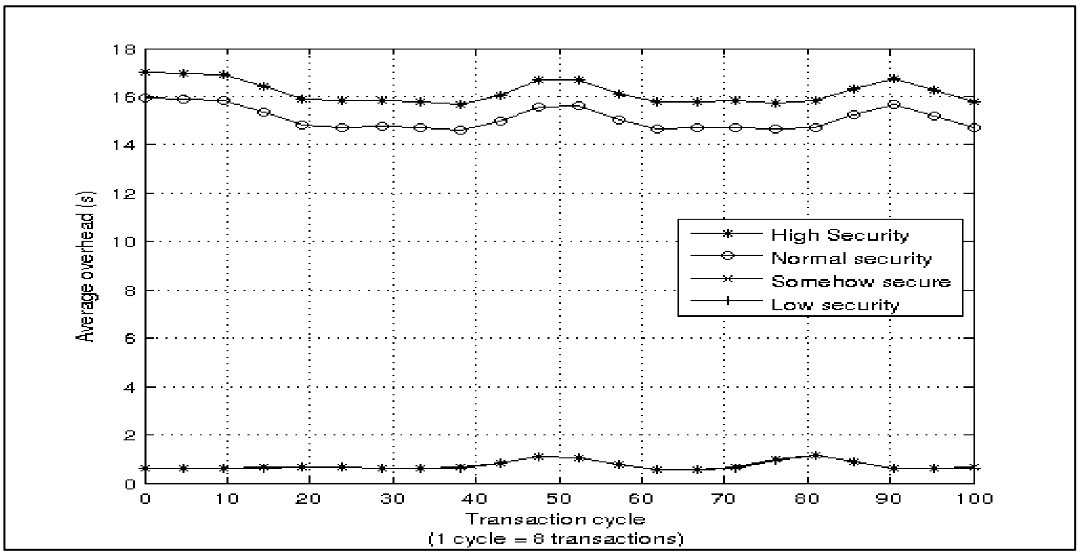


Figure 5.29: MLSTM average overhead for 100 transaction cycles

Figure 5.30 shows the overhead required to complete the parallel classification of a set of clouds. This implies that when the MLSTM model is used for the continuous discrete sampling/classification of clouds in parallel, there is a proportional increment in time to complete a classification.

The data used in plotting Figures 5.27, 5.28, 5.29 and 5.30 were measured from the Brunel Eucalyptus test-bed [103].

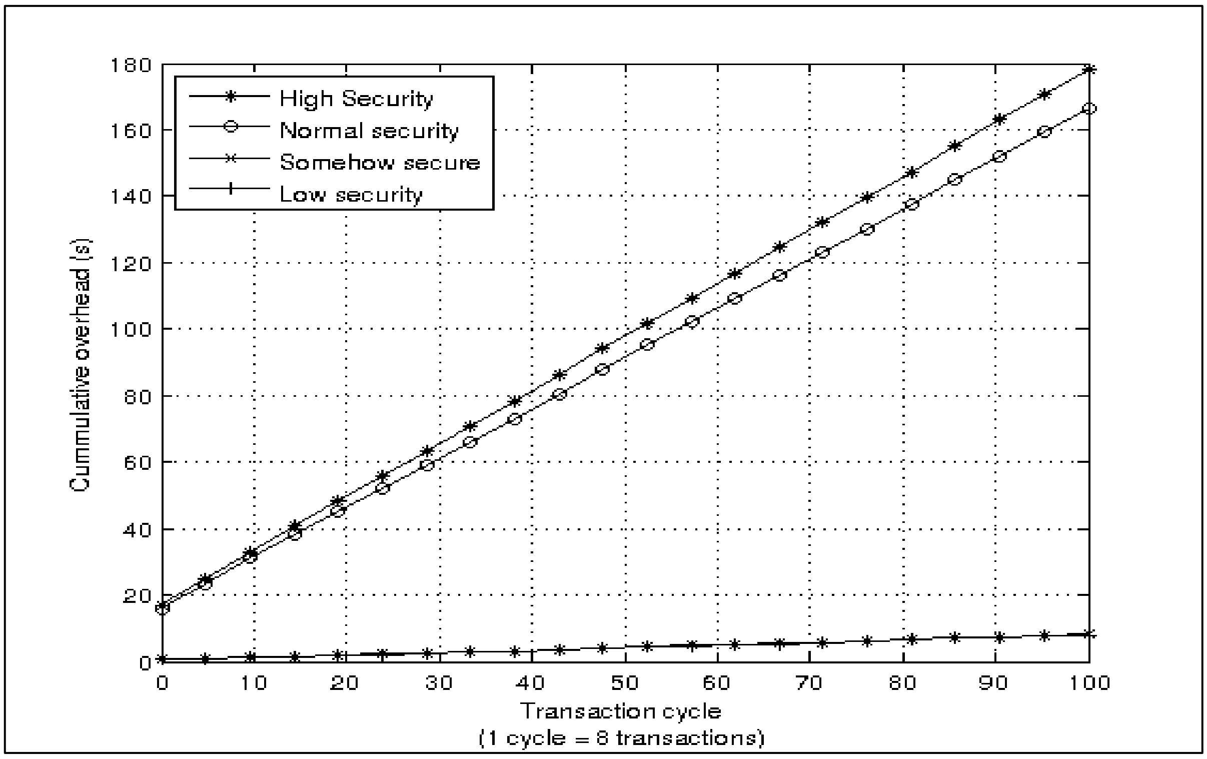


Figure 5.30: MLSTM cumulative overhead for 100 transaction cycle

5.7 Summary

The results presented above from the MLSTM suggest that the initial trust value of 0.235 shows marginal trust by end-users. Unlike other models, which show a convergence towards a maximum trust value of 1, the MLS trust model presented here has a maximum value of 0.7548 when all probes are in the OK state suggests that “cloud always include some inherent risk”. The MLSTM requires continuous discrete probing (sampling) to ensure that trust is maintained. The MLSTM fast convergence to a result in all 4 scenarios suggests a low overhead for parallelization and/or integration with normal cloud transactions.

The mechanisms used in this research to model trust in cloud are unique. This chapter presents the techniques and how it would be used to model trust in the cloud. Further work would include implementing and evaluating the techniques. The evaluation would include using fuzzy logic algorithms to select trustworthy cloud platform.

Chapter 6

Conclusions and Future Work

This chapter concludes the contribution of this research. It also outlines potential opportunities for further or improved work presented in this dissertation.

6.1 Conclusions

The research reported in this thesis focused on trusted cloud computing with distributed end-user attestable multi-layer security across the cloud platform.

Cloud computing is rapidly gaining acceptance in mission critical applications such as power, health, finance and education to mention but a few. Cloud computing allows remote processing using multiple computers and varied instances running at the same time. Cloud computing also allows shared resources to a variety of users on a single or shared resource; it allows the user to pay for only the consumed resources while allowing for scalability within the resources. Even with its potential in providing resources to multiple users at a reduced cost, it is becoming clear that end-users are still drawing back from fully adopting cloud computing and its deployed resources.

Cloud platforms remain easy targets for intruders due to its multi-tenancy and distributed nature. A cloud user wants to be confident and trust that the resources they are accessing are secure and available and its integrity is not compromised. An end-user wants to also ascertain ownership of the data or other resources they are accessing or processing. The end-user wants the data to be free of any interference while accessing or processing the data, basically an end-user wants a transparent system.

This thesis presented a unique end-user attestable multilayer security model that adequately addresses these challenges given that no one solution fits or cuts across all the layers of the cloud platform. The MLS model comprises of individual and distinct security mechanisms across the 4 major layers of a cloud platform – physical, IaaS, PaaS and SaaS.

The research work on the MLS security model began with the deployment of a community cloud platform on the Brunel network where a number of security configurations were implemented. Trusted Computing based on hardware Trusted Platform Module device was then implemented to secure the IaaS layer. At the PaaS layer, a virtual (software) TPM device was deployed to provide TC-like service that was subsequently combined with an intrusion detection engine (IDE) system where file level integrity verifications are carried out based on stored hashes. A high-security image for secure-instances was created and used by power-users to verify (on-demand) the integrity of the physical and IaaS layer using TC attestation and the PaaS layer using the IDE. This image was evaluated and used for deployment of a mission critical SaaS application for the energy sector. The results obtained show the average overhead in starting secure instances is less than 5% (< 3 sec) which is next to minimal and the gains in running a secure instance with such overhead is considered to be beneficial when compared with a less secure instance.

Subsequently, at the SaaS layer, a unique implementation of data-colouring was developed that uses a combination of digital fingerprints and steganography for securing data integrity and highlighting a possible path-of-loss in the event of a theft or comprise. In the implementation deployed on the community cloud platform, the process of data colouring is carried out off-line to improve on the integrity of the data as only then is the user or data owner sure that the integrity of the data is substantiated before enforcing any protection. This also improves the trust relationship between a user, provider and co-tenants on the same platform.

To provide a holistic trust status of the whole system, a fuzzy logic computing technique was used to develop a model known as Multilayer Security Trust Model (MLSTM) for combining the states of several security mechanisms across all cloud layers to provide a unified trust status for the platform or depending on the needs of the user the trust status of the desired layer. This MLSTM model is useful for classifying and comparing various cloud platforms which would then provide an informed decision on "which to choose" based on its trustworthy state. That is, it provides a tool to measure transparency, integrity and accountability of the resources even for major providers like Google, Microsoft Azure ore Amazon EC2. MLSTM was developed as a matlab simulation and also evaluated on the deployed cloud platform at Brunel. The results obtained show that the model has improved characteristics over the TMFM and DMTC models and MLSTM models cloud platforms with better accuracy.

The MLS trust model implemented in this research, models cloud trustworthiness across its layers or the whole cloud platform and it provides a much needed tool for end-users to attest or verify or classify a cloud for trustworthiness on a continuous and on-demand basis.

The research work provides an autonomous method for continuous testing and probing across multiple (all) cloud layers using the different security mechanisms and thus enables the end-user to make informed decision on-demand.

The outputs of this research work are applicable across any cloud platform as they do not require changes in the underlying software code layer.

While, the implementation of TC based on TPM is scalable as each virtual server running on any platform has its own keys and each one is uniquely identified as far TPM is concerned, end-user attestation/verification is based on the use of pre-seeded databases that may limit overall scalability. The management/distribution of these pre-seeded databases is a possible future work

As more applications are moving online, processing data on the cloud platform is gradually becoming important and thus the need for an alternative to encrypting data on the cloud is important. Using data colouring as implemented in this research work allows transparent processing of protected data on cloud platforms preserving its integrity. However, the identification of data loss or theft path from fingerprinting in the implementation of data colouring is suggestive and limited by other factors outside the scope of this work.

The MLS trust model also allows users to assess the platform they want to use based on trust values derived from multiple security mechanisms. While, using multiple mechanisms provides different level of protection across different layers of the platform possibly mitigating effects of single-point of failure of security mechanisms on the platform, the implementation is based on UNIX (Linux) and would require some adjustments for non-Unix cloud platforms. Continuous testing/probing using the MLS trust model is limited by the combined duration/overhead from individual probes/tests.

6.3 Future Work

Future work on the MLS trust model would involve reducing overall overhead, possibly by optimising and caching heavy transactions. The results show that in particular TC and vTPM transactions account for over 90% of the overhead. A possible strategy would be to perform a TC and vTPM transaction once in 5 minutes and cache the results for use during probe cycles. It is estimated that a cycle performed using the cached values for TC and vTPM would complete in less than a second.

An alternative is to study the use of parallel transactions especially in a context of testing multiple cloud servers/services concurrently. Here it would be necessary to study both parallelisation on a single host and parallelization on a cluster of hosts with some inter-host communication.

A future work would also be to study the automated scheduling and periodic-execution of MLS trust model transactions as a tool for automated cloud security testing, verification and reporting. In this case, the MLS trust model would be studied as a possible tool for level 3 certification in the CSA STAR [8] Open Certification and similar framework.

Even as NIST, ITU-T, IEEE and other cloud focus groups work on standardisation and interoperability, it is of paramount importance too that extensive work be carried out on in the area of interoperability so another area of future work would involve studying the interoperability and portability of the outputs of this research work on diverse (non-Unix) cloud implementations, especially in the context of seamless migrations of end-user service/data across cloud platforms/providers. It shouldn't pose a problem to a user that a change in provider would mean also rebuilding one's application to suite the new platform depending on the provider's settings, adjusting the security settings of the user's application or resources to match the capabilities of the provider or even having to be concerned about how the data would be handled while in transit between the providers.

In line with interoperability different jurisdiction may have different data policy so it may also be necessary to look into having a global and neutral body handle issues surrounding interoperability implementation and guidelines.

The use of the outputs from this research work in the context of ISO27018 certification [123] and the planned European based infrastructure is required. The ISO27018 is the code of practise for protection of personally identified information (PII) in public clouds. Where processing requires that sensitive data remain encrypted. Specifically, studying the merits of data colouring based on fingerprinting even for existing services such as Dropbox is immeasurable. A possible study would involve uploading coloured versions of a data-file via different cloud services (for example dropbox and gmail) and the random tampering/hidden-exchange of these files between participants would be detected using techniques documented in Chapter 4.

The model presented in this thesis provides a needed tool to determine the trustworthy status of a cloud platform and its impact on the global deployment of cloud services requires additional study as users become more confident of data integrity, availability of service and non-repudiation of theft or data tampering.

As the identification of data loss or theft path from fingerprinting in the implementation of data colouring is suggestive and not conclusive, further research can be carried to ascertain in total if it was as result of a break in on the platform or it was actually tampered with by an authorised user. Mechanisms may be put in place to further secure the data in the event the platform is attacked.

References

[1] M. R. Nelson, “OECD Briefing Paper for the ICCP Technology Foresight Forum: Cloud Computing and Public Policy | Publications | Georgetown University,” 2009. [Online]. Available: http://explore.georgetown.edu/publications/47143/. [Accessed: 28-Oct-2015].

[2] M. Armbrust, I. Stoica, M. Zaharia, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, and A. Rabkin, “A view of cloud computing,” *Commun. ACM*, vol. 53, no. 4, p. 50, Apr. 2010.

[3] R. Chalse, A. Selokar, and A. Katara, “A New Technique of Data Integrity for Analysis of the Cloud Computing Security,” in *2013 5th International Conference on Computational Intelligence and Communication Networks*, 2013, pp. 469–473.

[4] P. Mell, “What’s Special about Cloud Security?,” *IT Prof.*, vol. 14, no. 4, pp. 6–8, 2012.

[5] P. Mell and T. Grance, “The NIST Definition of Cloud Computing (Draft) Recommendations of the National Institute of Standards and Technology,” *Nist Spec. Publ.*, vol. 145, no. 6, p. 7, 2011.

[6] NIST, “NIST Cloud Computing Security Reference Architecture - M0007_v1_3376532289.pdf.” [Online]. Available: http://bigdatawg.nist.gov/_uploadfiles/M0007_v1_3376532289.pdf. [Accessed: 03-Mar-2016].

[7] NIST, “An Introduction to Computer Security: The NIST Handbook - handbook.pdf.” [Online]. Available: http://www.davidsalomon.name/CompSec/auxiliary/handbook.pdf. [Accessed: 04-Mar-2016].

[8] CSA, “CSA Security, Trust & Assurance Registry (STAR) : Cloud Security Alliance.” [Online]. Available: https://cloudsecurityalliance.org/star/. [Accessed: 03-Mar-2016].

[9] CSA, “Consensus Assessments : Cloud Security Alliance.” [Online]. Available: https://cloudsecurityalliance.org/group/consensus-assessments/. [Accessed: 03-Mar-2016].

[10] W. K. Daniel, “Challenges on privacy and reliability in cloud computing security,” in *2014 International Conference on Information Science, Electronics and Electrical Engineering*, 2014, vol. 2, pp. 1181–1187.

[11] A. Behl and K. Behl, “An analysis of cloud computing security issues,” in *2012 World Congress on Information and Communication Technologies*, 2012, pp. 109–114.

[12] Z. Tari, X. Yi, U. S. Premarathne, P. Bertok, and I. Khalil, “Security and Privacy in Cloud Computing: Vision, Trends, and Challenges,” *IEEE Cloud Comput.*, vol. 2, no. 2, pp. 30–38, Mar. 2015.

[13] A. Gordon, “The Hybrid Cloud Security Professional,” *IEEE Cloud Comput.*, vol. 3, no. 1, pp. 82–86, Jan. 2016.

[14] A. Hendre and K. P. Joshi, “A Semantic Approach to Cloud Security and Compliance,” in *2015 IEEE 8th International Conference on Cloud Computing*, 2015, pp. 1081–1084.

[15] D. Gollmann, *Computer Security*, 2 edition. John Wiley & Sons, Inc., 2006.

[16] K. D. Mitnick and W. L. Simon, “The Art of Deception: Controlling the Human Element in Security,” *BMJ: British Medical Journal*, 2003. [Online]. Available: http://www.bmj.com/content/347/bmj.f5889. [Accessed: 04-Mar-2016].

[17] J. G. Bronson, “Unfriendly eyes,” *IEEE Trans. Prof. Commun.*, vol. PC-30, no. 3, pp. 173–178, 1987.

[18] L. M. Kaufman, “Data security in the world of cloud computing,” *IEEE Secur. Priv.*, vol. 7, no. 4, pp. 61–64, 2009.

[19] BBC, “Edward Snowden: Leaks that exposed US spy programme - BBC News.” [Online]. Available: http://www.bbc.com/news/world-us-canada-23123964. [Accessed: 04-Mar-2016].

[20] U. C. Office, “Security_Requirements_for_List_X_Contractors.pdf.” [Online]. Available: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/367514/Security_Requirements_for_List_X_Contractors.pdf. [Accessed: 05-Mar-2016].

[21] P. Giannoulis and S. Northcutt, “Physical Security.” [Online]. Available: http://www.sans.edu/research/security-laboratory/article/281. [Accessed: 05-Mar-2016].

[22] TPM, “Trusted Computing Group - Solutions - Cloud Security.” [Online]. Available: http://www.trustedcomputinggroup.org/solutions/cloud_security. [Accessed: 20-Jul-2015].

[23] TPM, “Trusted Computing Group - Trusted Platform Module (TPM) Summary.” [Online]. Available: http://www.trustedcomputinggroup.org/resources/trusted_platform_module_tpm_summary. [Accessed: 04-Mar-2016].

[24] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, and K.-Y. Tung, “Intrusion detection system: A comprehensive review,” *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 16–24, Jan. 2013.

[25] R. Bace and P. Mell, “NIST Special Publication on Intrusion Detection Systems.” [Online]. Available: http://www.21cfrpart11.com/files/library/reg_guid_docs/nist_intrusiondetectionsys.pdf. [Accessed: 05-Mar-2016].

[26] M. R. Nelson, “The Cloud, the Crowd, and Public Policy,” *Issues in Science and Technology*, 2009. [Online]. Available: http://issues.org/25-4/nelson-2/. [Accessed: 20-Jul-2015].

[27] W. Jansen and T. Grance, “Guidelines on Security and Privacy in Public Cloud Computing,” *Director*, vol. 144, no. 7, pp. 800–144, 2011.

[28] S. Kaur and P. S. Mann, “A review on cloud computing issues and challenges,” *Intl J. Res. Comput. Appl. Robot.*, vol. 2, no. 5, pp. 63–68, 2014.

[29] P. A. Laplante, J. Zhang, and J. Voas, “What’s in a Name? Distinguishing between SaaS and SOA,” *IT Prof.*, vol. 10, no. 3, pp. 46–50, May 2008.

[30] D. Meyer and G. Zobrist, “TCP/IP versus OSI,” *IEEE Potentials*, vol. 9, no. 1, pp. 16–19, Feb. 1990.

[31] C. Saravanakumar and C. Arun, “Survey on interoperability, security, trust, privacy standardization of cloud computing,” in *2014 International Conference on Contemporary Computing and Informatics (IC3I)*, 2014, pp. 977–982.

[32] S. De Chaves, R. Uriarte, and C. Westphall, “Toward an architecture for monitoring private clouds,” *IEEE Commun. Mag.*, vol. 49, no. 12, pp. 130–137, Dec. 2011.

[33] Z. Chen and J. Yoon, “IT Auditing to Assure a Secure Cloud Computing,” in *2010 6th World Congress on Services*, 2010, pp. 253–259.

[34] M. Guizani, “Protecting private cloud located within public cloud,” in *2013 IEEE Global Communications Conference (GLOBECOM)*, 2013, pp. 677–681.

[35] A. M. Khan, L. Navarro, L. Sharifi, and L. Veiga, “Clouds of small things: Provisioning infrastructure-as-a-service from within community networks,” in *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2013, pp. 16–21.

[36] M. Gall, A. Schneider, and N. Fallenbeck, “An Architecture for Community Clouds Using Concepts of the Intercloud,” in *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, 2013, pp. 74–81.

[37] V. Varadharajan and U. Tupakula, “Security as a Service Model for Cloud Environment,” *IEEE Trans. Netw. Serv. Manag.*, vol. 11, no. 1, pp. 60–75, Mar. 2014.

[38] X. Chen, C. Chen, Y. Tao, and J. Hu, “A Cloud Security Assessment System Based on Classifying and Grading,” *IEEE Cloud Comput.*, vol. 2, no. 2, pp. 58–67, Mar. 2015.

[39] K. Hashizume, D. G. Rosado, E. Fernández-Medina, and E. B. Fernandez, “An analysis of security issues for cloud computing,” *J. Internet Serv. Appl.*, vol. 4, no. 1, p. 5, 2013.

[40] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, “Cross-VM side channels and their use to extract private keys,” in *Proceedings of the 2012 ACM conference on Computer and communications security - CCS ’12*, 2012, p. 305.

[41] Q. Yan, F. R. Yu, Q. Gong, and J. Li, “Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments: A Survey, Some Research Issues, and Challenges,” *IEEE Commun. Surv. Tutorials*, vol. 18, no. 1, pp. 602–622, 2016.

[42] M. Ficco and M. Rak, “Stealthy Denial of Service Strategy in Cloud Computing,” *IEEE Trans. Cloud Comput.*, vol. 3, no. 1, pp. 80–94, Jan. 2015.

[43] S. Yu, Y. Tian, S. Guo, and D. O. Wu, “Can We Beat DDoS Attacks in Clouds?,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2245–2254, Sep. 2014.

[44] L. A. Zadeh, “Fuzzy sets,” *Inf. Control*, vol. 8, no. 3, pp. 338–353, Jun. 1965.

[45] J. Singla, “Comparative study of Mamdani-type and Sugeno-type fuzzy inference systems for diagnosis of diabetes,” in *2015 International Conference on Advances in Computer Engineering and Applications*, 2015, pp. 517–522.

[46] E. H. Mamdani and S. Assilian, “An experiment in linguistic synthesis with a fuzzy logic controller,” *Int. J. Man. Mach. Stud.*, vol. 7, no. 1, pp. 1–13, Jan. 1975.

[47] H. Sato, A. Kanai, and S. Tanimoto, “A Cloud Trust Model in a Security Aware Cloud,” in *2010 10th IEEE/IPSJ International Symposium on Applications and the Internet*, 2010, pp. 121–124.

[48] D. Gonzales, J. Kaplan, E. Saltzman, Z. Winkelman, and D. Woods, “Cloud-Trust - a Security Assessment Model for Infrastructure as a Service (IaaS) Clouds,” *IEEE Trans. Cloud Comput.*, pp. 1–1, 2015.

[49] S. Saadat and H. R. Shahriari, “Towards a process-oriented framework for improving trust and security in migration to cloud,” in *2014 11th International ISC Conference on Information Security and Cryptology*, 2014, pp. 220–225.

[50] P. Sirohi and A. Agarwal, “Cloud computing data storage security framework relating to data integrity, privacy and trust,” in *2015 1st International*

Conference on Next Generation Computing Technologies (NGCT), 2015, pp. 115–118.

[51] S. M. Habib, V. Varadharajan, and M. Muhlhauser, “A Trust-Aware Framework for Evaluating Security Controls of Service Providers in Cloud Marketplaces,” in *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, 2013, pp. 459–468.

[52] D. Wallom, M. Turilli, A. Martin, A. Raun, G. Taylor, N. Hargreaves, and A. McMoran, “myTrustedCloud: Trusted Cloud Infrastructure for Security-critical Computation and Data Managment,” in *2011 IEEE Third International Conference on Cloud Computing Technology and Science*, 2011, pp. 247–254.

[53] K. Hwang and D. Li, “Trusted Cloud Computing with Secure Resources and Data Coloring,” *IEEE Internet Comput.*, vol. 14, no. 5, pp. 14–22, Sep. 2010.

[54] A. Mohsenzadeh and H. Motameni, “A trust model between cloud entities using fuzzy mathematics,” *J. Intell. Fuzzy Syst.*, vol. 29, no. 5, pp. 1795–1803, Jul. 2015.

[55] S. Han’guk Chŏnja T'ongsin Yŏn'guwŏn., M. Han’guk Chŏngbo Sahoe Chinhŭngwŏn., Global IT Research Institute (Korea), and IEEE Communications Society., *The 12th International Conference on Advanced Communication Technology : ICT for green growth and sustainable development : ICACT 2010 : Phoenix Park, Korea, Feb. 7-10, 2010 : proceedings*, vol. 2. IEEE, 2010.

[56] A. Donevski, S. Ristov, and M. Gusev, “Security assessment of virtual machines in open source clouds,” *Information & Communication Technology Electronics & Microelectronics (MIPRO), 2013 36th International Convention on*. pp. 1094–1099, 2013.

[57] S. Biedermann, M. Zittel, and S. Katzenbeisser, “Improving security of virtual machines during live migrations,” in *2013 Eleventh Annual Conference on Privacy, Security and Trust*, 2013, pp. 352–357.

[58] A. A. M. Matsui, S. Michalsky, and M. A. Gerosa, “Using Virtual Machine Security to Reinforce Components Constraints,” in *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*, 2012, pp. 138–141.

[59] R. Mehrotra, A. Dubeyy, S. Abdelwahed, and K. Rowland, “On state of the art in virtual machine security,” in *2012 Proceedings of IEEE Southeastcon*, 2012, pp. 1–6.

[60] S. Berger, R. Cáceres, K. Goldman, D. Pendarakis, R. Perez, J. R. Rao, E. Rom, R. Sailer, W. Schildhauer, D. Srinivasan, S. Tal, and E. Valdez, “Security for the cloud infrastructure: trusted virtual data center implementation,” *IBM J. Res. Dev.*, vol. 53, no. 4, pp. 560–571, Jul. 2009.

[61] W. A. Jansen, "Cloud Hooks: Security and Privacy Issues in Cloud Computing," in *2011 44th Hawaii International Conference on System Sciences*, 2011, pp. 1–10.

[62] K. Vieira, A. Schulter, C. B. Westphall, and C. M. Westphall, "Intrusion Detection for Grid and Cloud Computing," *IT Prof.*, vol. 12, no. 4, pp. 38–43, Jul. 2010.

[63] AIDE, "Advance Intrusion Detection Environment." [Online]. Available: http://aide.sourceforge.net/. [Accessed: 03-Mar-2015].

[64] C. S. Collberg and C. Thomborson, "Watermarking, tamper-proofing, and obfuscation - tools for software protection," *IEEE Trans. Softw. Eng.*, vol. 28, no. 8, pp. 735–746, Aug. 2002.

[65] Y.-C. Liu, Y.-T. Ma, H.-S. Zhang, D.-Y. Li, and G.-S. Chen, "A method for trust management in cloud computing: Data coloring by cloud watermarking," *Int. J. Autom. Comput.*, vol. 8, no. 3, pp. 280–285, Aug. 2011.

[66] Eucalyptus, "Plan Your Deployment | HP Helion Eucalyptus." [Online]. Available: https://www.eucalyptus.com/eucalyptus-cloud/plan-deployment. [Accessed: 28-Aug-2015].

[67] Seiji Munetoh, "userguide-0.2.4.pdf." [Online]. Available: http://jaist.dl.osdn.jp/openpts/51879/userguide-0.2.4.pdf. [Accessed: 28-Aug-2015].

[68] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-Source Cloud-Computing System," in *2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2009, pp. 124–131.

[69] B. I. Santoso, "DESIGNING NETWORK SECURITY FOR EUCALYPTUS PRIVATE CLOUD INFRASTRUCTURE USING SSL-VPN," in *Seminar Nasional Teknologi Informasi dan Komunikasi 2014 (SENTIKA 2014)*, 2014.

[70] Eucalyptus, "Eucalyptus Documentation." [Online]. Available: http://docs.hpcloud.com/eucalyptus/4.2.1/#install-guide/euca_components.html. [Accessed: 19-Apr-2016].

[71] HP-Cloud, "Eucalyptus Documentation." [Online]. Available: http://docs.hpcloud.com/eucalyptus/4.2.1/#user-guide/index.html. [Accessed: 20-Apr-2016].

[72] Xen, "The Xen Project, the powerful open source industry standard for virtualization." [Online]. Available: http://www.xenproject.org/. [Accessed: 19-Apr-2016].

[73] K. Ren, C. Wang, and Q. Wang, "Security Challenges for the Public Cloud," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, Jan. 2012.

[74] X. Zhang, H. Du, J. Chen, Y. Lin, and L. Zeng, “Ensure Data Security in Cloud Storage,” in *2011 International Conference on Network Computing and Information Security*, 2011, vol. 1, pp. 284–287.

[75] G. C. Kessler, “Steganography for the Computer Forensics Examiner,” 2015. [Online]. Available: http://www.garykessler.net/library/fsc_stego.html. [Accessed: 20-Jul-2015].

[76] C. S. Collberg and C. Thomborson, “Watermarking, tamper-proofing, and obfuscation - Tools for software protection,” *IEEE Trans. Softw. Eng.*, vol. 28, no. 8, pp. 735–746, 2002.

[77] S. U. S. Sandosh, “AN AUTHENTICATION IN CLOUD THROUGH DATA COLORING USING PROGRESSIVE APPROACH,” *Int. J. Curr. Res. Rev.*, vol. 4, no. 21, pp. 179–182, 2012.

[78] T. Morkel, “Steganography and Steganalysis,” 2005.

[79] C. S. Collberg and C. Thompson, “Watemarking, tamper-proofing, and obfuscation - tools for software protection,” *IEEE Trans. Softw. Eng*, vol. 28, no. 8, pp. 735–746, 2002.

[80] OutGuess, “OutGuess - Information,” 2015. [Online]. Available: http://www.outguess.org/info.php. [Accessed: 20-Jul-2015].

[81] Ubuntu, “Howtomd5sum.” .

[82] R. Rivest, “The md5 message-digest algorithm.” .

[83] M. Armbrust, M. Armbrust, A. Fox, A. Fox, R. Griffith, R. Griffith, A. Joseph, A. Joseph, RH, and RH, “Above the clouds: A Berkeley view of cloud computing,” *Univ. California, Berkeley, Tech. Rep. UCB* , pp. 07–013, 2009.

[84] R. Shaikh and M. Sasikumar, “Cloud Simulation Tools: A Comparative Analysis,” *IJCA Proc. Int. Conf. Green Comput. Technol.*, vol. ICGCT, no. 3, pp. 11–14.

[85] M. H. Rahman and M. Adnan, “Survey on cloud simulator.” [Online]. Available: http://www.slideshare.net/habibur01/survey-on-cloud-simulator. [Accessed: 30-Oct-2015].

[86] G. Gębczyński, J. Kołodziej, and S. U. Khan, “Secure-Sim-G: Security-Aware Grid Simulator - Basic Concept and Structure,” *J. Telecommun. Inf. Technol.*, vol. nr 1, pp. 33–42, 2012.

[87] M. Brinklov and R. Sharp, “Incremental Trust in Grid Computing,” in *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid ’07)*, 2007, pp. 135–144.

[88] M. B. B. K. Thomas Beth, “Valuation of Trust in Open Networks.”

[89] P. Humane and J. N. Varshapriya, “Simulation of cloud infrastructure using CloudSim simulator: A practical approach for researchers,” in *2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, 2015, pp. 207–211.

[90] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” *Softw. Pract. Exp.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[91] W. Long, L. Yuqing, and X. Qingxin, “Using CloudSim to Model and Simulate Cloud Computing Environment,” in *2013 Ninth International Conference on Computational Intelligence and Security*, 2013, pp. 323–328.

[92] Xiang Li, Xiaohong Jiang, Kejiang Ye, and Peng Huang, “DartCSim+: Enhanced CloudSim with the Power and Network Models Integrated,” in *2013 IEEE Sixth International Conference on Cloud Computing*, 2013, pp. 644–651.

[93] S. K. Garg and R. Buyya, “NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations,” in *2011 Fourth IEEE International Conference on Utility and Cloud Computing*, 2011, pp. 105–113.

[94] Eucalyptus, “HP Helion Eucalyptus | Open Source Private Cloud Software.” [Online]. Available: https://www.eucalyptus.com/. [Accessed: 28-Aug-2015].

[95] TPM, “Trusted Platform Module (TPM) - Trusted Platform Module Summary_04292008.pdf.” [Online]. Available: http://www.trustedcomputinggroup.org/files/resource_files/4B55C6B9-1D09-3519-AD916F3031BCB586/Trusted Platform Module Summary_04292008.pdf. [Accessed: 04-Mar-2016].

[96] J. Luna, N. Suri, M. Iorga, and A. Karmel, “Leveraging the Potential of Cloud Security Service-Level Agreements through Standards,” *IEEE Cloud Comput.*, vol. 2, no. 3, pp. 32–40, May 2015.

[97] S. Pearson and A. Benameur, “Privacy, Security and Trust Issues Arising from Cloud Computing,” in *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, 2010, pp. 693–702.

[98] R. K. L. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, and B. S. Lee, “TrustCloud: A Framework for Accountability and Trust in Cloud Computing,” in *2011 IEEE World Congress on Services*, 2011, pp. 584–588.

[99] K. M. Khan and Q. Malluhi, “Establishing Trust in Cloud Computing,” *IT Prof.*, vol. 12, no. 5, pp. 20–27, Sep. 2010.

[100] A. Nagarajan and V. Varadharajan, “Dynamic trust enhanced security model for trusted platform based services,” *Futur. Gener. Comput. Syst.*, vol. 27, no. 5, pp. 564–573, May 2011.

[101] K. Ren, C. Wang, and Q. Wang, “Toward secure and effective data utilization in public cloud,” *IEEE Netw.*, vol. 26, no. 6, pp. 69–74, Nov. 2012.

[102] D. Andert, R. Wakefield, and J. Weise, “Trust Modeling for Security Architecture Development.” [Online]. Available: http://www-it.desy.de/common/documentation/cd-docs/sun/blueprints/1202/817-0775.pdf. [Accessed: 19-Apr-2016].

[103] M.-J. Sule, M. Li, G. A. Taylor, and S. Furber, “Deploying trusted cloud computing for data intensive power system applications,” in *2015 50th International Universities Power Engineering Conference (UPEC)*, 2015, pp. 1–5.

[104] ssh communication Security, “Why SSH Communication Security?,” 2016. [Online]. Available: http://www.ssh.com/.

[105] K. A. Beaty, J. M. Chow, R. L. F. Cunha, K. K. Das, M. F. Hulber, A. Kundu, V. Michelini, and E. R. Palmer, “Managing sensitive applications in the public cloud,” *IBM J. Res. Dev.*, vol. 60, no. 2–3, pp. 4:1–4:13, Mar. 2016.

[106] L. A. Zadeh, “The Concept of a Linguistic Variable and its Application to Approximate Reasoning,” *J. Inf. Sci.*, vol. 8, pp. 199–249, 1975.

[107] J. Huang and D. M. Nicol, “Trust mechanisms for cloud computing,” *J. Cloud Comput. Adv. Syst. Appl.*, vol. 2, no. 1, p. 9, 2013.

[108] R. Shaikh and M. Sasikumar, “Trust Model for Measuring Security Strength of Cloud Computing Service,” *Procedia Comput. Sci.*, vol. 45, pp. 380–389, 2015.

[109] Wu Xu, “A Fuzzy Reputation-based Trust Management Scheme for Cloud Computing,” *Int. J. Digit. Content Technol. its Appl.*, vol. 6, no. 17, pp. 437–445, Sep. 2012.

[110] W. Fan and H. Perros, “A novel trust management framework for multi-cloud environments based on trust service providers,” *Knowledge-Based Syst.*, vol. 70, pp. 392–406, Nov. 2014.

[111] C. Qu and R. Buyya, “A Cloud Trust Evaluation System Using Hierarchical Fuzzy Inference System for Service Selection,” in *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, 2014, pp. 850–857.

[112] Z. Raghebi and M. R. Hashemi, “A new trust evaluation method based on reliability of customer feedback for cloud computing,” in *2013 10th International ISC Conference on Information Security and Cryptology (ISCISC)*, 2013, pp. 1–6.

[113] H. Banirostam, A. Hedayati, A. K. Zadeh, and E. Shamsinezhad, “A Trust Based Approach for Increasing Security in Cloud Computing Infrastructure,” in *15th International Conference on Computer Modelling Simulation*, 2013.

[114] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, “‘DACS-MACS: Effective Data Access Control for Multiauthority Cloud Storage Systems,’” *IEEE Trans. Inf. Forensics Secur.*, 2013.

[115] L. Gu, C. Wang, Y. Zhang, J. Zhong, and Z. Ni, “Trust Model in Cloud Computing Environment Based on Fuzzy Theory,” *Int. J. Comput. Commun. Control*, vol. 9, no. 5, p. 570, Aug. 2014.

[116] D. Sun, G. Chang, L. Sun, F. Li, and X. Wang, “A dynamic multi-dimensional trust evaluation model to enhance security of cloud computing environments,” *Int. J. Innov. Comput. Appl.*, 2012.

[117] E. D. Canedo and R. R. de C. and R. de O. A. Rafael Timóteo de Sousa Junior, “TRUST MEASUREMENTS YELD DISTRIBUTED DECISION SUPPORT IN CLOUD COMPUTING,” *Int. J. Cyber-Security Digit. Forensics*, vol. 1, no. 2, pp. 140–151, 2012.

[118] V. Prasath, N. Bharathan, N. N.P, N. Lakshmi, and M. Nathiya, “Fuzzy Logic In Cloud Computing,” *Int. J. Eng. Res. Technol.*, vol. Vol.2, no. Issue 3 (March - 2013), Mar. 2013.

[119] H. Liao, Q. Wang, and G. Li, “A Fuzzy Logic-Based Trust Model in Grid,” in *2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*, 2009, vol. 1, pp. 608–614.

[120] S. N. Sivanandam, S. Sumathi, and S. N. Deepa, *Introduction to Fuzzy Logic using MATLAB*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[121] V. Chang and M. Ramachandran, “Towards achieving Data Security with the Cloud Computing Adoption Framework,” *IEEE Trans. Serv. Comput.*, vol. 9, no. 1, pp. 1–1, 2015.

[122] P. B. Osofisan, “Fuzzy Logic Control of the Syrup Mixing Process in Beverage Production,” *Leonardo J. Sci.*, no. 11, pp. 93–108, 2007.

[123] “Dropbox gears up for new EU data protection rules.” [Online]. Available: http://www.computerweekly.com/news/450280565/Dropbox-gears-up-for-new-EU-data-protection-rules?utm_medium=EM&asrc=EM_ERU_55332335&utm_campaign=20160405_ERU Transmission for 04/05/2016 %28UserUniverse: 2007572%29_myka-reports@techtarget.com&utm_source=ERU&src=5497085. [Accessed: 17-Apr-2016].

Appendices

A1: Adding pre-built Ubuntu Cloud image to Eucalyptus

1. Inside your terminal window on the front-end node load the admin credentials for eucalyptus
    a. ***source ~/credentials/admin/eucarc***
2. Download and install the kexec-loader kernel using instructions from the kexec-loader web-page, note the following command is all on one line
    a. ***wget -O vmlinuz https://github.com/monolive/euca-single-kernel/blob/master/examples/vmlinuz?raw=true***
    b. Bundle the ubuntu kernel for eucalyptus
        i. ***euca-bundle-image -i vmlinuz -r x86_64 --kernel true***
        ii. Note the path/location of the manifest.xml
    c. Upload the kernel to eucalyptus
        i. ***euca-upload-bundle –d {path_to_directory_of_manifest} –b kexec –m full_path_to.manifest.xml***
    d. Register the kernel to walrus
        i. ***euca-register –n kexec-vmlinux kexec/vmlinuz.img.manifest.xml***
        ii. Note the image ID returned for the kernel
3. Download and install the kexec-loader initrd file using instructions from the kexec-loader web-page
    a. ***wget -O initrd-kexec https://github.com/monolive/euca-single-kernel/blob/master/examples/initrd-kexec_load?raw=true***
    b. Bundle the ubuntu ramdisk for eucalyptus
        i. ***euca-bundle-image -i initrd-kexec -r x86_64 --ramdisk true***
        ii. Note the path/location of the manifest.xml
    c. Upload the ramdisk to eucalyptus
        i. ***euca-upload-bundle –d {path_to_directory_of_manifest} –b kexec –m full_path_to.manifest.xml***
    d. Register the ramdisk to walrus
        i. ***euca-register –n kexec-initrd kexec/initrd-kexec.img.manifest.xml***
        ii. Note the ID returned for the ramdisk
4. For an ubuntu distribution of your choice, download the tar.gz file containing the rootfs image, kernel and loader. Note, you may have to scroll down the page to get to the tar.gz versions

   a. wget http://cloud-images.ubuntu.com/saucy/current/saucy-server-cloudimg-amd64.tar.gz
   b. Extract the files from the archive
      i. ***tar -zxvf saucy-server-cloudimg-amd64.tar.gz***
      ii. *Note that the tar archive also contains a kernel but we shall not use these. As the kexec utility will determine and call the right kernel located inside the rootfs image file.*
   c. Bundle the root filesystem image for eucalyptus
      i. ***euca-bundle-image –i saucy-server-cloudimg-amd64-disk1.img -r x86_64***
      ii. Note the path/location of the manifest.xml file (under /var/tmp)
   d. Upload the image to the eucalyptus
      i. ***euca-upload-bundle –d {path_to_directory_of_manifest} –b saucy –m {full_path_to.manifest.xml_from_step_above}***
   e. Register the image
      i. ***euca-register -n ubuntu-saucy --kernel {kernel_id_from_step_2d} --ramdisk {initrd_id_from_step_3d} {saucy/saucy-server-cloudimg-amd64-disk1.img.manifest.xml}***
   f. Note the name of the emi shown
5. Verify that your image(s) is now listed (kernel – eki, ramdisk – eri and rootfs – emi)
   a. euca-describe-images
6. Optionally set permissions on the new image to be launch-able by all users.
   a. euca-modify-image-attribute -l –a all {emi-62443F41}
7. Create a new key-pair for use with Eucalyptus
   a. euca-create-keypair mytest –f mytest.private
8. Launch an instance of your virtual machine using the command
   a. euca-run-instances –k mytest {emi-62443F41}
   b. Note the instance id returned
9. Check the status of your instance
   a. euca-describe-instances {instance_id}
10. Once the status of your instance is "running", you can now ssh into your instance, using the key-file in step 23.
    a. ssh -i mytest.private -v root@{IP_of_running_instance}
11. To shutdown your instance, simply from the server run the
    a. euca-terminate-instances {id_of_instance}

A2: Creating a cloud image from an Ubuntu 13.10 CD/ISO disk-image:

1. Install virt-viewer, tightvnc and other applications
   a. ***yum install virt-viewer tightvnc virt-manager qemu-kvm***
2. Create a raw qemu image file of about 2GB (less than 5GB is better) in a directory of your choice
   a. ***qemu-img create -f raw /scratch/ubuntu.img 2G***
3. Ensure that the raw disk has an msdos style label
   a. ***parted /scratch/ubuntu.img mklabel msdos***
4. Start the installation of the virtual machine
   a. ***virt-install --name ubuntu13.10 --ram 1024 --os-type linux -c /home/onime/Downloads/ubuntu-13.10-server-amd64.iso --disk path=cloudtest.img,device=disk,bus=virtio --graphics vnc,listen=0.0.0.0 --force***
   b. It may be necessary to specify the network type
   -w NETWORK,bridge=br0,model=virtio
5. During installation
   a. DO NOT PERFORM GUIDED PARTITIONING
      i. Select manual partitioning and create only 1 primary ext4 partition for / on the whole disk. DO not create or use a swap partition or setup additional partitions.
   b. create account with username ec2-user
      i. Set the password to something you remember ..
6. After installation check the vm details in virt-manager.
   a. Set the disk model to virtio
   b. Set the network device model to virtio
   c. Set the NIC Source device to the entry with NAT
      i. If there is an error about o the entry with NATault has not been started" then
         1. Run the command
            a. ***virsh net-start default***
         2. In the virt-manager settings click **Cancel** and then **reselect** the NAT entry in the Source Device settings
            a. **Apply** should work OK now.
7. Start the vm and login as ec2-user

a. Verify the status of the network using the **ifconfig** command, if there is no IP on eth0 then

  i. Edit the interfaces file and add a new block for eth0

    1. ***sudo vi /etc/network/interfaces***
    2. Add the following 3 lines, note leave a blank line before the new lines

```
#default network interface
auto eth0
iface eth0 inet dhcp
```

    3. Save and exit the file
    4. Restart networking

      a. ***sudo service networking restart***

b. Check the status of the network using the **ifconfig** command and optionally update packages using the following commands

  i. ***sudo apt-get update***

  ii. ***sudo apt-get upgrade***

c. Modify the sudo settings

  i. ***sudo visudo***

    1. Modify the line with %sudo to

```
%sudo ALL=(ALL:ALL) NOPASSWD: ALL
```

    2. Add the following line below the above line

```
ec2-user  ALL=(ALL:ALL)       NOPASSWD: ALL
```

    3. Save and exit the file (for vi use ***:wq***)

d. Modify the disk label for the disk

  i. ***sudo e2label /dev/vda1 cloudimg-rootfs***

e. Modify **/etc/fstab** to mount by label

  i. Modify the file /etc/fstab and change the entry for root to

    1. `LABEL=cloudimg-rootfs          /   ext4   defaults   0  0`

  ii. Comment or remove any reference to swap

```
#UUID=a954336633683638  none  swap  swap  0 0
```

  iii. Save the file

f. Modify grub and enable serial console

  i. ***sudo vi /etc/default/grub***

    1. Add the following 2 lines and remove other entries that have the same name part (before the = sign).

```
GRUB_CMDLINE_LINUX="console=tty1 console=ttyS0”
GRUB_TERMINAL=console
```

```
GRUB_DISABLE_LINUX_UUID=true
```

2. Save and exit the file
3. Update the grub configuration
    a. ***sudo update-grub***

g. Update the initramfs volume

- ***i.*** ***sudo update-initramfs -u***

h. Install the rc.local script for updating ssh keys (for root user).

- ***i.*** ***sudo wget –O /etc/rc.local https://raw.github.com/eucalyptus/Eucalyptus-Scripts/master/rc.local***
- ***ii.*** ***sudo chmod a+rx /etc/rc.local***

i. Add some security enhancements

- ***i.*** By default, the rc.local script will setup ssh-keyed login for user root. You can disable this by commenting out the right block inside the file /etc/rc.local

```
#if [ ! -f /root/.ssh/authorized_keys ]; then
#    echo >> /root/.ssh/authorized_keys
#    curl --retry 3 --retry-delay 10 -m 45 -s http://169.254.169.254/latest/meta-
data/public-#keys/0/openssh-key | grep 'ssh-rsa' >> /root/.ssh/authorized_keys
#    echo "AUTHORIZED_KEYS:"
#    echo "************************"
#    cat /root/.ssh/authorized_keys
#    echo "************************"
#fi
```

- ***ii.*** Ensure user root has no pre-existing authorized_keys by adding the following line before the above block.

```
echo > /root/.ssh/authorized_keys
```

- ***iii.*** Modify /etc/rc.local to also perform keylogin for user ec2-user by adding the following block.

```
echo > /home/ec2-user/.ssh/authorized_keys
curl --retry 3 --retry-delay 10 -m 45 -s http://169.254.169.254/latest/meta-
data/public-keys/0/openssh-key | grep 'ssh-rsa' >> /home/ec2-
user/.ssh/authorized_keys
 echo "AUTHORIZED_KEYS:"
 echo "************************"
 cat /home/ec2-user/.ssh/authorized_keys
echo "************************"
chown ************************"ized_k
chmod 0600 /home/ec2-user/.ssh
chmod 0640 /home/ec2-user/.ssh/authorized_keys
```

iv. Modify the /etc/rc.local to optionally lock the passwords of root and ec2-user accounts by adding the following lines below the above block.

/usr/bin/passwd -l ec2-user

/usr/bin/passwd -l root

v. Save the file /etc/rc.local

j. Install cloud-init

i. ***sudo apt-get install cloud-init***

k. Upload the kernel and ramdisk to the host

i. ***tar ad the kernel and ramdisk to the hostck the passwords***

ii. ***scp /tmp/ubuntu-kernel.tgz {your_username}@{real_ip_of_host}***

l. Power off the machine

i. ***sudo poweroff***

8. Extract the root partition from the qemu disk image using the following steps

a. First determine the sector size (512) and start of partition 1 (usually 2048) in the image file using the following command

i. ***fdisk determine the se***

b. Use dd to extract just the raw partition (vda1), replace 512 with correct sector size and skip with correct start of 1st partition as report by the fdisk command.

i. ***dd if=ubuntu.img bs=512 skip=2048 of=rootfs.img***

9. Inside your terminal window load the admin credentials for eucalyptus

a. ***source /root/credentials/admin/eucarc***

10. Untar tar the kernel and ramdisk

a. ***tar r tar the kernel and ramdiskme/ubuntu-kernel.tgz***

11. Bundle the ubuntu kernel for eucalyptus

a. ***euca-bundle-image -i /scratch/vmlinuz* -r x86_64 --kernel true***

b. Note the path/location of the manifest.xml

12. Upload the kernel to eucalyptus

a. ***euca-upload-bundle o eucalyptusdirectory_of_manifest} calubuntu13.10 –m full_path_to.manifest.xml***

13. Register the kernel to walrus

a. ***euca-register ernel to walrus-kernel ubuntu13.10/vmlinuz.img.manifest.xml***

b. Note the image ID returned for the kernel

14. Bundle the ubuntu ramdisk for eucalyptus

   a. ***euca-bundle-image -i /scratch/initrd* -r x86_64 --ramdisk true***
   b. Note the path/location of the manifest.xml
15. Upload the ramdisk to eucalyptus
    a. ***euca-upload-bundle to eucalyptusirectory_of_manifest} eucalyptus3.10 ory_of_manifest} eucalyptusl***
16. Register the ramdisk to walrus
    a. ***euca-register an ubuntu_13_10-initrd ubuntu13.10/initrd.img.manifest.xml***
    b. Note the ID returned for the ramdisk
17. Bundle the root filesystem image for eucalyptus
    a. ***euca-bundle-image lesystem image for eucalyptus6_64***
    b. Note the path/location of the manifest.xml file (under /var/tmp)
18. Upload the image to the eucalyptus
    a. ***euca-upload-bundle the eucalyptusectory_of_manifest} the eucalyptus ctory_of_manifest} the eucalyptus_step_17***
19. Register the image
    a. ***euca-register magebuntu_13_10 r magel {kernel_id_from_step_14} alyptusk {initrd_id_from_step_17} ubuntu13.10/rootfs.img.manifest.xml***
    b. Note the name of the emi shown
20. Verify that your image(s) is now listed (kernel g.manifest.xmlp)fest.xmlt sector size an***euca-describe-images***
21. Optionally set permissions on the new image to be launch-able by all users.
    a. ***euca-modify-image-attribute -l he new image to be lau***
22. Create a new key-pair for use with Eucalyptus
    a. ***euca-create-keypair mytest se with Eucalyptu***
23. Launch an instance of your virtual machine using the command
    a. ***euca-run-instances of your virtual machin}***
    b. Note the instance id returned
24. Check the status of your instance
    a. ***euca-describe-instances {instance_id}***
25. Once the status of your instance is }achin} using the commandh into your instance, using the key-file in step 23.
    a. ***ssh -i mytest.private -v root@{IP_of_running_instance}***
26. To shutdown your instance, simply from the server run the
    a. ***euca-terminate-instances {id_of_instance}***

A3: Creating an EBS backed image for Eucalyptus:

1. Start/run a new instance of the image you want to convert to an EBS backed image. This will be used as a helper instance.
    a. ***source ~/credentials/admin/eucarc***
    b. ***euca-describe-images***
        i. *Note the the id from the image you choose*
    c. ***euca-create-keypair –f mykey mykey***
    d. ***euca-run-instances -k mykey {image_id_from_step_1b}***
        i. *Note the new instance ID*
2. Create a new ebs volume, big enough to hold the rootfs, hint you can use euca-describe-clusters to identify and select your cluster name. The size is specified as number of GB. It should be larger than size of vda.
    a. ***euca-create-volume -z {cluster_name} -s 6***
        i. Note volume id of new volume
3. Attach the volume to the running instance from step 1. Note this assumes that it will be the second disk of the instance (vbd), if it is not, then please change to vdc or other suitable device.
    a. ***euca-attach-volume {id_of_volume_from_step_2a} -i {id_of_instance_from_step_1d} -d vbd***
4. The following procedures are performed inside the "helper" instance
    a. Login to the instance
        i. ***ssh -i mykey ec2-user@{ip_of_instance}***
    b. Become root user
        i. ***sudo su –***
    c. verify that new disk (vdb) is larger than the size of old disk (vda) from the output of the following two commands
        i. ***fdisk -l /dev/vda***
        ii. ***fdisk -l /dev/vdb***
        iii. *If the new disk is not bigger then detach it using* ***euca-detach-volume {volume_id_from_step_2a}*** *and delete it using* ***euca-delete-volume {volume_id_from_step_2a}*** *and repeat step 2a.*
    d. Setup grub on the disk, note this command maybe called grub-setup instead of grub-bios-setup.
        i. ***grub-bios-setup /dev/vda***
            1. If there is an error about missing boot.img then use the following command

a. grub-bios-setup -b i386-pc/boot.img -c i386-pc/core.img /dev/vda

ii. grub-install /dev/vda

e. Ensure network configuration is clean
   i. Use one of the following commands to remove persistent network configuration file
      1. rm -f /etc/udev/rules.d/70-persistent-net.rules
      2. rm -f /etc/udev/rules.d/*persistent*net*
      3. rm -f `grep -l eth0 /etc/udev/rules.d/*`
   ii. Remove hardware mac address from the file /etc/sysconfig/network-scripts/ifcfg-eth0 if it exists
      1. perl –pi –e 's/^HWADDR/#HWADDR/g' /etc/sysconfig/network-scripts/ifcfg-eth0

f. Copy the root disk (should be vda) to the new device (should be vbd) using the dd command:
   i. dd if=/dev/vda of=/dev/vdb bs=1M
      1. Note: This command will take a while to run.

g. When complete, leave the ssh session open as it will be needed again in step 7

5. Detach the volume from the instance
   a. euca-detach-volume {volume_id_from_step_2a}
   b. Wait 1 minute for the detach to complete
6. Re-attach the volume to the instance, this will allow the now defined partitions (vbd1, vdb2) to now show up
   a. euca-attach-volume {id_of_volume_from_step_2a} -i {id_of_instance_from_step_1d} -d vbd
7. The following procedures are performed inside the "helper" instance
   a. Perform a file system check on the root partition (and other mounted partitions) of the new device (vbd) to ensure the filesystem is marked as clean/OK.
      i. fsck -y /dev/vdb1
      ii. fsck -y /dev/vdb2
8. Detach the ebs volume from the helper instance
   a. ***euca-detach-volume {id_of_volume_from_step_2a}***
   b. You may want to terminate the helper instance as it is no longer needed
      i. ***euca-terminate-instance {instance_id_from_step_1d}***
9. Create a snapshot of the ebs volume

   a. ***euca-create-snapshot {id_of_volume_from_step_2a}***

10. Register the snapshot as a new image on the system
    a. ***euca-register --name ubuntu-ebs --snapshot {id_of_snapshot_from_step_9a} --root-device-name /dev/vda***
       i. *Note the image id from output of command*
11. Make the new image available to all users
    a. ***euca-modify-image-attributes -l -a all {image_id_from_step_10a}***
12. Ensure that the snapshot creation is complete to 100%
    a. ***euca-describe-snapshots {id_of_snapshot_from_step_9a}***
13. Start the new image using the normal euca-run-instances.
    a. ***euca-run-instances -k {mykey} {image_id_from_step_10a}***
       i. *Note instance id from output of command*
       ii. *Note that each instance will get a separate EBS snapshot/volume created for it that is deleted whenever the instance is terminated.*
14. Connecting to new instance is same as before using the ssh command and keys.
    a. ssh -v -i {mykey} ec2-user@{ip_of_instance}
15. Suspend (or stop) the ebs backed instance with the command euca-stop-instances.
    a. ***euca-stop-instances {instance_id_from_step_13a}***
       i. *Note: This command will free up the IP addresses used but keep the ebs backed storage.*
16. Resume an ebs backed instance with the command euca-start-instances
    a. ***euca-start-instances {instance_id_from_step_13a}***
17. Terminating an instance is with the normal euca-terminate-instance command. **NOTE: IT APPEARS THAT WILL ERASE THE SNAPSHOT DISK-copy THAT WAS CREATED FOR THE INSTANCE, maybe you wanted euca-stop-instances**
    a. ***euca-terminate-instance {instance_id_from_step_13a}***

A4: Adding HADOOP to pre-built Ubuntu (Eucalyptus) cloud image

Requirements

- Ubuntu Cloud image TAR file downloaded from one of the following locations
  - http://cloud-images.ubuntu.com/saucy/current/
  - OR
  - http://cloud-images.ubuntu.com/
  -
- hadoop-2.4.0.tar.gz TAR file from http://www.motorlogy.com/apache/hadoop/common/current/
- Eucalyptus cloud server.

Procedure/STEPS (NOTE ALL COMMANDS SHOULD BE RUN AS root)

27. Inside your terminal window on the front-end node load the admin credentials for eucalyptus
    - ***a. source ~/credentials/admin/eucarc***
28. For an ubuntu distribution of your choice, download the tar.gz file containing the rootfs image, kernel and loader. Note, you may have to scroll down the page to get to the tar.gz versions
    - ***a.*** wget http://cloud-images.ubuntu.com/saucy/current/saucy-server-cloudimg-amd64.tar.gz
    - ***b.*** Extract the files from the archive
      - ***i. tar -zxvf saucy-server-cloudimg-amd64.tar.gz***
      - ***ii.*** *Note that the tar archive also contains a kernel but we shall not use these but use those from inside the IMG file.*
29. Create a new and bigger img file from the existing one, to have space for the hadoop application
    - ***a.*** Create a blank image file of the right size, 3GB in this case
      - ***i. dd if=/dev/zero of=mynewfile.img bs=1M count=3072***
    - ***b.*** Copy the existing image to the new file
      - ***i. dd if=saucy-server-cloudimg-amd64.img of=mynewfile.img conv=notrunc,nocreat bs=10M***

   c. Check the filesystem using e2fsck
      i. ***e2fsck he filesystem us***
   d. Resize the image of the copied file
      i. ***resize2fs mynewfile.img***
   e. Check the file system of the new file
      i. ***e2fsck he file system o***

30. Mount the new image file under /mnt and prepare it for modifications
    a. Mount the image
       i. ***mount the imagemage file under /***
    b. Mount the proc, sys and dev on the new file system
       i. ***mount the proc, sys and dev o***
       ii. ***mount the proc, sys and dev***
       iii. ***mount the proc, sys and dev***
31. Copy the name server configuration to the image
    a. ***mkdir he name server configu***
    b. ***cp /etc/resolv.conf /mnt/run/resolvconf/***
32. Now enter the image file system for modifications. The prompt should change, you might get an error about groups, which may be safely ignored
    a. ***chroot /mnt***
33. Prepare the image for updates and perform
    a. ***rm pare the image for upda***
    b. ***mkdir e the image for updates and p***
    c. ***apt-get update***
    d. Upgrade the openssh server/client to the latest patch level
       i. ***apt-get install openssh-server openssh-client***
34. Install Java
    a. ***apt-get install default-jdk***
35. Download and install the hadoop tar file
    a. Change directory to the root user folder
       i. ***cd /root***
    b. Download

        i. ***wget http://www.motorlogy.com/apache/hadoop/common/current/hadoop-2.3.0.tar.gz***

   c. Untar the archive

        i. ***tar r the archiveotorlogy.com***

   d. Install the extracted directory to /usr/local

        i. ***mv hadoop-2.4.0 /usr/local/hadoop***

36. Create a hadoop startup configuration file for all users

   a. Edit the file /etc/profile.d/hadoop.sh and add the indicated lines

        i. ***vi /etc/profile.d/99_hadoop.sh***

```
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_INSTALL=/usr/local/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_YARN_HOME=$HADOOP_INSTALL
export
HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/li
b/native
export HADOOP_OPTS="-
Djava.library.path=$HADOOP_INSTALL/lib"
#HADOOP VARIABLES END
```

   b. Make the file executable

        i. ***chmod a+rx /etc/profile.d/99_hadoop.sh***

37. Edit various hadoop configuration files and setup single node operation

   a. Edit /usr/local/hadoop/etc/hadoop/hadoop-env.sh and set JAVA_HOME to the indicated value

        i. ***vi /usr/local/hadoop/etc/hadoop/hadoop-env.sh***

```
JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

b. Edit core-site.xml and add an entry for hdfs (indicated lines) in between the <configuration> and </configuration>

   i. vi /usr/local/hadoop/etc/hadoop/core-site.xml

```
<property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:9000</value>
</property>
```

c. Edit yarn-site.xml and add entries for the default mapreduce and nodemanager in between the existing <configuration> and </configuration> entries

   i. vi /usr/local/hadoop/etc/hadoop/yarn-site.xml

```
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.aux-
services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
```

d. Create the mapreduce site file from the template

   i. cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template /usr/local/hadoop/etc/hadoop/mapred-site.xml

   ii. Edit the mapred-site.xml and add the indicated entries in between the <configuration> and </configuration> entries.

      1. ***vi /usr/local/hadoop/etc/hadoop/mapred-site.xml***

```
<property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
</property>
```

e. Create the hdfs storage directories and configure the hadoop hdfs

i. Create the hdfs directories

1. ***mkdir -p /var/hadoop_store/hdfs/namenode***
2. ***mkdir -p /var/hadoop_store/hdfs/datanode***

ii. Edit the hdfs-site.xml and add the indicated lines in between the <configuration> and </configuration> tags.

1. ***vi /usr/local/hadoop/etc/hadoop/hdfs-site.xml***

```
<property>
  <name>dfs.replication</name>
  <value>1</value>
 </property>
 <property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/var/hadoop_store/hdfs/namenode</value>
 </property>
 <property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/var/hadoop_store/hdfs/datanode</value>
 </property>
```

38. Add entries in the system startup file to start up hadoop at boot time, Entries must be before the line with tries.> a

a. vi /etc/rc.local

```
#Give ownership of the various hadoop files to user ubuntu
chown oR ubuntu /usr/local/hadoop
chown oR ubuntu /usr/local/hadoop
#Place entry for hostname in /etc/hosts file
X=`hostname | cut -dtname in /etc/hosts fileto
[  -n  name | cut -dtname in /etc/hosts fileto user ubu
[  -z  name | cut -dtname in /etc/hosts filetoame`r ubuntuat boot
#Start hadoop as user ubuntu
su ubuntu oop as user ubuntu /etc/hosts fi
```

39. Create a new file for starting up hadoop as a normal user.

a. vi /etc/start-hadoop-as-user.sh

```
#/bin/bash
source  /etc/profile.d/99_hadoop.sh
```

```
/bin/rm /etc/profi/id_rsa*
#Create an ssh key for user ubuntu Note after normal user.c/hosts
time, Entrie
ssh-keygen ssh key for user ubuntu Note a
#Ensure that user-ubuntu can login without the need to supply a
password
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 0644  ~/.ssh/authorized_keys
ssh d 0644  ~/.ssh/authorized_keysthorized_key
ssh d 0644  ~/.ssh/authorized_keysthorized_k
ssh d 0644  ~/.ssh/authorized_keysthorized_keys
#Format the hdfs filesystem as user ubuntu
/usr/local/hadoop/bin/hdfs namenode ubuntu_
#Start up Hadoop
/usr/local/hadoop/sbin/start-dfs.sh
/usr/local/hadoop/sbin/start-yarn.sh
```

- ***b.*** Make the script executable by all users
    - ***i.*** ***chmod a+rx /etc/start-hadoop-as-user.sh***

40. Exit from the image/ubuntu
    - ***a.*** ***exit***
41. Copy the vmlinuz and initrd files from the image
    - ***a.*** ***cp /mnt/boot/vmlinuz-3.11.0-18-generic .***
    - ***b.*** ***cp /mnt/boot/initrd.img-3.11.0-18-generic .***
42. Umount the image
    - ***a.*** ***umount /mnt/dev***
    - ***b.*** ***umount /mnt/sys***
    - ***c.*** ***umount /mnt/proc***
    - ***d.*** ***umount /mnt***
43. Bundle the extracted ubuntu kernel for use with Eucalyptus
    - ***a.*** Bundle the image
        - ***i.*** ***euca-bundle-image -i vmlinuz-3.11.0-18-generic -r x86_64 --kernel true***
        - ***ii.*** Note the path/location of the manifest.xml
    - ***b.*** Upload the kernel to eucalyptus

    i. ***euca-upload-bundle o eucalyptusairectory_of_manifest} _64 --kernel truedEntries must be before***

  c. Register the kernel to walrus

    i. ***euca-register ernel to walrususairectory_of_manifest} _64 --kernel truedEntries mus***
    ii. Note the image ID returned for the kernel

44. Bundle the extracted ubuntu ramdisk for eucalyptus

  a. Bundle the initrd

    i. ***euca-bundle-image -i initrd.img-3.11.0-18-generic -r x86_64 --ramdisk true***
    ii. Note the path/location of the manifest.xml

  b. Upload the ramdisk to eucalyptus

    i. ***euca-upload-bundle to eucalyptusnifest.xmlgeneric -r x86_64 --ramdisk trueries must be before***

  c. Register the ramdisk to walrus

    i. ***euca-register amdisk to walrus_directory_of_manifest} x86_64 --ramdisk trueries must***
    ii. Note the ID returned for the ramdisk

45. Upload the img file to eucalyptus

  a. Bundle the root filesystem image for eucalyptus

    i. ***euca-bundle-image lesystem image for eucalyptus***
    ii. Note the path/location of the manifest.xml file (under /var/tmp)

  b. Upload the image to the eucalyptus

    i. ***euca-upload-bundle d {path_to_directory_of_manifest} –b hadoopimg –m {full_path_to.manifest.xml_from_step_above}***

  c. Register the image

    i. ***euca-register -n hadoop-single-node --kernel {kernel_id_from_step_2d} --ramdisk {initrd_id_from_step_3d} {hadoopimg/mynewfile.img.manifest.xml}***

   d. Note the name of the emi shown

46. Verify that your image(s) is now listed (kernel rnel_id_from_step_2d} --ramdisk {initrd
    a. ***euca-describe-images***
47. Optionally set permissions on the new image to be launch-able by all users.
    a. ***euca-modify-image-attribute -l he new image to 43F41}***
48. Create a new key-pair for use with Eucalyptus
    a. ***euca-create-keypair myhadoop with Eucalyptus244***
49. Launch an instance of your virtual machine using the command
    a. ***euca-run-instances of your virtual machine {emi-62443F41}***
    b. Note the instance id returned
50. Check the status of your instance
    a. ***euca-describe-instances {instance_id}***
51. Once the status of your instance is }achine {emi-62443F41}and by all users.disk {initrd_id_from_step_3d} {hadoopi
    a. ***ssh -i myhadoop.private -v ubuntu@{IP_of_running_instance}***
52. Once inside, check the status of hadoop using the command jps, you should get a list with 6 items as show below
    a. ***jps***

```
1858 ResourceManager
1396 NameNode
2433 Jps
1519 DataNode
1724 SecondaryNameNode
1978 NodeManager
```

53. You can now as user ubuntu run hadoop applications without problems.
54. To shutdown your instance, simply from the server run the

euca-terminate-instances {id_of_instance}