Avoiding Dark Cloud: Secure Storage and Trusted Computing

Joolokeni Haimbala

Department of Computer Science

University of Westminster

A Dissertation Submitted to the Department of Computer Science and the Committee on Graduate Studies of University of Westminster in Partial Fulfillment of the Requirements for the

Computer Science Msc Cyber Security and Forensics

London, UK, 2016

Academic Supervisor: Dr. Antonis Michalas, University of Westminster

Day of the defense: September 19, 2016

Abstract

Cloud Computing offers a wide range of attractive benefits, however its adoption is met with concerns regarding the protection of data whilst it is in the cloud. Moving data into the cloud means that the users have less control over their data, this means that the cloud users must trust the cloud provider to protect the data from both external and internal attacks. Several studies has provided security threats in cloud computing and several protocols has been proposed to counter these threats. One of the main concern is data confidentiality in the public cloud which prompted proposals for secure storage systems. Searchable encryption is one of the technique believed to be suitable for providing data confidentiality in the cloud, several techniques has been proposed over the years, however none of this techniques has been implemented by a public cloud services providers. Furthermore several security concerns has been raised in regard to Infrastructure as a Service in the cloud. Several vulnerabilities has been pointed out in the public cloud, raising serious concerns on using IaaS in public clouds. This thesis combines secure storage and trusted computing to provide security in the IaaS. The thesis reviewed Searchable Symmetric Encryption Schemes(SSE) that can be used to provide data confidentiality without compromising the effciency of the cloud services. SSE schemes provides good security notions however there is a tradeoff between effeciency and privacy. The trusted launch protocols are aimed at reducing the abstraction to the virtual machines launch and migration process, although there is good progress in providing a trusted security platform, more need to be done inorder to make close box execution a reality in virtualization.

To Ndawana and Ndeyapo. You two are my sunshine!

Acknowledgements

Firstly, I would like to thank Dr. Antonis Michalas for his support. I consider myself really fortunate to have him as my guide and advisor.

Secondly, I am highly indebted to the Foreign and Commonwealth Office through the Chevening Secretariate and the British High Commission/Namibia for granting an opportunity to study in UK.

Finally, I would like to thank myself for the hardwork, energy and enthusiasm in completing this work. Well done Pawa!

Contents

List of Figures

Thesis Title: Avoiding Dark Clouds: Secure Storage and Trusted Computing
Name of Student: Joolokeni Haimbala
Name and title of supervisor: Dr Antonis Michalas, University of Westminster, Department of Computer Science

Chapter 1

Introduction

Cloud computing is regarded as the next big thing in the industry. It enable users to remotely store data into the cloud; access applications, services and infrastructure through shared-pool of resources. Cloud computing offers a wide range of services to enterprises, government agencies, universities and the general public.

There are three service models of cloud services namely: Software as a Service (SaaS) in which users purchase subscriptions to use software that is centrally hosted by the cloud provider; Platform as a Service (PaaS) in which users are provided with a platform to deploy their applications and Infrastructure as a Service (IaaS) users lease storage, network and hardware from the cloud service provider. All these services are provided on-demand.

Cloud services has become ubiquitous. Individuals use applications to manage their day to day activities, share photos on the cloud, use password management software, use Google drive or Microsoft's Onedrive to manage their documents to mention but a few. Small enterprise that cannot afford the setting up a data center of its own can use cloud computing by choosing the service model that best suit its need. The University of Westminster for example uses Google mail services to provide email services to the staffs and students. All these are examples of how widely cloud services are used.

There are two distinct cloud computing deploying models; private cloud and public cloud. In private cloud, the cloud infrastructure is managed and controlled by the organization itself or trusted parties. Public cloud infrustractures is made available to the general public and is owned by the organization that is selling the services [1]. Customers may move their data to the public cloud to cut the costs of building and managing infrastructures by paying for the services they need. Amazon's S3, Microsoft's Azure, Amazon's EC2 and IBM SmartCloud are examples of public cloud service providers. The benefits of cloud computing paradigm has prompted the need for enterprises and government organizations to adopt cloud computing. For example, businesses can rent computational and storage from the cloud service provider instead of owning their own IT infrastructures [2].

While technological and economical benefits of public clouds are unarguable, cloud computing is prone to security issues. This is because the cloud users and the cloud provider are not in the same security domains, in addition cloud computing architecture provides a layer of abstraction which keeps the users in the "Dark" with their data. The "Dark" area brings uncertainty about the security of the data. Cyber crimes are also on the rise, active attacks and passive attacks can be launched on cloud service providers. In 2012 Dropbox was hacked with over 68 million users' credentials were stolen and leaked on internet [3].

Security and privacy concern has been named as the key reason to the slow adoption of storage and computational services by enterprises and government organizations [4, 5]. This is because security issues are quite significant in the IaaS and PaaS which are service models most likely to be used by the large organizations. Organizations that keep sensitive data such a health data and financial data may have regulatory obligation to protect users data and therefore cannot only rely on the cloud service provider to ensure data confidentiality and integrity.

Storage is one of the function that is mostly used in cloud computing. Users' data is stored on a remote server and therefore security mechanisms should be in place protect users' data from any form of data breach. Virtualization is the backbone of cloud computing, although it provides a broad range of advantages such as multi-tenancy, resources sharing and elasticity; these properties brings security threats. All these vulnerability affects the adoption of cloud services by this group of users.

Cloud specific security solutions that can guarantee verifiable data confidentiality, integrity and privacy may attract enterprises and government agencies to join the cloud [6]. Several mechanisms are employed to provide data confidentiality, however this mechanisms targets mostly external attacks and offer little protection against the cloud providers(Insider Attacks). Therefore, there should be some mechanisms that can provide users with some control over their data.

Data confidentiality can be achieved by using encryption techniques. Searchable Encryption (SE) is a technique that fits the cloud model because it allow users to store encrypted data on the cloud and be able to search over the encrypted data. The SE schemes are used to provide secure storage. For users that opt for *IaaS* service model, computational resources need to be secured against external attacks, tenant-to-tenant attacks and insider attacks.

This study looks into mechanisms that can be used to provide data confidentiality stored on a untrusted cloud provider (secure storage) and integrity of the computations during the virtual machine (VM) launch process hosted on a untrusted cloud provider (trusted launch). There has been an increase in research that leans towards providing secure storage systems in public clouds. The main goal for the secure storage is to provide data confidentiality and integrity. Although many solutions has been proposes, none has been deployed in the real world. The purpose of this study is to evaluate these proposed solutions in order

to establish issues that might be hinder their implementation in the cloud.

1.1 Contributions

The contribution of this work is threefold. More precisely, we analyze several protocols with main aim to strengthen the security of both IaaS and PaaS cloud models. To this end, we first present a list of protocols that focus on the problem of trusted launch of VM instances in IaaS. Second, we extensively study existing approaches that offer a solution for the urgent problem of secure cloud storage. Moreover, during the analysis of the existing secure storage schemes we mainly focus on solutions that are based on SE schemes. As a result, we study many forward-looking designs that even though they have not been fully implemented will be the future in the cloud-based services. Third, we conduct a detailed vulnerability analysis for both the Trusted Launch and the Secure Storage schemes. This analysis, is based on a concrete list of attacks that can be applied to these schemes. The insights we obtain from this analysis allow us to create a list of concrete requirements that a trusted cloud host should have while at the same time provide a proper guidance regarding the requirements for building a secure cloud storage. We hope that this analysis will help protocol designers to build even better trusted launch and secure cloud storage schemes and will eventually provide a clear direction for the construction of secure and privacy-preserving cloud-based services.

1.2 Outline

The rest of this dissertation is divided into the following parts:

- **Chapter 2 – Background:** Discusses the main idea of cloud computing and problems related to cloud storage and trusted computing.
- **Chapter 3– Preliminaries:** Introduces the basic cryptographic primitives that are used throughout the thesis.
- **Chapter 4 – Problem Definition:** Describes the the problem this thesis is addressing.
- **Chapter 5– Existing Protocols:** Presents a detailed list of protocols studied.
- **Chapter 6 – Threats and Attack:** Presents threats and attacks specif to the SSE schemes and virtual machine launch process.
- **Chapter 7 – Analysis and Discussion:** Discuss the strengths and weakness of studied protocols against relevant attacks.

- **Chapter 8 – Conclusions:** Summarizes the thesis and concludes with some future research directions.

Chapter 2

Cloud Computing

This chapter presents a high level overview of cloud computing. Even though cloud computing has become a wide area this chapter will mainly focus on three key aspects that will form the main focus of the thesis. To this end, we will briefly describe the main idea as well as important problems related to the following areas:

- Cloud Storage;
- Trusted Computing;
- Security issues mainly on the IaaS and PaaS cloud environments;

2.1 Cloud Definition

During the last few years, cloud computing has seen an exponential growth and has attracted many researchers as well as key industrial players. Furthermore, cloud computing is considered as the next best thing because of its benefits to both the industry and the end-users'. Cloud computing is defined as an on-demand service model for IT provision, often based on virtualization and distributed computing technologies. This means that businesses and individuals can choose a service model which best suit their needs and therefore pay for what they only use. As a result, cloud-based services has the potential to offer important cost saving services not only to the end-users' but also to companies. By migrating an existing platform to the cloud, organizations can get powerful functionality in the most cost effective manner. Cloud migration can help organizations reduce the need for IT teams to operate and maintain expensive internal infrastructure, reduce software costs and shed at least some of their expensive IT infrastructure and shift computing costs to more manageable operational expenses. Nonetheless, if the transition is not planned carefully, it can lead to disappointing results. The main reason for that is the fact that there is a plethora of available solutions and many of them are not able to meet the specific needs of an existing platform.

Thus, a cloud solution that will not attain the above mentioned criteria can have catastrophic results [7].

There has been different definition of cloud computing, but after 15 drafts NIST has published a formal definition. According to NIST, cloud computing is defined as follows:

Definition 1 (Cloud Computing)**.** *Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.*

The basics of cloud computing that illustrate it's relation to and difference from the traditional network paradigm are explained by NIST [1]. The idea is that customers should be provided with on-demand self-services without human interaction with the cloud provider. These services are provided through resource pooling in which the cloud provider uses a multi-tenant model for pooling resources such as storage, processing, memory, network bandwidth and virtual machines; in order to dynamically serve multiple customers. The customers then uses Broad network to access the services through mechanisms that can be used by heterogeneous thin or thick client platforms such as laptops, mobile phones and tablets.

In addition to that, more characteristic capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. From the consumer side, the available capabilities for provisioning, often appear to be *unlimited* and can be purchased in any quantity at any time. Cloud systems automatically control and optimize resource use by leverage a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported providing transparency for both the provider and consumer of the utilized service.

2.1.1 Service Models

There are the following three *main* different types of service models for cloud computing defined by [1]:

- *Software as a Service (SaaS):* In this service model, applications such as web-mail, customer relations management, photosharing, etc are provided as a service to the users'. These applications run by the cloud service provider and can be accessed by the users' through a web browser or application program interface (API). The users' do not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

- *Platform as a Service (PaaS):* The users outsources the platform to build applications or software. This means that they do not need to worry about installing or managing software and operating systems. Additionally, the consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations.

- *Infrastructure as a Service (IaaS):* In this service model, the user outsources infrastructures such as storage, networks, servers are provided to the user. The user can deploy and manage operating systems, applications and sofware. The cloud provider manages and controls the underlying computing resources.

2.1.2 Deploying Models

Apart from the service models, NIST [1] defined the following deploying models for the cloud:

- *Private Cloud:* The cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on premise or off-premise.

- *Community Cloud:* The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party and may exist on premise or off premise.

- *Public Cloud:* The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.

- *Hybrid Cloud:* The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g cloud bursting for load-balancing between clouds).

2.2 Cloud Security Threats

In Chapter 1 we have seen that one of the major obstacles for enterprises and governments to adopt cloud computing is security. Unlike the traditional IT computing where the user/Enterprise is in control of the IT security; Cloud computing system removes that control. This means that the user/Enterprise must

rely on the Cloud provider for security – in other words, in many cases the cloud provider must be treated as a trusted entity. However, it has been observed that among the most important reasons for the slow adoption of cloud computing is the fear of storing sensitive data online. Without proper security mechanisms to protect user' data from unauthorized access, it is much easier for sensitive information to be leaked to interested third parties, such as insurance companies or potential employers.

As a result, cloud providers have started to invest a lot into cloud security research in order to find a solution that will encourage and motivate both individuals and enterprises to join the cloud. There has been an increase in research that leans towards secure storage systems where they are looking at designing systems that can provide data confidentiality and integrity. Assurance of privacy and data security is one of the fundamental factor that may attract businesses and individuals to join the Cloud [6, 8, 9, 10, 11, 12].

Security threats can be both external and internal. Although there are many different mechanisms to mitigate the external attacks; insider threats still remain a big problem. In a study done by Semantics, 2011; they identified security classes issues that are specific to the cloud environment these are insecure API, malicious insiders, shared resources, denial of service [13, 14, 15, 16], data breach or data loss and stolen credentials. Whereas [4] classified Cloud security threats into three categories: data security, virtualization-related security and application-related security. This research specifically focus on data security and virtualization security.

2.2.1 Cloud Storage

Cloud storage is one of the cloud computing functionality commonly used. The data is stored in multiple servers which are managed by the cloud provider. The actual physical location of the underlying servers is unknown to the users'. The users' only see a particular place with a name where data appear to be stored but in reality the data can be stored on one or more servers at different locations. The actual locations can vary from time to time but the client should not notice any difference. This is one of the essential characteristics of cloud computing defined by [1]. The cloud system therefore should provide a high degree of reliability, availability, performance and data security.

For the entire cloud storage concept to work, there must be at least one server where the data is stored and can be accessed online. The customer then uses a thin or thick client to access the storage through the internet. When a user wishes to upload data to the cloud she uses a client running on a computer, laptop, mobile device or through a web-based interface. This client allows the user to transfer data to the cloud as well as to access already stored data. As a result, a fundamental requirement of cloud-based services is to protect data while stored on the server and when it is in transit.

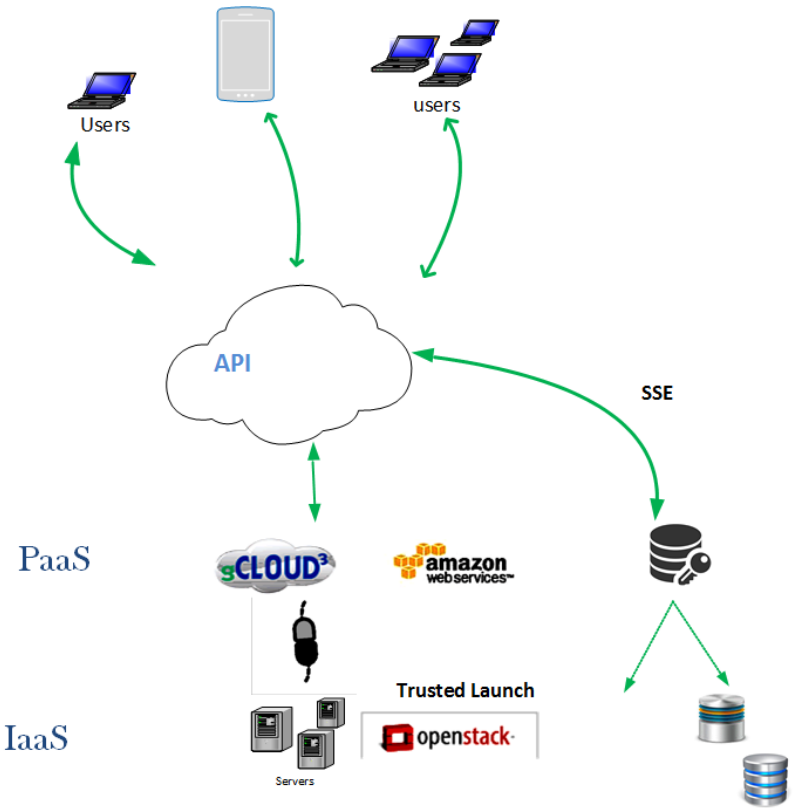

Figure 2.1: Secure Storage

2.2.2 Searchable Encryption

Searchable encryption enables a user to store encrypted data (such as person health records) on a untrusted remote server and be able to search over the encrypted data. The idea of searchable encryption is the same regardless of the cryptographic techniques used. The general model of Searchable Encryption works as follows: There are users and the untrusted remoter server. The users have sensitive files that they would like to store on the server without the server learning the content of these files. The only way to hide the content of these files is by encrypting the files before they are uploaded on the server; therefore the users uploads encrypted files on the server alongside the encrypted index which is used by the server to locate the requested files. The users uses a function to generate a trapdoor for each keyword. To search for a file; the user sends the trapdoor to the server and the server retrieves the files. Although the server does not learn the content of the files, it can learn the trapdoors that are associated with certain files. Searchable encryption can be implemented using either symmetric or asymmetric encryptions, however asymmetric encryption is computational expensive. On the other hand, symmetric encryption is more efficient

which makes it more suitable for the cloud and big databases. This study focus on searchable symmetric encryption schemes.

2.2.3 Virtualization

One of the key component of cloud computing is virtualization. Virtualization is the "creation of a virtual (rather than actual) version of something, such as a server, a desktop, a storage device, an operating system or network resources" [17]; it which allows to share a single physical instance of a resource or an application among multiple customers and organizations. It does so by assigning a logical name to a physical storage and providing a pointer to that physical resource when demanded [17]. Virtualization is used to achieve multi-tenancy in cloud computing it's essential characteristics such as location independence, resource pooling and rapid elasticity. The key characteristics of virtualization are listed below:

- *Partitioning:* enables multiple operating systems to run on one physical machine and divides the system's resources between the virtual machines.
- *Isolation:* Provides security isolation at the hardware level and also control resources.
- *Encapsulation:* Saves the entire state of the VM to files which make it easy for the VM to be moved from one physical device to the next.
- *Hardware independent:* Any virtual machine can be moved to a different physical machine without compatibility issues and as easy as copying and pasting a file. This is made possible by the VM management software like hypervisor.

In order to enable multiple operating systems to run on one physical machine, a virtual machine management (VMM) known as a hypervisor is used. Hypervisor creates a virtual platform on the host computer, on top of which multiple guest operating systems are executed and monitored. This way, multiple operating systems, which are either multiple instances of the same operating system, or different operating systems, can share the hardware resources offered by the host. In [18] authors classified hypervisors into two types as shown in figure 2.3 below:

2.3 Trusted Computing

Trusted computing is a concept developed by the Trusted Computing Group (TCG) [19]. TCG proposed a set of standards for the design of a chip entitled Trusted Platform Module (TPM) which is now included in the hardware many devices. TPM provides the devices with mechanisms to protect systems from unauthorized changes and attacks from malicious software. Standards-based

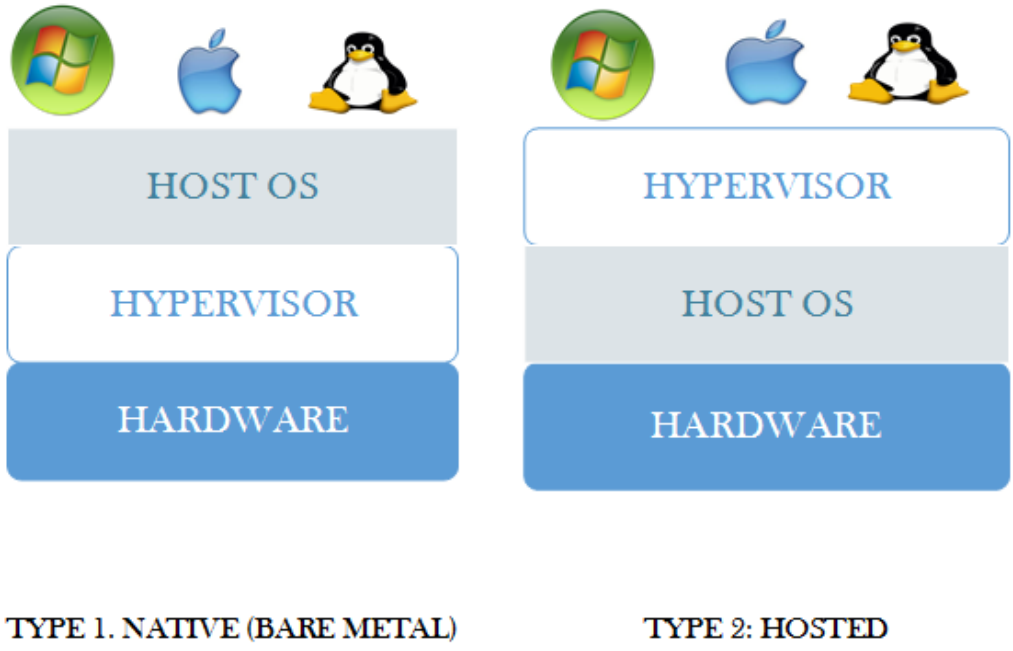

Figure 2.2: Hypervisors Classification

Trusted Computing technologies developed by TCG members are currently deployed in enterprise systems, storage systems, networks, embedded systems, and mobile devices and can secure cloud computing and virtualized systems. Thousands of vendors offer a variety of Trusted Computing-based products, including hardware, applications, and services [19]. The components that constitute the TPM are described below.

2.3.1 Trusted Platform Module

The TPM chip layout is made up by the following *eleven discrete components* 2.3 [19]:

- *Input/Output:* I/O manages the information flow over the communication bus and also uses a protocol to encode/decode messages sent/received by the TPM.
- *Non-volatile:* Stores Endorsement Key (EK), Storage Toot Key (SRK) and specific owner authorization data.
- *Platform Configuration Registers(PCR):* Can be implemented in either volatile or non-volatile storage. These registers are reset at system start or after power loss. TCG specifies a minimum number of registers to implement (16). Registers 0-7 are reserved for TPM use. Registers 8-15 are available for operating system and application use.

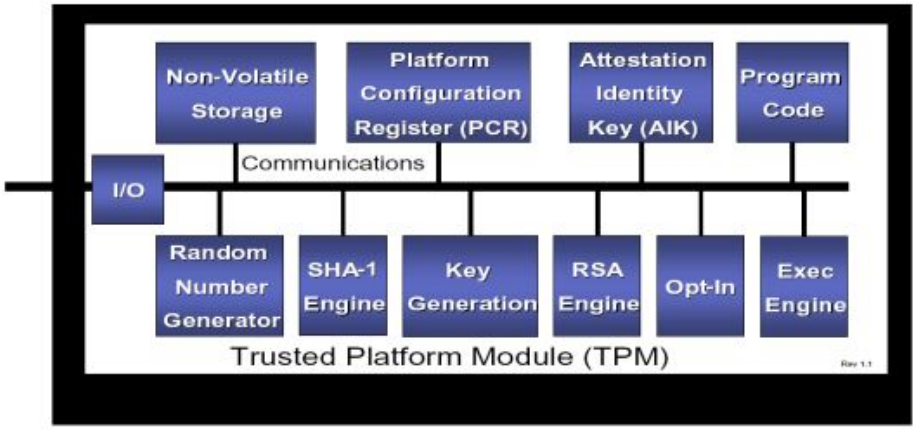

Figure 2.3: TPM Architecture

- *Attestation Identity keys(AIK):* AIK stores persistent keys that are used to sign and authenticate the information provided by the TPM when external attestation is required. In cases where there are multiple users' running on the same platform, the keys can be stored in an encrypted form in an external storage.
- *Program Code:* Contains firmware for measuring platform devices. It is the representation of Core Root of Trust for measurement (CRTM).
- *Random Number Generator (RNG):* Generates random numbers such as nonce or keys.
- *SHA-1 Engine:* Used for creating signatures.
- *RSA Key Generation:* TCG standardizes the RSA algorithm for use in TPM modules. Its recent release into the public domain combined with its long track record makes it a good candidate for TCG. The RSA key generation engine is use to create signing keys and storage keys. TCG requires a TPM to support RSA keys up to a 2048-bit modulus, and mandates that certain keys (SRK and AIKs) must have at least a 2048-bit modulus.
- *RSA Key Generations:* Is used in order to perform the signing, public-key encryption and decryption operations based on the RSA algorithm.
- *Opt-in:* Allows to maintain the activation state of the TPM chip, the possible states being enabled, disabled, deactivated.
- *Execution Engine:* Is a component that executes the operations prescribed by the logic in the program code.

2.3.2 Terra

Terra [20] is a traditional trusted computing platform that uses a thin virtual machine monitor (VMM) to allow multiple virtual machines (VMs) to share resources and run concurrently but independently by running the machines in a a closed box execution platform. It inherits the properties of the traditional VM which are *isolation*, *extensibility*, *efficiency*, *compatibility* and *security*. What distinguish Terra from the traditional VM is the following three properties:

- *Root Secure:* Terra provides a closed secured box platform that ensures that even a user with privileges access cannot inspect or modify a VM that is running.
- *Attestation:* This feature allows an application running in a closed box to cryptographically identify itself to a remote party. That is, to tell the remote party what is running inside the closed box. This allows the corresponding party to put trust in the application [20].
- In a TVMM, a trusted path allows a user' to establish which VMs they are interacting with each other as well as allowing a VM to ensure that it is communicating with a human user' and not with a robot. It also ensures the privacy and integrity of communications between users' and VMs. Therefore, snooping or tampering by malicious programs is prevented.

Even though Terra has been proved to be a secure trusted computing platform it can only provide security on a single host. This means that if it is to be adapted from the IaaS model, it should be implemented to each and every node. The two problems that are not addressed by Terra are explained in the problem definition section.

2.4 Summary

This chapter presented a high level overview of cloud computing; although cloud computing covers a wide area, we presented areas that are relevant to this thesis. Next section, presents preliminaries and definitions that are used throughout this paper.

Chapter 3

Preliminaries

This chapter presents the preliminaries and definitions used throughout the thesis.

Cryptographic Primitives

Definition 2 (Symmetric encryption)**.** *A symmetric encryption scheme is a set of three polynomial time algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *such that:*

- Gen *is a probabilistic algorithm that takes as input a security parameter* λ *and outputs a cryptographic symmetric encryption key* (K)*. The finite set of all possible keys that* Gen *can output is called the key space and is denoted by* κ*. We denote this by* $\mathsf{K} \leftarrow \mathsf{Gen}(1^{\lambda})$*.*

- Enc *takes input* $\mathsf{K} \epsilon \kappa$ *and a message* m *and outputs a ciphertext* $\mathsf{C_m}$*. The encryption algorithm is deterministic and is denoted by* $\mathsf{C_m} := \mathsf{Enc}(\mathsf{m}, \mathsf{K})$*.*

- Dec *is a deterministic algorithm that takes input key* K *and a ciphertext* $\mathsf{C_m}$ *and outputs message* m*. We denote is by* $\mathsf{m} := \mathsf{Dec}(\mathsf{K}, \mathsf{C_m})$*.*

Definition 3 (Asymmetric encryption)**.** *Asymmetric encryption: Also referred to as a public encryption scheme is a set of three polynomial time algorithms* $(\mathsf{Gen}; \mathsf{Enc}; \mathsf{Dec})$ *such that:*

- Gen *is a probabilistic algorithm that takes as input a security parameter* λ *and outputs a public/private key pair* (pk/sk)*.*

- Enc *is probabilistic algorithm that takes as an input public key* pk *and a message* m *and outputs a ciphertext* $\mathsf{C_m}$*. We denote this by* $\mathsf{C_m} := \mathsf{Enc_{pk}}(\mathsf{m})$*.*

- Dec *is a deterministic algorithm that takes as an input private key* sk *and a ciphertext* $\mathsf{C_m}$ *and outputs message* m*. We denote is by* $\mathsf{m} := \mathsf{Dec_{sk}}(\mathsf{C_m})$*.*

Public Key (pk): A public key is known to the public or everyone. It can be used for encrypting messages and verifying digital signatures. A public key for a user (u_i) is denoted as $\mathsf{pk_i}$.

Private Key (sk): A private key is only known by the owner. It can be used to decrypt messages and creating digital signatures. A private key for a user (u_i) is denoted as $\mathsf{sk_i}$.

Cryptographic Hash Function (h): is a technique that, among other things, can be used for verifying the integrity of a message m. A hash of message m is denoted as $\mathsf{h(m)}$.

Digital signature (σ): Is a technique that verifies the integrity and authenticity of a message .The digital signature over a message m is denoted by $\sigma = \mathsf{Sign_{sk}(m)}$. The corresponding signature verification is denoted by $\mathsf{b} = \mathsf{Verify_{pk}(m, \sigma)}$, where $\mathsf{b} = 1$ otherwise $\mathsf{b} = 0$.

Random Oracle (RO): In a **RO** model,cryptographic primitives are replaced by an idealized versions (e.g.,replacing a cryptographic hash function with a genuinely random function). Solutions in the **RO** are often more efficient than solutions in the **ST** but have the additional assumption of idealized cryptographic primitives[21].

Standard Model (ST): In this model, the adversary is limited only by the amount of resources available such as time and computational power. This means that only complexity assumptions are used to prove a scheme secure[21].

3.1 Entities

User (u): A user' [1] who is using a cloud service will be denoted as u_i where i is the unique identifier of the user'. The operations that a user can perform are the following: *a)* register to the service, *b)* generate encryption keys to safely protect her data, *c)* store data in the cloud as well as retrieve and search over her private data that has been sent to the cloud. The set of all users' is denoted as $U = \{u_1, \ldots, u_n\}$.

Cloud Service Provider (CSP): We consider a cloud computing environment based on a trusted IaaS provider like the one described in [22]. The IaaS platform consists of cloud hosts which operate virtual machine guests and communicate through a network. In addition to that, we assume a PaaS provider that is built on top of the IaaS platform and can host multiple outsourced databases. Furthermore, the PaaS provider offers an API through which a developer can built a privacy-privacy preserving application that offers searchable encryption functionality.

[1] In the rest of the paper we will be referring to user' either as user' or as a client.

System Administrator (*sysadmin*)**:** Is a CSP employee account which has privileged access rights.

Trusted Third Party (TTP): For the needs of the thesis, we assume a "trusted third party", which is trusted by the community and plays a key role in the described protocols. We rely on the commonly supported proposition that a large code base normally contains a proportionally large number of vulnerabilities [23]. To reduce the code base, it is important that the TTP only supports the minimal necessary functionality. TTP is able to communicate with components deployed on compute hosts to exchange integrity attestation information, authentication tokens and cryptographic keys. In addition, TTP can attest platform integrity based on the integrity attestation quotes provided by the TPM on the respective compute hosts, as well as seal data to a trusted configuration of the hosts. Furthermore, TTP can verify the authenticity of a client as well as perform necessary cryptographic operations. Finally, the TTP cannot be controlled by the CSP or any other external entity.

Chapter 4

Problem Definition

Cloud computing security threats as pointed in the previous chapter is one of the great concern why individuals and enterprises are relucted to join the cloud. In the studies done by [4, 24, 6, 25, 26, 27] found that data security and virtualization threats are one of the current burning security issues in the Cloud. Sabahi [27] presented security concerns in all three service modes of cloud computing (SaaS,PaaS and IaaS) which included data security, network security, data locality, data integrity, data segregation, data access, authorization and authentication etc. If this problems and threats are solved, this may encourage governments and enterprises to join the cloud. This chapter will present two main problems that this research is addressing; the first problem is with virtualization and the second problem is data security in cloud storage.

4.1 Virtualization security issues

All three service models of cloud computing (SaaS, PaaS, IaaS) leaves some part of computing control power to the cloud user. The client gets more control as we move down the cloud computing stack resulting in the SaaS with the least control and the IaaS with the highest. In *(IaaS)*, the cloud provider such as Amazon's EC2 and Microsoft Azure hosts the infrastructures on behalf of the client. The IaaS provides an abstruction layer of the underlying components of cloud computing such as hardware configuration, Virtual Machine Management(VMM) software such as hypervisor and Networks but the users have complete control of guest operating system, databases, runtimes, managing and deploying applications. The cloud provider have the responsibility to protect tenants from external threats, threats from malicious tenants and threats from its employees.

Cloud providers are however working hard to ensure that they provide secure systems that minimizes the inside threats and therefore reinforce confidences of the clients [28]. For example they protect the physical hardware infrastructures, restrict the number of people that have access to critical components, adopt accountability and auditing procedures. For these reasons the hardware attacks

and side-channel attacks are excluded in this context.

Cloud providers administer and implement the infrastructure level security aspects of the Virtual Machines (VMs), employees of the CSP with the privileged access to the hosts such as a systems administrator can read and manipulate the client's data. The clients leave their data in the hands of a untrusted cloud provider, and thus they have no control over their data, have no means to see what is done to their data; hence there is no assurance to data confidentiality and integrity nor transparency to prove the integrity or confidentiality of their data.

4.2 Secure Storage Security issues

As mentioned in the previous chapter, cloud storage is one of the commonly used function of cloud computing. Data stored in the Cloud just like in the traditional IT systems need to achieve three aspects of data security namely; Confidentiality, Integrity and Availability. When any these aspects is violated it leads to data breach or data loss.Data breach happens when sensitive, private, protected or confidential data has been seen, stolen, altered or become available to an unauthorized person. This is a serious concern especially when data is sensitive and confidential such as health records, personal identity information and intellectual properties.Access to this data violates two data security properties namely confidentiality and integrity. In cases of the organizations and governments; they have an obligation to protect data that is classified as critical data by the regulatory bodies in their jurisdictions and therefore they need guarantee and assurance of data integrity, confidentiality and availability of the data whilst in the cloud.

Confidentiality refers to the prevention or unintentional unauthorized disclosure of information [29]. Therefore confidentiality is achieved if the data is only known, seen or accessible to authorized party. To achieve data confidentiality in the cloud, the following condition must be satisfied; The cloud provider does not learn any information about the client's data. Integrity is achieved when the accuracy, completeness and consistency of data is achieved. This means that data is not changed during transmission or while it is at rest therefore the client should be able to detection of her data by the cloud provider. And availability refers to accessibility of data by the authorized parties whenever it is required.

There are cryptography techniques that can be used to provide data confidentiality. Unfortunately the traditional cryptographic techniques for providing data security can not be directly adopted because of the nature of the cloud computing architecture in which the user looses control of the data. Because the user does not trust the CSP, the techniques that will be used for providing data security in the cloud should allow data verification and computation operations. Therefore to achieve secure storage that guarantees data security in the cloud, the following basic requirements should be met:

1. *The data should be encrypted when it is uploaded:* The User encrypt data

before uploading it to the Cloud and operations such as update, delete and search can be done on the encrypted data. The decryption key is either kept by the client or a trusted third party.

2. *The disk where the data is stored is encrypted:* however the key to this disk is accessible by the cloud provider the cloud provider. Such that the hard drive (D) is $\mathsf{Enc}_{\mathsf{K}}(\mathsf{D})$.

3. *Performance is retained:* The process of secure storage should not degrade the performance of cloud computing performance. Users should be able to enjoy the benefits of cloud computing.

Assume a user u_i has a set of n files $F_i = \{f_1, \ldots, f_n\}$. Furthermore, assume that u_i wishes to upload a file $f_k \in F_i$ to a remote storage offered by the CSP. To do so, u_i first encrypts f_k by using a symmetric secret key K and running $\mathsf{Enc}_{\mathsf{K}}(\mathsf{f_k}) = c_{f_k}$. Then, u_i uploads the generated ciphertext c_{f_k} to the CSP. As a result, as long as K is known only to u_i the CSP cannot extract any valuable information regarding the content of f_k.

4.3 Summary

We looked at two problems, the first one is trusted computing, which require users to launch the VM on a trusted platform such that no adversary can manipulate the VM. If that problem is solved, then the second problem we are faced with is the privacy of data. That is securing data while it is in the cloud from both external and internal attacks. The next chapter will look at the techniques that are developed to solve these two problems.

Chapter 5

ExistingProtocols

This chapter presents protocols that are developed in an effort to solve the problems mentioned in the previous Chapter 4. The first section of this chapter presents protocols related to trusted computing while the second section presents protocols that focus on secure cloud storage.

5.1 Trusted Launch of Virtual Machines

A significant number of research has been done in the field of trusted computing in an effort to build trusted cloud computing environment specifically for the IaaS cloud services. This section presents the existing protocols regarding the trusted launch of a virtual machine in an IaaS environment.

5.1.1 Trusted Computing Re-visited

As it was previously discussed in Chapter 2, TCG introduced the standards for developing the TPM module which is used to protect the systems from unexpected and unauthorized changes by malicious software programs. TPM has a wide range of capabilities and can offer reliable and efficient solutions in many security related protocols. More precisely, TPM offers secure management of cryptographic keys, allows remote attestation of a host, provide the necessary integrity measurements and can also protect private information through sealing and binding. In addition to that, software uses TPM to authenticate the hardware devices and also authenticate platforms because of the RSA keys embedded in the TPM.

5.1.1.1 Remote Attestation

Attestation is a mechanism for software to prove its identity. The primary goal is to prove to the remote entity that the applications running on the machine are trustworthy. The verifier in turn trusts that attestation data is accurate because

it is signed by a TPM whose key is certified by the CA. A basic attestation protocol according to [30] is demonstrated below.

1. The hardware platform is like a Certificate Authority CA for the platform and has a signing key $\mathsf{SK}_{\mathsf{CA}}$ and a public key certificate $\mathsf{Cert}_{\mathsf{CA}}$.
2. When a software or application is started first; it generates a pair of Attestation identity keys (AIK) public/private key. We refer to the software or application as a Client C and it's public/private keys pair as $\mathsf{PK}_{\mathsf{C}}/\mathsf{SK}_{\mathsf{C}}$.
3. C requests the CA to certify its PK_{C} and CA issues a signed certificate
4. When the Client C wants to attest to a remote entity (Server S), C send the certificate $\mathsf{Cert}_{\mathsf{C}}$, the hash of the image $h(image_C)$ and the CA certificate.
5. S verifies if the both the certificates are valid and check if the C's hashed image have a match in the list of trusted applications.
6. S and C can authenticate each other by generating an encrypted session with PK_{C} and the application can decrypt using its SK_{C}.

The hashed image of the application is the heart of attestation, because it proves the integrity of the application. In order for the attestation to be secured the application and the remote entity should use a session key to encrypt their communication, this will protect the attestation process from cold boot attacks [31].

5.1.2 Trusted Cloud Computing Platform (TCCP)

TCCP by Santos, Gummadi and Rodrigues [28] is based on the traditional trusted computing model Terra [20]. Terra consists of a VMM which implements a closed box execution environment such that a guest OS running cannot be modified or tampered with by the privileged user of the host OS. However, Terra can only be implemented on a single host. Therefore, for it to be successful in the cloud computing setup where there are multiple hosts it must be installed on each and every host. This process is however inefficient and impractical in the IaaS model. Having identified that drawback, Santos et al. [28] developed TCCP to address the shortcomings of Terra. TCCP attack model assumes that the Cloud Provider have implemented access control policy, installed surveillance and security devices that secure physical access to the host machines therefore the *SysAdm* have no access to the physical host machines.

TCCP provides closed box execution environment for the user VM in order to ensure that the confidentiality and integrity of the computations are retained by extending the trusted launch platform to the entire *IaaS* backend. TCCP have two protocols, one for trusted launch which ensures that the clients VM is launched on a trusted host and trusted migration which ensures that the VM

migration will only be successful if the VM is migrated to a trusted node. All the components of TCCP are shown in the figure 5.1.

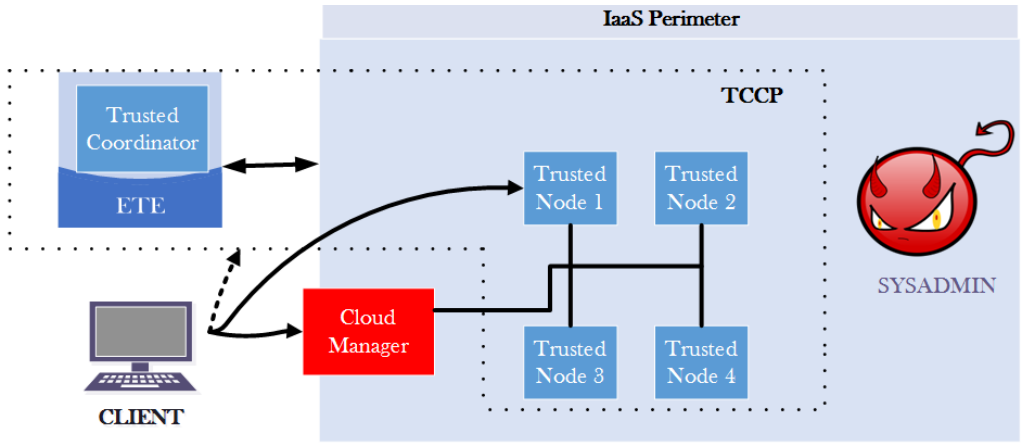

Figure 5.1: TCCP architecture

Each node of the backend runs a TVMM that hosts clients' VMs, and prevents privileged users from inspecting or modifying them. The TVMM protects its own integrity over time, and complies with the TCCP protocols. Nodes are embedded with a certified TPM chip and must go through a secure boot process to install the TVMM [28]. The TVMM supports attestation which allows applications running in a closed box to authenticate itself to remote parties [20]. The Nodes that are located inside the security perimeter and have a TVMM installed are called *trusted nodes* and are managed by the TC. The nodes' registration process involves a node initiating the communication by sending a nonce N to the TC. This is done to prevent impersonation of the TC by an attacker. The TC attest itself to the node using cryptographic keys. The node is registered when the both the node and the TC successfully attest each other. The TC is managed by the ETE for security reasons such as that the cloud provider have no access to the configurations of ETE. Therefore, the TC cannot be tampered with. The protocol steps are explained below.

1. The user' encrypts the VM instance with a session key K_{VM}) and then encrypts the session key and the nonce with the TC trusted public key $VMstate_{K_{VM}}, N, K_{VMPK_{TC}}$ and sends it to the TC through the Cloud Provider (CSP) which allocates the available node N_i;

2. Node N_i identifies itself to the TC, and if it is a trusted node, TC encrypts the session key K_{VM} with the node's Trusted public key K_{VMPK_N}. Only then the node will be able to launch the VM.

.

TCCP achieves two important security guarantees *(1)* the VM cannot successfully run outside the security parameter. Therefore, it can only be launched on a trusted node and *(2)* system administrator cannot temper with the users' VM. Although the protocol provides strong security notions for trusted computing in IaaS, authors are yet to provide a working prototype. Furthermore, the protocol stores the trusted keys in the node's memory, which makes the protocol vulnerable to cold-boot attacks. This attacks are explained in further details in the next chapter.

5.1.3 Seeding The Clouds with Trust Anchors

This paper address the lack of complete transparency in cloud computing and the attestation process. Schiffman, Moyer, Vijayakumar, Jaeger and McDaniel [32] proposed a protocol for integrity-verifiable applications on a cloud system; the main componet of the protocol is am(CV) which provides attestation for the VM images which can be used to verify the integrity of both the VM and the host. This protocol improves the External Trusted Entity (ETE) described in [28]; the primary aim is to have the CSP to prove its own integrity and that all the security requirements are met to the user.

The protocols adopts the Outbound Verification (OA) model [33] which authenticates the programs running on a secure-coprocessor to a model running applications on a complex system. The authors [32] extended the verifying single entity process to multiple entities so that it suits the IaaS model.

Figure 5.2 shows in detail how the verification protocol works. It has four distinct components namely the Alice (user), Cloud Controler and Verifier(CV), Node Controller(NC) and the Cloud Service Provider (CSP). The main goal for this protocol is for the user (Alice) to be able to determine that the cloud platform satisfies the integrity criteria set and whether the VM runtime security perimeters are enforced.

Protocols Preliminaries

- E, γ represents the current entities (codes and programs) running in the layer and the policies.
- H shows the initial configuration prior to execution.
- R operations carried out during the run of the layer.
- $\mathsf{SK_i}$ the private portion of the Attestation Identity Key (AIK)of entity i.
- $\mathsf{PK_i}$ the public portion of the Attestation Identity Key (AIK)of entity i.
- $\mathsf{C_i}$ the criteria which includes the security and policies for entity i.
- $Attest(i)$ attest for entity i.

The protocol steps as shown in figure 5.2 are explained below

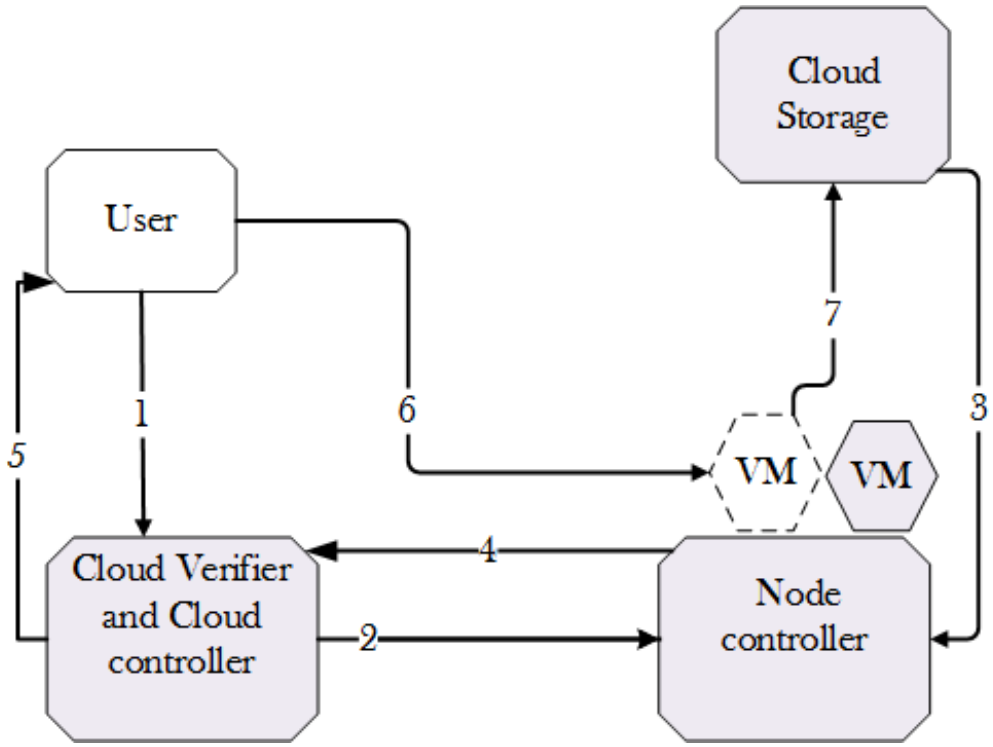

Figure 5.2: Verification Protocol

1. Alice request for attest from CV and CV responds with $\mathsf{Attest}(\mathsf{CV}) = \mathsf{sign}(\mathsf{E}, \gamma, \mathsf{H}, \mathsf{R})\mathsf{Sk}_{\mathsf{CV}}$ and C_{CV}. Alice verifies the Attest(CV) and verify that C_{CV} is compliant to C_{A} .

2. CV request for attestation to NC, and CV verifies the attestation. The attestation process is the same as the previous step, just the keys and criteria changes to suit the entities involved.

3. If NC is verified, Alice sends a request to start the VM on the CS.

4. The NC sends attestation for the VM to CV, the CV signs the keys with its private key $\mathsf{Sk}_{\mathsf{CV}}$

5. Alice verifies the VMs attestation against C_{A} VM integrity properties and validates that the VM identity keys signature chain comes from the CV. Finally, she uses the identity key to establish an authenticated connection with the VM in step (6) and sends an authorization for the VM to access application data hosted on the cloud. Without this authorization, no VM is able to access the data.

This protocol does not allow the user to have direct access to the Node Controller but through the Cloud Verifier. This is done to to provide security the the

NC. The authors implement the prototype of the protocol and evaluated it. From their evaluation, some steps can be run simultaneous which makes it favorable for the IaaS environment. However the protocol requires human intervention where the user need to verify the information from the CV.

5.1.4 Securely Launching Virtual Machines Trustworthy Platforms in a Public Cloud:An Enterprises Perspective

This paper focuses on verifying the security guarantees provided by the CSP and ensuring that the the entire process of VM launch meets all security requirements such as secure authentication and secure data transfer. The authors [34] analyzed the initial phase of the VM life cycle which is the launch phase of the VM from the security perspective and how to protect the entire launch and migration process. The paper focused on solve four major threats related to VM launch in the public cloud namely: cloud platform authenticity, insider attacks, mis-configured cloud platform and user-data leak and threats that are general to distributed systems namely: Insecure API, Provider network attacks, user client authenticity and disowning malicious VM.

The protocol uses TPM's remote attestation, sealing and key binding as the mechanism to provide secure VM launch. The protocol have three phases explained below.

1. **Connect and Discover** This is the phase of authentication where users authenticate themselves to the CSP gateway which authenticate users and establishes a secure connection. The authors used IPSec to establish VPN connection. After a secure connection is established, the user requests for host to the procurement server(PS). The procurement server function is to keep the list of all the hosts available to the user and present them. The user picks the host which matches the user criteria. The PS sends a file which consist of important information about the host such as links to the policies, service level agreements, access rights, reference measures, etc to the user. This file is TPM-signed and is used for integrity verification by the user.

2. **Platform Integrity Verification** The user validates the integrity of the platform by using attestation protocol where the user sends a nonce n_u to the platform. Once the attestation process has been successful the user compute a hash of the attestation response. This hashed value is used to bind in the VM launch process.

3. **VM Launch**

   This is the final stage of the protocol. The user data is protected by binding it to the platform using TPM, this means that only trusted hosts can access the user data. The user request for assurance that key for binding is not

migratable or it is a certified migratable TPM key, once this assurance is given the user generates a symmetric key (K_{VM}) and encrypts her VM image. The symmetric key is encrypted with the public binding key. This ensures that only hosts that are attested and trusted can decrypt the VM image. The user finally sign the VM image so that she cannot repudiate the launch at later stage. The Platform perform validations checks, decrypt the VM and finally run the VM instance. The user then can connect to the running instance. If the launch failed, a message is send to the user and the cloud platform for further actions.

This protocol provides strong security notions such as authentication, confidentiality, integrity, non-repudiation and replay protection. The protocol however does not address the VM that might not be pre-packed. Also just like the [32] requires too many intervention from the user. This can hinder it's adoption to the cloud.

5.1.5 Trusted Launch of Generic Virtual Machine Images in Public IaaS Environments

One of the drawbacks the proposed protocol by Aslam et al. [34] is that it is only applicable to pre-packaged VM and that, it is unsuitable for generic VMs. This limitation is addressed in [35], which heavily re-uses the ideas in [34] but improved the shortcomings. Generic Virtual Machine images are defined in the paper to be binary identical and not be customized to suite the user's need. Four entities are present in the protocol

- User U: Is the User of the Cloud Services that intends on launch a VM. The User can be a novel user or an expert user;
- Scheduler S: receives VMs instance requests from U and also handles to scheduling and rescheduling of available VM instances on the hosts H;
- Hosts H: Physical or virtual server that is able to host a virtual machine. The host must be TCG complaint and be equipped with TPM;
- Trusted Third Party TTP: attest the configuration of the hosts that will host the VM and assess their security profile according to pre-defined rules ;
- security profile SP: is a verified setup of an OS including underlying libraries and configuration files, which are considered to be trusted by all parties (User and CSP). SP can range on an ascending integer scale which reflects the level of verification, from least to most strict (and hence more restrictive) [35].

The protocol works as follows

1. The User U initiates the protocol by generating a nonce N which serve as a proof of token for the interaction between U and VM instance. This nonce N is kept a secret. The user U also generates a token γ which represents the users criteria which contain security information, the preferred security profile and the hash of the VM image $\mathsf{h_{vm}}$. The user then encrypts the token with the public key of TTP that is $\mathsf{Enc}(\gamma_{\mathsf{PKTTP}})$ and send it to SP;

2. SP schedule the VM on the appropriate h, sends a request for the binding key $\mathsf{PK_{bind}}$ and provides the TTP URL;

3. When the host h receives the request, it retrieves the binding key $\mathsf{PK_{bind}}$ and proves that the key is non-migratable to U;

4. The host h sends a request for attestation which includes the attest data and the $\mathsf{Enc}(\gamma_{\mathsf{PKTTP}}$ from the user to the TTP;

5. TTP validates the data it received from the host h;

6. Once everything is validated and correct; the TTP encrypts the N and the $\mathsf{h_{vm}}$ that is $\mathsf{Enc}(\mathsf{N}, \mathsf{h_{vm}})_{\mathsf{PKbind}}$) and sends it to H. The host uses the binding key stored in it's TPM to decrypt and verifies the hash of the VM image, if everything is correct, the host injects the N before launching the VM;

7. The host H sends an acknowledgment to S to confirm the launch and the U sends a challenge to the VM launched to prove it's knowledge of N.

In order to counter man in the middle attacks, the N since it is kept a secret can be used as a pre-shared key that can be used for secure communication between the U and the launched VM. This protocol assumes close box execution; however, the authors did not explain why they made such an assumption. Moreover, such assumption is not practical in the real world. It is a problem that need to be solved.

5.2 Cryptographic Secure Storage

In this section we provide a review of existing secure storage protocols which realies heavily on cryptographic techniques in order to provide security. We describe their main features and security notions. Before looking into these techniques, we look at the notion of a cryptographic secure storage as introduced by Kamara and Lauter [36]. The authors used non-standards cryptographic methods to illustrate how to achieve data security goals such as data integrity and confidentiality in a public cloud setup where the cloud provider is not trusted by the clients. Their solution was adapted on both consumers or individual and enterprise scenarios. At the core of the Cryptographic Storage, Kamara et al [36] proposed that the architecture consist of the following components:

- Data Processor (DP): This component process the data before sending it to the cloud for storage
- Data Verifier (DV): Verify the integrity of the data in the cloud to establish whether the data has been tampered with or not
- Token Generator (TG): Generates a token that is used by the cloud provider to retrieve the segment of requested data and also generates access credentials according to the access control policy. These credentials enables the parties to decrypt encrypted files.

This solution is regarded as the first contribution to cryptographic-based secure storage [37]. The authors recommended varieties of cryptographic techniques that can be used for the implementation. These are searchable encryption, attribute based encryption and proof of storage. The focus of this thesis is on searchable encryption, we will now look at the solutions that uses searchable symmetric encryption in order to implement a secure storage.

5.2.1 Searchable Symmetric Encryption

Searchable encryption is a technique which allows a client to encrypt her data in such a way that search tokens are generated that allows the server to search over the encrypted data and return the appropriate encrypted files [38, 36]. Data in this case is a collection $f = \{f_1, \ldots, f_n\}$ of n files where f_i is a sequence of words $\{w, \ldots, w_m\}$ from some keyword space W. In addition each file f_i have a unique file identifier id (f_i). The searchable encryption is dynamic if it allows addition and removal of files from the server. Therefore add/delete tokens are generated when the client wants to add/delete files. We adapt formalized the definitions of searchable symmetric encryption (SSE) by the Curtmola et al. [39] and the dynamic extension of SSE by Kamara et al. [40].

Definition 4 (Dynamic Index-based Searchable Symmetric Encryption)**.** *an index-based SSE scheme is a collection of nine polynomial algorithms* $SSE = (Gen, Enc, SearchToken, AddToken, DeleteToken, Search, Delete, Add, Dec)$ *such that:*

- *Gen is a probabilistic key generation algorithm that is used by the user to generate a secret key. It takes* λ *as a security parameter input and outputs a secret key* K*.*
- *Enc is a probabilistic algorithm that takes input* K *and a collection of files* $f = \{f_1, \ldots, f_n\}$ *and outputs a secure index* γ *and a sequence of ciphertexts* $c = \{c_1, \ldots, c_n\}$*.*
- *SearchToken is a (possibly probabilistic) algorithm that takes as input a secret key* K *and a keyword* w *and outputs a search token* $\tau_s(w)$*.*

- *AddToken is a (possibly probabilistic) algorithm that takes as input a secret key K and a file f and outputs an add token $\tau_a(f)$ and a ciphertext cf. It is used by the client to add and encrypt the new file which is sent to the sever .*

- *DeleteToken is a (possibly probabilistic) algorithm that takes as input a secret key K and a file f and outputs a delete token $\tau_d(f)$. It is used by the client to create a delete token for some file to be deleted which is then sent to the server.*

- *Search is a deterministic algorithm that takes as input an encrypted index , a sequence of ciphertexts c and a search token $\tau_s(w)$ and outputs a sequence of file identifiers I_w.This algorithm is used by the server to search over the encrypted data and determine which ciphertexts correspond to the searched keyword and which should be sent to the client.*

- *Add is a deterministic algorithm that takes as input an encrypted index γ, a sequence of ciphertexts c, an add token $\tau_a(f)$ and a ciphertext c_f and outputs a new encrypted index γ and a new sequence of ciphertexts c'. This algorithm is used by the server to update the encrypted index and the ciphertext vector to include the data corresponding to the new file.*

- *Delete is a deterministic algorithm that takes as input an encrypted index γ, a sequence of ciphertexts c and a delete token $\tau_d(f)$ and outputs a new encrypted index γ' and a new sequence of ciphertexts c'. This algorithm is used by the server upon receive of a delete token in order to update the encrypted index and the ciphertext vector to delete the data corresponding to the deleted file.*

- *Dec is a deterministic algorithm that takes as input a secret key K and a ciphertext c and outputs a file f.*

5.2.1.1 Secure Indexes per Document

The is the first practical scheme introduced by Song, Wagner and Perrig [41] for searching over encrypted data [21, 38], the scheme works by using a two layered encryption that allows searching on the ciphertexts with a sequential scan. In this construct each word is encrypted separately by using a deterministic encryption and then uses a stream cipher with a special structure to do the second encryption. The detailed steps of the scheme are described below:

1. When the client Alice A wants to create a searchable ciphertexts, the data is split into sequence of words $w_1, \ldots, c_i$ and uses a deterministic algorithm to encrypt w_i and the encrypted word that is $X_i = Enc(w_i)$. X_i is split into two parts $(X_i = L_i, R_i)$

2. A pseudo-random generator is used to generate a value S_i, a key $k_i = f_{k'}(Li)$ where $f()$ is a pseudo-random function and The S_i is hashed with $F()$ such that $Y_i = S_i, F_{ki}(S_i)$.

3. To complete the encryption process a stream cipher is used so that $C_i = X_i \oplus Y_i$

4. To search over the encrypted files, a trapdoor which contain encrypted keywords is required. That is $X = E(w_i) = (L, R)$ and $k = f_{k'}(L)$ so that the server can perform search by Xoring $C_i \oplus X = (s, F_k(S))$ and if that is found then the keyword was found.

Although this scheme does not use any formal security definition for searchable encryption, it is IND-CPA secure given that the pseudo-random functions are proven to be secured. It however leaks the position of the possible match. This makes the scheme vulnerable to statistical analysis attacks. Finally the complexity of the encryption and search algorithm is linear to the number of words.

5.2.1.2 Forward Index Approach

This approach introduced by Goh [42] have stronger security guarantees and address the limitations of the scheme by Song et.al [41] which uses fixed size words and special document encryption. The idea is to have an index for each document which is independent of the underlying encryption scheme, these indexes are build using Bloom filter [43]. The Bloom filter is a data structure that represent a set $S = \{s_1, ..., s_n\}$ of n elements as an array of m bits that are initially set to 0. Generally the filter uses r independent hash functions $\{h_1, ..., h_p\}$, where $h_i : \{0,1\}^* \rightarrow [i, m]$ for $\{i\epsilon[1, r]$, for each maps a set of element to one of the m array. For each element fx to be added in a set $S = \{x_1, ..., x_m\}$ the bits at position $\{h_1(x_i), ..., h_r(x_i)\}$ are set to 1. In order to check whether the element e in this data structure is contained in the element or not, check if the bits at positions $\{h_1(e), ..., h_m(e)\}$ are set to 1 if that is true then e is a member of set S otherwise e is not a member of S. The scheme consist of the following algorithms:

- $\mathsf{Gen}(\lambda)$: is a probabilistic algorithm that takes security parameter λ as input and outputs a secret key K . such that a pseudo-random function

- $\mathsf{Trapdoor}(\mathsf{K}, \mathsf{w})$ Is deterministic algorithm which takes in two inputs the secret key K, and the word w to output the trapdoor for w T_w.

- $\mathsf{BuildIndex}(\mathsf{D}, \mathsf{K})$: Is deterministic algorithm which outputs the index γ given a document D and a secret key K.

- $\mathsf{SearchIndex}(\mathsf{Tw}, \gamma)$:Is deterministic algorithm, given the trapdoor T_w for word w and the index γ for document D which outputs 1 for a match otherwise 0.

Bloom filters' inherent problem is that they produce false positives however these false positives can be minimized by adjusting appropriate parameters settings. In order to avoid leakage,the scheme process each distinct document in a pseudo-random function twice. The second run, takes in the first run results as an input into the pseudo-random function and add a unique document identifier to make all the Bloom Filters look different even for the documents with the same keywords. The search time of this scheme is linear to the number of documents. In terms of security, this scheme is **IND1-CKA** secure however the security definitions do not guarantee the security of the trapdoors therefore this scheme leaks some information to the server.

Chang and Mitzenmacher [44] proposed a scheme similar to Goh's [42] scheme but without false positives. The idea is to use pre-built dictionary of search keywords to build an index for each document. The scheme assumes two constructions where (1) the user u is at home and wishes to store her files on the server S and a dictionary can be stored on the user mobile device, (2) u does not have sufficient storage on her mobile device, therefore u is a mobile user and wants to retrieve encrypted files from the S using keywords. In both constructions, pseudo-random permutations and pseudo-random functions are used. The scheme basically works as follows.

- The user u chooses random two secret keys s, r and prepares an index γ_i for each file f_i such that if f_i contains a keyword $\{w_j$, u sets the index $\gamma_i[P_s(j)]$ to 1 otherwise it is set to 0 where $F_k(x)$ is a pseudo-random permutations function.
- For each file u computes a masked index string M_i such that $M_i[j] = \gamma_i[j] \oplus G_{rj}(i)$ where $G_k(x)$ is a mapping pseudo-random function
- u submits $Enc(f_i)$ and a corresponding masked index string M_i
- u keeps the secret keys s, r and the dictionary on her mobile device
- To retrieve the files, u retrieves the corresponding index λ and sends the two values $p = P_s(\lambda)$ and $f = F_r(p)$ to S.
- S computes if $\gamma_i[p] = M_i[p] \oplus G_f(i)$. The $Enc(f_i)$ is returned to u if $\gamma_i[p] = 1$

This scheme associate each masked keyword index to each file. In addition the index is linear to the number of distinct words per file therefore the search time is proportional to the total number of files.

5.2.1.3 Secure Indexes per Keyword

The schemes we have seen above where all based on secure indexes per document, this indexes are based on forward index approach. Curtmola, Garay, Kamara, and Ostrovskyl [39] proposed two constructions which reduces the search time to the number of files that contains the keyword. The idea is to use an inverted index per distinct keyword instead of per distinct document which makes the scheme sublinear and optimal. The first construction we refer to it as SSE-1 works as follows; For each distinct keyword w, a linked list l_w which contain the document identifier for the document that contain w is created. Each node consists of three fields (1) document identifier of the document that contains the keyword w (2) the key that is used to encrypt the next node (3) the pointer to the next node. The nodes of all linked lists are encrypted with random keys and then stored in an array A in a random order. A table T is used to locate and decrypt the first node of each l_w. Each keyword w have a corresponding entry is T with two values (1) address that is used to locate the entry in T (2) value which is encrypted using a pseudo-random function and contain the location in array A and the decryption key for the first node of l_w. When a user u wants to retrieve documents from the server S , u prepares the decryption key and sends it to the S together with the T address associated with the keyword w. The server uses this information to locate and decrypt the entry of T, gets the pointer to l_w and its decryption key. The server is then able to decrypt all relevant nodes in order to obtain the document identifiers. SSE-1 is $IND - CKA1$ secure and because the index is linear in the number of distinct word per document it is optimal.

The second construction is SSE-2 which is secure against adaptive attacks but requires more storage. SSE-2 uses a look up table T with extended labels, using the same addressing as SSE-1. In SSE-2 for each w appearing in n documents, the extended labels is made out of concatenating w with n so that it looks like this $w|1, ..., w|n$. Each of the label is associated with a pseudo-random entry of T containing the document's identifiers. To hide the number of distinct keyword for each document T is padded so that all document identifiers have the same number of entries. To search for the document containing w all labels are search, but since each label is unique, a search reveals a single document identifier.

Both these constructions proved to be secure against non-adaptive (CKA1) and adaptive searches(CKA2) respectively. Additionally, the SSE-2 is not explicitly dynamic and it requires higher communications cost and storage on the server. We will look at two more schemes that address this limitations.

5.2.1.4 Dynamic SSE-1

The SSE-1 in [39] was improved by [40] in order to overcome it's two limitations namely (1) it is only IND1-CKA secure and (2) it is not explicitly dynamic. The first limitation is straightforward to overcome since it can be done by making the scheme non-committing. The second limitation is however complicated and

difficult to solve because the addition, deletion or modification of a file requires the server to add, delete and modify the encrypted nodes of the l_w stored in the search array A. This means that firstly, when the file f is deleted, the nodes in A corresponding to f should be cleared but the server does not know where the nodes are stored; secondly when a node of a list is inserted or deleted, the server cannot modify the pointer of the previous node because its encrypted and lastly when a node is added, the server does not know which locations in A are free. To address the above difficulties [40] used the following techniques

1. *File Deletion:* Added a encrypted deletion array A_d which stores for each file fa list of nodes that point to search array positions that need to be modified if a file is deleted.

2. *Pointer Modification:* The pointers stored in the node are encrypted with a homomorphic encryption, which enables the server to modify the pointer without having to decrypt it. The encryption is done by simply XORing the message with the output of a PRF. This construction is non-committing.

3. *Memory Management:* Keeps a list of free nodes which keep tracks of free positions in the search Array.

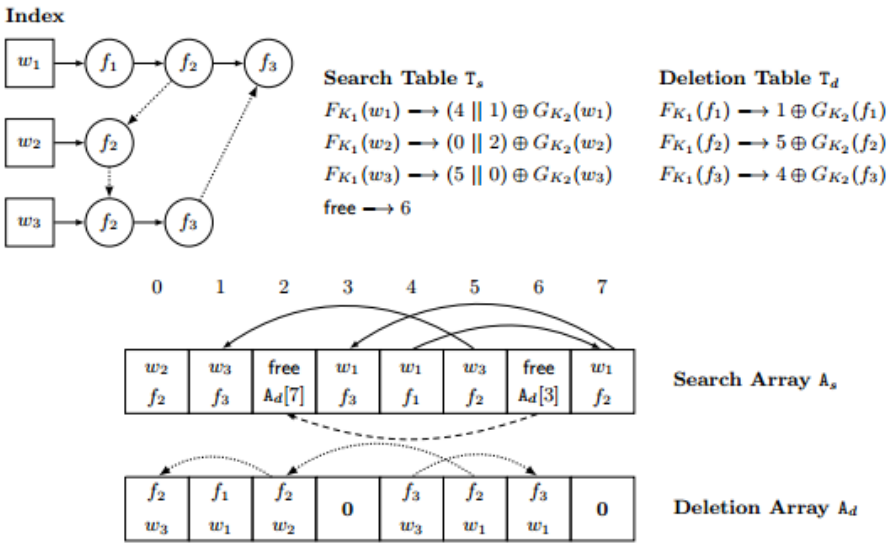

Figure 5.3: Fully Dynamic SSE-1

The fully dynamic SSE-1 illustration is shown in 5.3 [40]. The index consist of three files f_1,f_2,f_3, three words w_1, w_2, w_3, a search table T_s, deletion table T_d, search Array A_s and the deletion Array A_d.

Searching: If a user wants to search for all files that contains w_1, a search token which contains FK_1w_1 and GK_2w_1 where $F_K()$ and $G_K()$ are pseudo-random functions and K is a random generated secret key. The first value FK_1w_1 is used to locate w_1's entry in T_s and GK_2w_1 is used to compute a value that will is used by the server to locate the pointer to the files containing w_1

Adding a file: When a user wishes to add a file f_4 containing w_1 and w_2, T_s remains the same. However the server looks for free entries positions in A_s in this case are position 2 and 6; and stores (w_1, f_4) and (w_2, f_4) because A_s and A_d are updated, the A_d is also updated with (w_1, f_4) and (w_2, f_4) at position 3 and 7 which were free and the server updates the pointers. Finally the T_d is updated, setting $F_{K1}(f_4)$ to point to position 3.

Deleting a File: When a user wants to delete a file f_3 containing w_1 and w_3, the server use $F_{K1}(f_3)$ from the deletion token to locate the right value in T_d which is used to locate the positions in the A_s and A_s in order to free positions 1 and 3; and 4 and 6 respectively.

In the static SSE schemes, searched keywords can be chosen based on the encrypted index and the results of the previous queries. A simulator is then required to create an encrypted index which is equivocable by for example creating fake encrypted index and when a keyword is search, the simulator creates an appropriate search token. This is achieved by using non commiting encryption schemes [39]. In a dynamic scheme however a higher equivocation is required because an adversary can initiate the search w inorder to get the simulator to commit a search token. The adversary then uploads a file with w and query w again. Because the simulator cannot modify the encrypted index, it is unable to update the index to reflect the changes. The appropriate level of equivocation in this scheme [40] is achieved by a random oracle which handles the adversary queries first in order to provide the required level of equivocation to the simulator.

This scheme provides strong security notions, although the add/delete leakage function leaks too much information about the search tokens corresponding to the keyword in the added or deleted file. The scheme however is proven secure in a random oracle.

5.2.1.5 Parallel and dynamic SSE

The construction by Kamara et al. [40] provided strong security notion but lacked parallelism. To address this problem Kamara and Papamanthou [45] uses multicore architecture to proposes a highly parallelizable and dynamic SSE scheme which achieve a sublinear search time [39]. This scheme is based on a Red-Black tree-based multi-map data structure and is referred to as Key Red Black tree (KRB).

The assumption is that the universe of keywords is fixed and the total number of the keywords (m) is smaller than the number of files (n) can grow dynamically.

Given a set of files $F = (f_{i1}..., f_{in})$, a universe of keywords w. A total order on the files $F = (f_{i1}..., f_{in})$ imposed by ordering the identifiers $i = (i_1..., i_n)$. The pointers to the appropriate documents are stored at the leaves of the tree. Each internal node u of the tree stores an m-bit vector $d_u = (d_{u1}, ..., d_{ui})$ where d_{ui} corresponds to the i-th keyword w (w_i in the keyword universe so that the bit d_{ui} is set to 1 only if there is atleast one one file associated with u's children that contains the keyword w_i.

To search for a keyword w, assuming w is at position i in the m-bit node, traverse from the root until the bit at the i-th position of node u's children is equal to 1 which means that there is a file associated to w and return all the leaves that has been traversed. The properties of KRB tree that makes it suitable for this construction is it allows both keyword-based operations (by following paths from the root to the leaves) which are used for searching and file-based operations (by following paths from the leaves to the root) which are useful for handling updates efficiently [45]. Another useful property is that the search in each children can continue using a different processor [38].

To encrypt the data structure; for each keyword w_i there is a unique key that is used to encrypt the bits $d_{u,i}$ (for all u). The encrypted bit $d_{u,i}$ is then stored at one of two hash tables associated with node u, at a pseudo-random position. The random oracle output determines which hash table the $d_{u,i}$ will be stored. The other table will contain a random value in the respective position. To update, the server performs a structure update on the KRB tree which involves the necessary rotations that are performed during an update of a red-black tree (in order to maintain a a logarithmic height). This operation requires only the file identifier. The server then sends to the client the part of the tree that needs to be updated, and the clients answers with a token that allows the server to update the values at those position [38].

The scheme is proved secure in a random oracle model. This scheme does not leak information during an update unlike the previous schemes [40] and it can perform efficient updates since all files f can be done in $O(Log|f|)$ time although it takes 1.5 rounds. The drawback of this scheme is that the data structure has size $O(m.|f|)$ and the constants are very high [38, 21].

5.2.1.6 Blind storage

Blind storage by Naveed, Prabhakaran and Gunter [46]is an SSE scheme that is based on a basic primitive called *Blind Storage*. The idea of the blind storage is that a client can store files on a server in such a way that a server does not learn any information about the file such as the number of files stored or the lengths of each file. The server however learns about the files existence when it is retrieved but it does not learn anything about the file content or file name. Blind storage also supports dynamic SSE operations such as adding new files, deleting files and updating files but such operations are hidden from the server.

The Blind Storage is build by storing each file as a collection of blocks that are

stored at pseudo-random locations. Given a file f with n blocks, αn locations are a set of $1, ..., N$ are chosen using a pseudo-random number generator and n blocks of f are stored in the n position where N is the upper bound of the number of data blocks that can be stored. In order to avoid collisions with other files storage position, many α blocks are chosen for a f. The αn positions retrieved from the server to access file f are independent of other files and f is stored encrypted. In order to retrieve f from the server, the client need to know the amount of blocks in f and this can be achieved by keeping this information at the client side. If the client have relatively many files, the alternative is to store this information in the first block and have an additional round of alternations for the client to retrieve the i first block of f.

The SSE can be build on top of the Blind Storage by storing index file for each keyword in the blind storage system. This will guarantee security notions of a **static SSE** since the server learns nothing about the files except the patterns of keywords accessed by the user. To achieve dynamic SSE; the scheme uses two different indexes for the original files and the added files. The index corresponding to the original files is done using the blind storage scheme and *lazy deletion strategy* in which the index file of a keyword is not updated until it is searched for. The index corresponding to the added files is done using a much simpler scheme which support efficient updates.

This scheme provides transparency because the computations are not done by the server but the server provides the interface for uploading/downloading files. The server also cannot perform decryption on files which provides security of files against malicious cloud providers. However the updates leak a deterministic function of the keywords and so the security guarantees for the added files are much weaker than for the original files [38].

The authors [46] implemented the scheme and found it to be more efficient, scalable and practical than prior the scheme by [45]. However, the scheme was implemented in [38], but the authors found it to be unpractical for the cloud.

Chapter 6

Threats and Attacks in Cloud Computing

The security threats and associated attacks in cloud computing raised several security concerns. The nature of the architecture of cloud computing such as multi-tenancy and data storage on a public cloud; brings in addition to traditional IT systems security threats new level of threats and vulnerabilities. A threat is any event that may be malicious or incidental, which compromises the cloud resources; an attack is an action to harm cloud resources [6]. In Chapter 5 we looked at the protocols that are designed to solve the problems identified in Chapter 4. This Chapter discusses the threats and attacks specific to the *IaaS*environment; and the encryption techniques used for Secure Storage.

6.1 Security Issues in Virtualization

IaaS providers such as Amazon AWS, Windows Azure and Google Compute Engine deploy virtual machines on top of the virtual machine management (VMM) to provide infrustructure as a service to their clients. Although multi-tenant model brings economic benefits to both the clients and the cloud service provider; it also brings risks of co-location attacks [47]. That exposes virtual machines to threats by other tenants, malicious system administrators and external attacks. The threats and possible attacks that exploits this threats are discussion in the following sections.

6.1.1 Threats

The Center of internet security [48] published security threats in the virtualization based on the nature of virtualization technology. These threats exposes the IaaS to attacks especially from the malicious insiders. The threats are discussed below

1. **Virtual Machine Monitoring:** The cloud service provider have the responsibility to perform tasks which controls the clients' VM and also the

allocation of VM resources. Although the cloud provider and clients signs a service level agreement that entails among others security guarantees such as confidentiality; a malicious *sysadmin* with root access can misuse his privileges to access the clients's VM and hence see their data. For example Xenacces [49] is a tool that gives a privileged user access to run a user level process in a privileged domain of XEN to in order to access the clients' VM memory [50]. In cases like this the *sysadmin* can abuse his power to compromise the clients' data.

2. **Virtual Machine Migration:** One of the advantages of virtualization is its ability to launch the VM on any available host. This technology enables the VM to be easily migratable offline or live. Although this is a benefit, it comes to a cost because it means that the VM file can be illegally copied by the malicious *sysadmin* either to an external storage or to another server. This causes threats to the clients' data security because data can be modified, read or destroyed.

3. **Virtual Machine Communication with Hosts or other VMs:** Virtualization allow resource sharing, such as its ability to host several operating systems on one hardware platform. Although there is a level of isolation between virtual machines, the hosts can still see the packets exchanged between the hosts and the VMs because all the network traffic go through the host, this can lead to packet sniffing and ARP poisoning. In addition, technologies such as shared clipboards which allows data exchange between VMs and hosts can be exploited by for example copying a malicious program to the VM from the host by a malicious *sysadmin*.

6.1.2 Threat Model

We share the threat model in [28, 51] which is based on Dolev- Yao adversary model [52]. In this model, the assumption is that the adversary is a malicious cloud service provider employee, a system administrator *sysadmin* who have root privileges that he can use to object remote access to any compute host maintained by the IaaS and exclusively wants to compromise the confidentiality of the IaaS. The following assumptions are made:

- Physical Security: The cloud service provider has applied maximum security to the data center so that physical security can be enforced and audited in accordance to the best practices and compliance to the industry standards.

- Low-Level Software Stack: integrity measurements of the low-level software stack: the Core Root of Trust for measurement; BIOS and host extensions; host platform configuration; Option ROM code, configuration and data; Initial Platform Loader code and configuration; state transitions and wake events, and a minimal hypervisor.

- Network Infrastructure: The cloud service provider has a control of the network. Therefore the *sysadmin* has full control of the network configuration and can overhear, create,replay and destroy all messages communicated between DM and their resources (VMs, virtual routers, storage abstraction components) and may use all he can to learn confidential information.
- Cryptographic Security: The encryption schemes are semantically secure and plain texts cannot be obtained from ciphertexts without decryption keys. Signature schemes and MAC algorithms are assumed to be unforgeable and verify the integrity correctly.

6.1.3 Attacks

6.1.3.1 Passwords in Memory Snapshots

This attack is based on the findings by [53] in which the author has shown that it is possible to extract clear text passwords from a Linx memory dump. Rocha et all. [54] demonstrated that it is possible for a system administrator with privileges access to obtain the a memory dump of a target VM using a dump-core comand in Xen management user interface and extract clear text passwords from the memory dump. Futhermore [55] used the same method in Cloudstack to extract plain text passwords. The attacker can use tools such as TrueCrypt [1] or sophisticated tools to automate the search for passwords in the memory dump.

6.1.3.2 Obtaining Private Keys Using Memory Snapshots

A pair of Public/Private keys can be obtained from the memory dump inorder for the attacker to impersonate the server. In this attack, the system administrator obtains the memory dump and use techniques used in the cold boot attack [31] to obtain the keys. The authors in [54] used $rsakeyfind$ (part of a package with the same name available for several Linux distributions) that can search and extract RSA keys from the memory dump. Although this attack is similar to the previous one; it is more devastating because the adversary can break all cryptographic protocols if he gets the Public/Private keys pairs.

6.1.3.3 Host Substitution Attack

In this attack, the malicious *sysadmin* have root access rights, he can set up a compromised host which is in his control. Although this is complex and requires analysis and modification of data [51], it can still de done. This attack is successful if the malicious *sysadmin* manages to divert the VM image and launch it on a compromised host. This way, the attacker will have a complete control of the client virtual machine. A practical example of this attack was demonstrated by [54] and [55]. The figure 6.1 presents the attack below.

[1] http://truecrypt.sourceforge.net

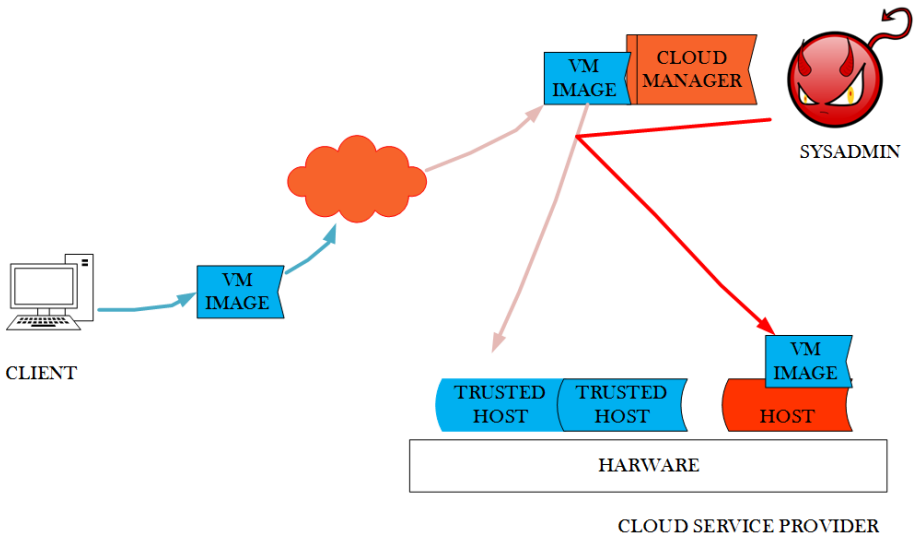

Figure 6.1: Host Substitution Attack

6.1.3.4 Virtual Machine Image Attack

A malicious *sysadmin* can compromise the VM image before it is launched on the compute host. This will can enable him to install malicious code that will leak the users information to a remote application. This can give the *sysadmin* complete control over the data of the user. A practical example of this attack was demonstrated in [55], they called it Snapshot Cracking,and was implemented in CloudStack [1]. What they did was to take a Snapshot of a VM and copied to another server. They then used a tool called System Rescue which they used to change the VM root password by modifying the shadow file. The figure below demonstrate this attack.

6.2 Data Storage Security

In the previous section, we have seen that once the system administrator successfully takes control of the users' VM, he gets control of data as well. Cloud providers protect the data from unauthorized access by encrypting the data at rest. This protects the data from unauthorized access by external attacks but does not protect the users' data from the cloud provider itself. Given that the cloud provider is not fully trusted by the users, the idea is to encrypt the data before uploading it to the *CSP*, this means that the users should be able to search over the encrypted data in order to retrieve specific file. Several cryptographic protocols has been developed to enable users to search over encrypted data, however this protocols are subjected to attacks which are discussed below.

[1] https://cloudstack.apache.org/

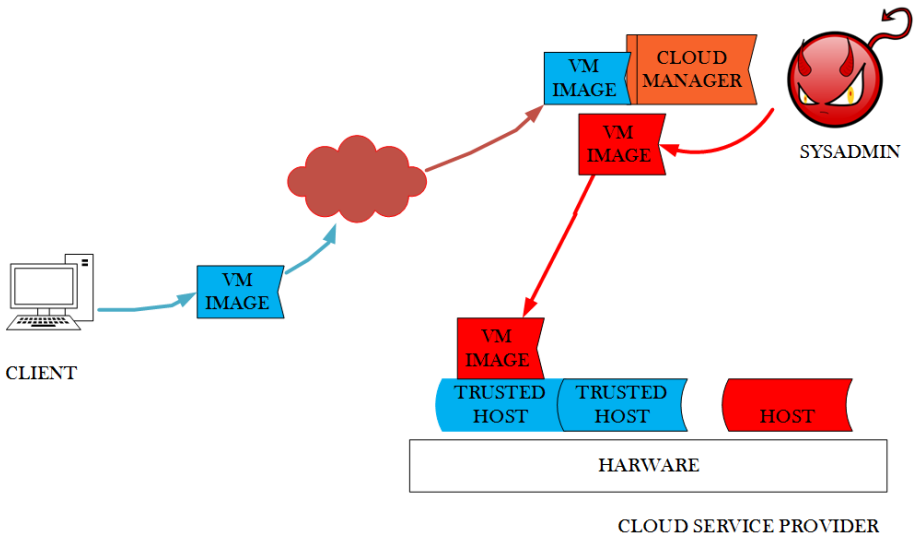

Figure 6.2: VM Image Attack

6.2.1 Threat Model

In addition to the threat model above,the assumption is that the CSP cannot be trusted,therefore even if the users' files are encrypted, the server can try to learn information about the files from the user's requests. Therefore the adversary have access to both the encryption and decryption oracle. The oracles retains the secret key and can encrypt/decrypt the files at the adversary request. The attack models are classified into two categories [56].

- An **honest-but-curious** [57, 58, 59] server follows the protocol and takes no actions beyond those of an honest server, and attempts to learn about the plaintext of documents or terms that were queried.
- An **active adversary** [60, 61, 62, 63] can carry out a chosen-document attack in which it tricks the client into including a chosen document in the document set.

6.2.2 Attacks

6.2.2.1 Attacks on Chosen Plaintext/Ciphertext

This definitions are based on Alex [64]

1. Adaptive Chosen Plaintext Attack (CPA): In this attack, the adversary choses an arbitrary number of plain texts to be encrypted and obtain the corresponding ciphertexts. The adversary strategy is to try and derive partial information by querying the encryption oracle based on the information gained from the preceding encryption. This is an important model in the

context of public key cryptography where the encryption key is public and adversary can encrypt any chosen plaintexts.

2. Chosen Ciphertext Attack(CCA): The adversary chooses an arbitrary number of ciphertexts to obtain the corresponding plaintexts.

3. Adaptive Chosen Ciphertext Attack(CCA2):The adversary tries to gain partial information by making queries to the decryption oracle based on the results of previous decryption

4. Indistinguishability under Adaptive Chosen Plaintext Attack (IND-CPA): A scheme is secured against INDA-CPA, if an adversary,which can query the encryption oracle a reasonable number of times, chooses two plaintexts for the challenger to encrypt and has only a negligible advantage over random guessing in distinguishing which plaintext belongs to which ciphertext

5. Indistinguishability under Adaptive Chosen Ciphertext Attack (IND-CCA2): A scheme is secured against INDA-CPA, if an adversary, which can queries the decryption oracle a reasonable number of times, chooses two plaintexts for the challenger to encrypt and has only a negligible advantage over random guessing in distinguishing which ciphertext belongs to which plaintext.

6.2.2.2 IKK Attack

The goal of this attacks is for the adversary to recover the plaintext keywords that corresponds to the search tokens send to the server. Therefore, this attack only succeed if the scheme leaks some sort of information to the server such as data access pattern or file access pattern. This attack was demonstrated by Islam, Kuzu and Kantarcioglu [65]. It assumes the honest but curious threat model in which the adversary follows the protocols and have full access to the communication channel. For the attack to be successfull; the protocol is assumed to atleast have a user send the trapdoor of the keywords that is T_w and the server returns the encrypted files $D = \{D_1, \ldots, D_n\}$ that contains the keyword w.By overhearing the communication, and some background information; the adversary A can deduce the keyword w in the T_w that contain in each document $D = \{D_1, \ldots, D_n\}$. By knowing the number n of possible keywords, the Adversay constructs the files words co-occurance probability in a $n \times n$ matrix N which can be simulated using frequency analysis techniques. The adversary observes the files access pattern which is reveals by the client. Given s is the number of unique search queries, s is used to construct a $s\times$ concurrence matrix S and it is normalized version S'. Then a best match of S' to N makes up a set of guesses of every query. This attack is independent of the number of queries used however it is success rate decline as the number of n keywords increases therefore its accuracy of the low that it is not usable in practice (unless the adversary already knows most of the underlying data) [56].

6.2.2.3 Query Recovery with a Counting Attack

In this attack by Cash, Grubbs, Perry and Ristenpart [56] the server knows the keywords co-occurrence matrix $n \times n$ and for each keyword w the number of files w occurring $N(w)$. When a keyword search S_w is queried with trapdoor T_w, it returns unique results and in return the server learns the number of files that contain w such that $N(T_w) = N(F_w)$. In order to recover the queries that do not have unique results, the compares the query results to the co-occurrence counts. This attack can be successful with or without the server knowing partial information.

6.2.2.4 File Injection Attack

This attacks by Zhang, Katz and Papamanthou [66] exploites the leakage functions of the SSE schemes using file injection. The first attack is what they called as a basic *binary-search attack* which solely rely on the file access pattern learned by the server and does not require any prior knowledge by the server. To execute this attack, the server injects a file that is containing half of the keywords from the universe W, then the server observes the response to the search token sent by the client. The results returned after the search leaks information to the server about the keywords corresponding to the search token. If the search token returns one of the injected file, then the server learns the exact keyword associated to the search token. This attack recovers all keywords without prior knowledge of the client's files. This attack limitation is that the injected files contains a large number of keywords, therefore by limiting the number of keywords per file decreases the effectiveness of the attack. The second attack is *Hierachical-Search Attack* which overcomes the limitations of the first attack by partitioning the keywords into subsets, the server then injects keywords in each subset inorder. The server then perform the binary search attack on the subsets to learn the exact keyword. The third attack uses the partial knowledge leaked to the server, it uses joint frequencies of the search token and the keywords. The server then performs a binary search attack to recover the exact keywords. This attack to recover the single keywords and multiple keywords. The figure below presents the file injection attack [66].

6.2.2.5 General and Adaptive attacks

Liu, Zhu Wang and Tan [67] presented attacks that exploits the search patterns. The first attack which the authors referred to as the *General attack* requires the adversary to have knowledge of the user search habits. The adversary create a frequency vector V_w of the known keywords. From the leaked search pattern, a frequencies vector V_s of the search queries over a period of time is recorded. To execute this attack, the adversary simply runs a function that determines the similarities in V_w and V_s. This attack is extended to be an *Adaptive attack* by

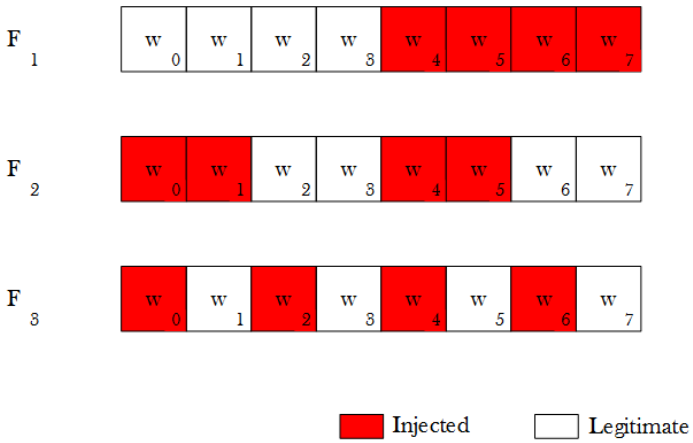

Figure 6.3: File injection attack

collecting keywords based on the results of the previous rounds of attack.The more rounds the attack is carried out, the more accurate it becomes.

Chapter 7

Analysis

This chapter presents vulnerability analysis of the protocols presented in chapter 5 against the attacks presented in chapter 6. The chapter is divided into three sections; the first section deals with the SE schemes for secure storage, the second section presents the analysis of the trusted launch protocols and thirdly we present the benefits of secure storage and trusted launch.

7.1 Searchable Encryption Schemes

The goal of searchable encryption schemes is to enable users to search over encrypted data that is stored on an untrusted server. This should be done in order to guarantee the users' data confidentiality and privacy; this means that the schemes should not leak information that may lead to the exploitation of the data security. The notable SE scheme is **oblivious RAM** by Goldreich et al. [68], this scheme can perform any type of search on the encrypted data without leaking any type of information including the data access pattern or search pattern to the server and therefore providing strong privacy guarantee. The scheme however is computationally expensive and inefficient, requiring logarithmic (in the number of documents) number of rounds of interaction for each read and write [39, 69] which make it unsuitable for the cloud environment. Making SE efficient comes with its own trade-offs, the issue is to what extend these trade-offs affects the practicality of the scheme in the real world scenario. In the following sections; we will look at the security, efficiency and privacy guarantees of each of the discussed SSE schemes in chapter 5.

7.1.1 Security

Secure Indexes per Document by Song at al. [41] is the first explicit contribution to searchable encryption. It is proven secure in a weak security model, but although it is a *indistinguishable against chosen plain text attacks*(**IND-CPA**)secure encryption scheme, it is not a secure searchable symmetric scheme [39]

because **(IND-CPA)** does not take in account the leakage function of the trapdoor. The scheme extracts keywords from the document and encrypt each keyword using a deterministic cipher. The nature of this scheme allows the server to learn the pattern location in the text where each word occurs and the total number of the word's occurrence in a document. After all keywords has been searched for, the server learns the pattern location in the text where each word occurs and the total number of the word's occurrence in a document [56]. This construction is non-interactive hence impractical in the real world application, furthermore it is vulnerable to passive statistical analysis attacks such as IKK attacks [65] and query recovery attacks [56].

Secure Indexes by Goh [42] Provided the first formal security for SSE which is *indistinguishable against chosen keyword attacks* (IND1-CKA) against the indexes. The scheme gives guarantee that (1) the adversary overhearing the communication is unable to get any information about the files content (except the file size and length) from the encrypted index γ and the ciphertexts c and (2) the encrypted index γ and a token τ_w reveal at most the outcome of the search I_w. This was improved by [44] who gave a new definition *indistinguishable against chosen keyword attacks* **(IND2-CKA)** which secured the trapdoor but was found insecure by [39]. Since this scheme leaks the search outcome, an adversary listening to the communication can make an association of the files (γ,τ_w) and this makes the scheme vulnerable to IKK attack, query recovery attacks on known/unknown documents and file injection attacks. This attacks exposes the query privacy because the adversary can know files that are associated to a certain keyword.

SSE-1 and SSE-2 [39] Is proven secure against non-adaptive **(IND-CKA1)** and adaptive searches **(IND-CKA2**) in a random oracle. However this scheme leaks the search history which consist of the keyword and the file identifiers. Overall the sever learns the search and access patterns; in addition to the file size. The authors suggested that padding can be used on the scheme to hide the file size. Given that the adversary learns the access this scheme is vulnerable to IKK attack, query recovery attacks and file injection attacks. However we note that the two attacks (IKK and query recovery) becomes less effective with padding and therefore making it impractical to carry out in the real word. With that said, [66] has shown that padding have very little effect on the Binary/Heuristic search attacks. SSE-2 leaks more information about the updated files, which makes it more vulnerable to these attacks than SSE-1.

Dynamic SSE-1 [40] This is an improvement to the SSE-1 in [39] to making it dynamic. Because of its dynamism, this scheme leaks more information that the static SSE-1. Although it is **IND-CKA2** secure and proven secure in a random oracle, the authors admitted that their scheme is not secure against IKK

attacks [65]. Because the scheme leaks data access pattern, it is vulnerable to File injection attacks [66] and query recovery attacks.

Dynamic and Parallel [45] This is a **IND-CKA2** secured highly parallel SSE design that can easily handles updates without leaking any information about the keywords that are in a added or deleted document. This scheme provides stronger security than the Dynamic SSE-1 [40] because it does not leak information during the update. The danger of update leakage in this scheme is that the server can learn that the added file contain the same keywords as the searched files and therefore the server can create some sort of file access pattern and co-occurrence matrix. Although this construction provides a stronger security notion, it still leaks the search pattern which makes the scheme vulnerable to the IKK attacks [65], query recovery attacks and File injection attacks [66].

Blind Storage [46] Is an adaptive **IND-CKA2** in a standard model. This design hides information about the files from the server until the files are accessed. The server learns information such as the length of the files and the number of files stored. This scheme also leaks the access pattern to the server. Because of this leakage, this scheme is vulnerable to IKK attacks [65], query recovery attacks and File injection attacks [66]. Finally the security of the files added is weaker than the original files which bring security concerns for large databases [38].

7.1.2 Privacy

The SSE schemes studied in this thesis aimed at removing the inefficiency in the Oblivious RAM in order to be practical for the implementation in the cloud, however each of these scheme's efficiency comes with privacy trade-off because they leak information to the server. One of the big question is to which extend a leakage is acceptable. We have seen that the SSE schemes discussed above leak either the query pattern, file access pattern or both; this is because hiding this information is expensive. Cash et al. [56] noted that the consequences of the leakage in the real world remains an open issue. The reason for this leakage is mainly because the search algorithms are deterministic, therefore the same keyword always generate the same search token. However the server can still learn the search pattern even if the algorithm is probabilistic, therefore randomizing the search algorithm will not hide the access pattern from the inside adversary such as a malicious system administrator [67].

This far, there has been four attacks that exploited the leaked information in order to learn sensitive information about the users files. According to Islam et al. [65],queries can be learn by the server if it knows the almost all the contents of the files and the keywords. Further more, [56] improved the attacks by reduce the amount of prior knowledge known to the server while the attack by Liu et al. [67] works only after the user has issued a large number of queries given that

the server knows the keywords distribution. Finally the latest attack in [66] has shown how devastating access pattern leakage can be to the query privacy in the SSE schemes. If the server learns the underlying keywords;this will enable the server to learn what type of information the user is storing. For example if the recovered keyword include "Profit", "Interest" or "Investment" the server can guess that the files are financial related.

Stefanov, Charalampos and Elaine [70] proposed a forward privacy SSE with minimum leakage based non-trivial ORAM-related techniques. Although the scheme hides the data access pattern, it is inefficient and induces a large bandwidth overhead on updates, despite supporting efficient deletions. So far hiding access pattern is has proven extremely important in encrypted keyword search and therefore is a necessary characteristics of a secure encrypted search scheme but still computational expensive and inefficient.

7.1.3 Efficiency

Efficiency is one of the important essential characteristics of cloud computing, therefore if an SSE scheme has to be deployed in the real world scenario; it must be computationally efficient. One of the efficiency parameter is the search time complexity. Schemes in [41, 42, 44] have a search time which is linear to the total number of words per document, linear to the number of documents and linear to the number of distinct words per document respectively. Both these schemes have a search time complexity which is impractical in most scenarios. Both SSE-1 and SSE-2 by Curtmola et al. [39] used inverted indexes in the schemes that provided not only sublinear but optimal search time, in which the search is limited to the number of documents that contains a certain keyword. The updates for SSE-2 is however expensive hence this scheme is more suitable for static databases.

Schemes that can make search in parallel [45] are particularly capable of deploying in the cloud environment. These schemes however need to be designed in such a way that it optimizes the I/O performance by improving the locality of the search data structure [38] Furthermore the size of the data structure that need to be stored at both the client and server side need to be optimal. Depending on the scenario, it may not be practical to require the client to store a larger data structure in a set up where the client uses a mobile phone to access the services but might be practical for different set of users. Finally the number of rounds required for server-client interaction should be kept at minimal to minimize network delays [38].

7.2 Trusted Launch Mechanisms

The properties of the Virtual Machine layer such as simplicity and access control through isolation makes it more secured than any other Operating System. The benefits of virtualizations however comes with different security issues for

example; if the isolation principle is violated, one virtual machine can be used to attack or spread malicious programs to other virtual machines. The Cloud Service Providers however works hard to ensure security that can prevent external attacks or tenant-to-tenant attacks. While this is commendable, it still remain an issue on how customers data can be protected against malicious cloud provider employees. We take note that the CSP have audit logs that can provide evidence of what the employees have done on the systems but this does not protect the customers data confidentiality.

The goal is therefore to provide cloud users with solutions that can protect their data from any form of data breach from both inside and external attacks. Due to the service model of *IaaS*, the users lose physical control of the data and there is a level of abstraction. Given the public cloud nature, there should be some sort of assurance to the users about the security and privacy their data. This section discuss the extend to which mechanism in 5 provides data security against the threats and attacks in 6.

7.2.1 TCCP

Santos et al. [28] improved the idea of a close box execution in Terra [20] to suit the *IaaS* model.The set up in the *IaaS* is that there is a privileged machine (domain controller) that have access to all VMs running under it. In Xen for example, $DOM0$ can run in a privileged mode such that it can control of all virtual machines running under it. We have seen that this control can be misused by the malicious *Sysadm* running the privileged domain controller such as $DOM0$ in Xen. This privilege can be used to access the VMs memory therefore for enabling the *Sysadm* to carry out two attacks namely (1) obtain passwords from the VM memory and (2) Obtain private keys and hence carry out cold-boot attacks [31]. To avoid this type of attacks [71] reduced the trusted computing base in Xen, which takes away user-space control from $DOM0$ therefore a user logged on to $DOM0$ cannot use any tool to access the VMs memory.

Security: The TVMM in [28] is built based on the architecture in [71] to provide the close box execution protection. With this protection it means that a *Sysadm* have no means to access the memory of the VMs under his control. **TCCP** offers protection against **VM substitution and Host substitution attacks** by (1) The user encrypts the VM image and sends it to the Node for launch, The Node sends the encrypted image to the **TC** which authorises the trusted node to launch the VM by decrypting the VM and encrypted it with the Node's trusted key on which the VM is to be launched (2) Only a trusted Node can decrypt the VM send from the TC. For secure migration, this protocol requires both Nodes to authenticate each other and also to prove that they are trusted. Confidentiality is provided by ensuring that the VM is encryption and it is hashed to provide **integrity**. Nonce are used to ensure freshness of the service requests and hence prevent replay attacks. Furthermore this TCCP architecture

is based on [71] which ensures that the *Sysadm* have no access to the VM memory therefore it is secured against memory snapshots attacks 6.1.3.2. On the downside; the protocol does not provide **non-repudiation** guarantees. Furthermore the trusted nodes are vulnerable to cold-boot attacks because **TCCP** keeps the trusted keys in the Node's memory until the Node is restarted [35].

Comments: The protocol presented good theoretical trusted launch security principles, however the authors did not implement the protocol. The authors also did not explain how the TVMM could enforce the property that takes away the user-space control from $DOM0$. Furthermore the attestation only happens between the TC and the nodes, the user is required to completely trust the TC without means to verify that the VM is launched on a trusted Node. Finally to secure trusted keys, we suggest that the protocol use TPM to seal the trusted keys instead of keeping the trusted keys in memory to avoid a possibility of node impersonation.

7.2.2 Integrity Attestation

Schiffman et al. [32] provided complete transparency to the launch and the attestation process, which enables the users to verify that their VM has been launched on the host machine that meets all security requirements and that their launched VM was not substituted or compromised while in transit. In addition the protocol enforces runtime integrity for the application running on the VMs.

Security: The protocol uses attestation for cloud controller to verify the Nodes and for the users to attest to the cloud controller and the Nodes. This means that for the VM data to be accessed by Node i, the user need to send the decryption key to Node i but only after Node i successfully attest to the user. This means that (1) a Node need to be trusted for a VM to launch because the user cannot send a decryption key if the Node is not verified and (2) the user verifies the VM integrity before sending the decryption key to Node i. With these mechanisms, The Node is expected to attest to the cloud verifier and the user, if the attestation fails; the Node will have no access to the VM data. This mechanism only provide security against **VM substitution 6.1.3.4 and Host substitution 6.1.3.3 attacks**. The protocol however is vulnerable to memory snapshots related attacks 6.1.3.1 6.1.3.2. We however not that this protocol focus on the verification of the launch process and without close box execution, the *Sysadm* could launch passive attacks such as obtaining passwords from the VM memory and data leak.

Comments: The protocol focus more on the verification and transparency of the VM launch process. It provides transparency and prevent some attacks as mentioned above. However the protocol requires more interventions from the

users; making it unsuitable for cloud deployment, therefore automating some of the actions that requires users interventions could solve this problem. Furthermore the mechanism does not give a clear indication on how it can be incorporated into trusted launch protocols such as the one by Santos et al. [28]. Finally the protocol provides security measures such as VM integrity and VM confidentiality.

7.2.3 Secure VM Launch and VM Migration

Security: Aslam et al. [34]protocol is specific to the enterprise environment and requires the VM Image to be pre-packaged. The protocol provides data **confidentiality** by using cryptographic mechanisms to bind the VM image to the trusted platform and by encrypting the VM image so that only the hosts within the trusted platform can decrypt the VM image. Furthermore to prevent attacks that require access to the VMs' memory and taking snapshots; the mechanism enforces access rights policy which contain the list of the users with privileges to do so. These users can be from both the *CSP* or the enterprise that uses the cloud services. **Integrity** is provided using TCG remote attestation mechanisms which allows the user to verify the cloud platform. **Non-repudiation** is an important aspect especially in the enterprise architecture in which one or more users can initiate the VM launch process; this property gives proof that a user requested to launch a VM by requiring the user to send a signature during the launch process. In order to prevent replay attacks on messages, nonce is required in the attestation process. This protocol is secured against all the attacks in subsection 6.1.3 on condition that the privileged *Sysadm* account have no access to the VMs' memory.

Comments: This protocol provided strong VM launch and migration security properties; and commentable security properties such as non-repundiation which is essential in the enterprise settings. However the approach to protect the VMs' memory from access is not effective enough because the protocol requires the limited number of cloud provider accounts to have this privileges. Even though the users may know the accounts that can access their VMs' memory, this in itself requires the users to trust the cloud provider employees. This weakens the protocol's security against memory snapshots attacks 6.1.3.1 6.1.3.2. We recommend that the protocol completely removes user-control rights like in [71].

7.2.4 Secure Generic VM Launch and Migration

Security: This protocol makes the assumption that the compute host offers close box execution such that the memory of the VM cannot be accessed by a privileged user. It is based on the work of Aslam et. al [34] but extended it to accommodate generic VMs. Confidentiality is achieved through cryptographic keys, TPM binding and sealing is used to protect the keys. VM integrity is verified before it is launched on a host and the host integrity is verified before

a VM is launched; therefore the protocol is secured against VM substitution attacks 6.1.3.4 and Host substitution attacks 6.1.3.3. The launch process is provided so that the user gets a proof that the VM is launch on a trusted host and it was not modified in transit. Based on the assumptions made, the protocol is secured against the memory based attacks 6.1.3.1 6.1.3.2.

Comments: This protocol is based on [34] therefore it provided as good security properties as in [34]. However the assumption made by the authors of a close box execution is unrealistic and makes the the protocol weak against memory based attacks 6.1.3.1 6.1.3.2.

7.3 Benefits of Secure Storage and Trusted Launch

7.3.1 Trusted Launch

Providing users with security guarantees of the virtual machine launch and migration process is of the out most importance. The core properties of trusted launch mechanism is (1) TPM is used to provide trust (2) the users get assurance that their VM is launched in a trusted platform. This section looks at how trusted launch protect users computational resources in the*IaaS*

- **Close box execution:** Trusted launch protocols provide close box execution, the importance of this mechanism is that it prevents a *sysadmin* with privileged rights to access memory of the VM running on a host computer under his control. The importance of this property is that it prevents tools or software to take a VM memory snapshots and therefore makes virtualization in the cloud immune to attacks that targets the VM memory.
- **Confining VM in a trusted platform:** The use of cryptographic techniques such as encryption ensures that the VM data can only be accessed by a host that is within the trusted platform because the decryption key is only sent to the host once the attestation is successful.
- **Integrity Verification:** The users have a chance to verify the integrity of the VM image before it is launched on the VM and to verify the integrity of the platform and host the VM is to be launched; in addition the host can also verify the image integrity in order to prevent VM-to-VM attacks and VM substitution attacks.
- **Confidentiality:** Cryptographic techniques such as encryption is used to protect the VM data from unauthorized access.

7.3.2 Secure Storage

Providing security at the lowest layer of cloud computing does not guarantee the security of data that is stored on the servers. The use of searchable symmetric techniques mitigate some of the security concerns.

- **Regulatory Compliance:** Data protection laws in most countries requires the organizations that store sensitive data such as health records and personal records to protect this data. If this data is not protected the organization is held responsible. SSE gives assurance that the data remains confidential and private because the data is encrypted before it is stored on the cloud and the customer retains the decryption keys.
- **Security Breaches:** Although cloud providers implements strong security mechanism, data breach can happen. For example; a malicious insider can steal the data and sell it interested parties. If any of the security breaches happens, the users' data is encrypted and therefore it remains confidential.
- **Subpoenas:** Investigations by the law enforcement officers may require access to the users data. However, this data might be requested from the cloud provider, at the same time, the cloud provider might be prevented to notify the user. In this case the user is kept in the dark about the disclouser of their data; although the subpoena might be challenged, the user does not get that option [36]. However, storing encrypted data in the cloud ensures that only the user can provided unencrypted data in this instance

7.4 Summary

We presented an analysis of the SSE techniques from the security, efficiency and privacy point of view. In addition, we have looked at the trusted mechanisms and the level of security they provide against the possible attacks. Furthermore, we discussed our findings and made recommendations on how the protocols can be improved in order to provided strong security guarantees and to be practical for cloud deployment. Finally we presented benefits of using the trusted launch from the enterprise or organization perspective.

Chapter 8

Conclusion

This thesis presented a state of the art review of security mechanisms that are designed to protect data confidentiality, integrity and privacy primarily from insider threats in an untrusted or semi-trusted cloud set up. This mechanisms are suitable for the *IaaS* and *PaaS* service models, because these are the service models that are most likely to used by industry and governments.

In *IaaS* the goal is to use Trusted Computing technologies to provide strong security notions for secure trusted launch. Trusted launch idea is to provide a closed box execution which prevents a privilege user of the host computer to access the memory of the virtual machines residing in it. In addition TPM and cryptographic techniques such as encryption, hashing, signatures and nonce; are used to confine the virtual machines within a trusted platform and therefore preventing host and VM image substitution attacks. TL provide confidentiality, integrity verification and non-repudiation.

One of the challenges that could hinder the deployment of these protocols in the real world is that all the protocol did not provide a clear outline of the closed box execution which prevents a *sysadmin* to access the VMs memory. If the VM memory is not protected, then the threat of malicious insiders remains.. Although closed box execution provides protection against devastating memory snapshots attacks as demonstrated in [54, 55], none of the mechanism gave a detailed design on how it can be achieved except in [71] but this was specific to Xen hypervisor and has not been implemented yet. Furthermore the security provided by these mechanisms heavily depends on the security of the TMP,therefore, all attacks that can successfully break the TPM [72] breaks the whole trusted launch protocols.

Once the trusted launch is in place, we need to secure the storage. This is because trusted launch does not protect users data that is stored on the cloud. The host computer have access to the location where the VM file is stored and read or write operations can take place if it is not encrypted. It is common practice that the users use standard OS built-in disk encryption tools such as Bitlocker [73] for Windows or dm-crypt for Linux; to protect the data, however since data need to be decrypted before operations is done, this does not protect the data from insider

malicious *SysAdm*; therefore data should be stored encrypted on the cloud.

In *PaaS* and *IaaS* for data storage purposes, users need to send their data to the cloud encrypted so that it remains confidential and private. Traditional encryption schemes are not suitable in this case because (1) because those schemes do not allow computations and operations to be done on the data (2) the data need to be decrypted first therefore the confidentiality of data depends on the entity that is managing the keys. To provide strong security guarantees, the user should be the one to decrypt the data and the TTP can be used for key management especially by enterprises which does not have enough resources to invest in key management. These security guarantees can be provided either by using symmetric cryptography or public-key cryptography. However, public-key schemes are less efficient because their search time is linear in the number of documents [38].

Searchable Symmetric Encryption schemes are efficient than public-key schemes, this is one of the property that makes these schemes fit for the cloud. Although these schemes have a trade-off for privacy over efficiency; the extend to which the leakage attacks exposes the privacy of the data is not yet clear apart from knowing the exact keyword contained in the files. However these knowledge can be combined with other types of attacks to learn more about the customer data. To date, there has been no proof that the data can be recovered apart from the associated keyword. Although these schemes do not provide complete privacy of the query, they still maintain the confidentiality of the files.

The ideal "Clear Cloud" would be a solution that incorporates TL and SSE schemes. This way, the users have a transparent view of the VM launch process and that they are assured that no other entity can read their data except with the authorised party. Although there are still challenges such as providing a complete close box execution in the TL mechanisms and minimizing the leakage functions without decreasing the efficiency of SSE schemes; these presents a good direction towards transparency in the IaaS.

The research on the trusted launch and SSE has made great progress however more need to be done to work around a solution that can protect the VMs' memory access by the host computer. Furthermore SSE schemes proved to be suitable for the cloud, however the extend to which the leakage in SSE schemes can compromise the data security need to be investigated further to establish whether the leakage can be accepted as the trade-off for efficiency. Trusted launch mechanisms rely heavily on a trusted third party, this may require a new business model. We believe that these are the issues that hinders the adoption of this mechanisms.

References

[1] P. Mell and T. Grance, "The nist definition of cloud computing," 2011. 1, 6, 7, 8

[2] L. Wei, H. Zhu, Z. Cao, X. Dong, W. Jia, Y. Chen, and A. V. Vasilakos, "Security and privacy for storage and computation in cloud computing," *Information Sciences*, vol. 258, pp. 371 – 386, 2014. 1

[3] 2016. 2

[4] P. You, Y. Peng, W. Liu, and S. Xue, "Security issues and solutions in cloud computing," in *2012 32nd International Conference on Distributed Computing Systems Workshops*, pp. 573–577, June 2012. 2, 8, 17

[5] B. James, "Security and privacy challenges in cloud computing environments," 2010. 2

[6] C. Modi, D. Patel, B. Borisaniya, A. Patel, and M. Rajarajan, "A survey on security issues and solutions at different layers of cloud computing," *The Journal of Supercomputing*, vol. 63, pp. 561–592, February 2013. The final publication is available at Springer via http://dx.doi.org/10.1007/s11227-012-0831-5. 2, 8, 17, 37

[7] A. Michalas, N. Paladi, and C. Gehrmann, "Security aspects of e-health systems migration to the cloud," in *e-Health Networking, Applications and Services (Healthcom), 2014 IEEE 16th International Conference on*, pp. 212–218, IEEE, 2014. 6

[8] A. Michalas and T. Giannetsos, "The data of things: Strategies, patterns and practice of cloud-based participatory sensing," in *Proceedings of the 1st International Conference on Innovations in InfoBusineess and Technology*, 2016. 8

[9] A. Michalas and M. Bakopoulos, "Secgod google docs: Now i feel safer!," in *Internet Technology And Secured Transactions, 2012 International Conference for*, pp. 589–595, Dec 2012. 8

[10] A. Michalas and R. Dowsley, "Towards trusted ehealth services in the cloud," in *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pp. 618–623, Dec 2015. 8

[11] Y. Verginadis, A. Michalas, P. Gouvas, G. Schiefer, G. Hbsch, and I. Paraskakis, "Paasword: A holistic data privacy and security by design framework for cloud services," in *Proceedings of the 5th International Conference on Cloud Computing and Services Science*, pp. 206–213, 2015. 8

[12] N. Paladi and A. Michalas, ""One of our hosts in another country": Challenges of data geolocation in cloud storage," in *Wireless Communications, Vehicular Technology, Information Theory and Aerospace Electronic Systems (VITAE), 2014 4th International Conference on*, pp. 1–6, May 2014. 8

[13] A. Michalas, N. Komninos, N. R. Prasad, and V. A. Oleshchuk, "New client puzzle approach for dos resistance in ad hoc networks," in *Information Theory and Information Security (ICITIS), 2010 IEEE International Conference*, pp. 568–573, IEEE, 2010. 8

[14] A. Michalas, N. Komninos, and N. R. Prasad, "Mitigate dos and ddos attack in mobile ad hoc networks," *International Journal of Digital Crime and Forensics (IJDCF)*, vol. 3, no. 1, pp. 14–36, 2011. 8

[15] A. Michalas, N. Komninos, and N. Prasad, "Multiplayer game for ddos attacks resilience in ad hoc networks," in *Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (Wireless VITAE), 2011 2nd International Conference on*, pp. 1–5, Feb 2011. 8

[16] A. Michalas, N. Komninos, and N. R. Prasad, "Cryptographic puzzles and game theory against dos and ddos attacks in networks," *International Journal of Computer Research*, vol. 19, no. 1, p. 79, 2012. 8

[17] G. P. Sharma, "Virtualization in cloud computing," 10

[18] G. J. Popek and R. P. Goldberg, "Formal requirements for virtualizable third generation architectures," *Commun. ACM*, vol. 17, pp. 412–421, July 1974. 10

[19] "Trusted computing." `http://www.trustedcomputinggroup.org/trusted-computing/`. Accessed: 2016-07-23. 10, 11

[20] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, "Terra: A virtual machine-based platform for trusted computing," *SIGOPS Oper. Syst. Rev.*, vol. 37, pp. 193–206, Oct. 2003. 13, 21, 22, 49

[21] C. Bösch, P. Hartel, W. Jonker, and A. Peter, "A survey of provably secure searchable encryption," *ACM Computing Surveys (CSUR)*, vol. 47, no. 2, p. 18, 2015. 15, 29, 35

[22] N. Paladi, A. Michalas, and C. Gehrmann, "Domain based storage protection with secure access control for the cloud," in *Proceedings of the 2014 International Workshop on Security in Cloud Computing*, ASIACCS '14, (New York, NY, USA), ACM, 2014. 15

[23] M. Hohmuth, M. Peter, H. Härtig, and J. S. Shapiro, "Reducing tcb size by using untrusted components: small kernels versus virtual-machine monitors," in *Proceedings of the 11th workshop on ACM SIGOPS European workshop*, p. 22, ACM, 2004. 16

[24] R. Bhadauria and S. Sanyal, "Survey on security issues in cloud computing and associated mitigation technsique," *J. Comput. Appl.*, vol. 47, pp. 47–66, June 2012. 17

[25] K. Neela and V. Kavitha, "A survey on security issues and vulnerabilities on cloud computing," *Int. J. Comput. Sci. Eng. Technol*, vol. 4, no. 7, pp. 855–860, 2013. 17

[26] M. Nazir, "Cloud computing: overview & current research challenges," *IOSR Journal of Computer Engineering (IOSR-JCE) ISSN*, pp. 2278–0661, 2012. 17

[27] F. Sabahi, "Virtualization-level security in cloud computing," in *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, pp. 250–254, May 2011. 17

[28] N. Santos, K. P. Gummadi, and R. Rodrigues, "Towards trusted cloud computing," in *Proceedings of the 2009 Conference on Hot Topics in Cloud Computing*, HotCloud'09, (Berkeley, CA, USA), USENIX Association, 2009. 17, 21, 22, 23, 38, 49, 51

[29] R. L. Krutz and R. D. Vines, *Cloud Security: A Comprehensive Guide to Secure Cloud Computing*. Wiley Publishing, 2010. 18

[30] T. Garfinkel, M. Rosenblum, and D. Boneh, "Flexible os support and applications for trusted computing," in *Proceedings of the 9th Conference on Hot Topics in Operating Systems - Volume 9*, HOTOS'03, (Berkeley, CA, USA), pp. 25–25, USENIX Association, 2003. 21

[31] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten, "Lest we remember: cold-boot attacks on encryption keys," *Communications of the ACM*, vol. 52, no. 5, pp. 91–98, 2009. 21, 39, 49

[32] J. Schiffman, T. Moyer, H. Vijayakumar, T. Jaeger, and P. McDaniel, "Seeding Clouds with Trust Anchors," in *CCSW '10: Proceedings of the 2010 ACM Workshop on Cloud Computing Security*, ACM, 2010. 23, 26, 50

[33] S. W. Smith, "Outbound authentication for programmable secure coprocessors," in *Proceedings of the 7th European Symposium on Research in Computer Security*, ESORICS '02, (London, UK, UK), pp. 72–89, Springer-Verlag, 2002. 23

[34] M. Aslam, C. Gehrmann, L. Rasmusson, and M. Björkman, "Securely launching virtual machines on trustworthy platforms in a public cloud," 2012. 25, 26, 51, 52

[35] N. Paladi, C. Gehrmann, M. Aslam, and F. Morenius, "Trusted launch of virtual machine instances in public iaas environments," in *International Conference on Information Security and Cryptology*, pp. 309–323, Springer, 2012. 26, 50

[36] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *International Conference on Financial Cryptography and Data Security*, pp. 136–149, Springer, 2010. 27, 28, 53

[37] P. Yong, Z. Wei, X. Feng, Z.-h. DAI, G. Yang, and D.-q. CHEN, "Secure cloud storage based on cryptographic techniques," *The Journal of China Universities of Posts and Telecommunications*, vol. 19, pp. 182–189, 2012. 28

[38] R. Dowsley, A. Michalas, and M. Nagel, "A report on design and implementation of protected searchable data in iaas," tech. rep., Swedish Institute of Computer Science (SICS), 2016. 28, 29, 35, 36, 47, 48, 55

[39] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, (New York, NY, USA), pp. 79–88, ACM, 2006. 28, 31, 32, 34, 45, 46, 48

[40] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 965–976, ACM, 2012. 28, 32, 33, 34, 35, 46, 47

[41] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Security and Privacy, 2000. S P 2000. Proceedings. 2000 IEEE Symposium on*, pp. 44–55, 2000. 29, 30, 45, 48

[42] E.-J. Goh *et al.*, "Secure indexes.," *IACR Cryptology ePrint Archive*, vol. 2003, p. 216, 2003. 30, 31, 46, 48

[43] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970. 30

[44] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proceedings of the Third International Conference on Applied Cryptography and Network Security*, ACNS'05, (Berlin, Heidelberg), pp. 442–455, Springer-Verlag, 2005. 31, 46, 48

[45] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," in *International Conference on Financial Cryptography and Data Security*, pp. 258–274, Springer, 2013. 34, 35, 36, 47, 48

[46] M. Naveed, M. Prabhakaran, and C. A. Gunter, "Dynamic searchable encryption via blind storage," in *2014 IEEE Symposium on Security and Privacy*, pp. 639–654, May 2014. 35, 36, 47

[47] D. A. Fernandes, L. F. Soares, J. V. Gomes, M. M. Freire, and P. R. Inácio, "Security issues in cloud environments: a survey," *International Journal of Information Security*, vol. 13, no. 2, pp. 113–170, 2014. 37

[48] T. C. for Internet Security, *Virtual Machine Security Guidelines*. Version 1.0, 2007. 37

[49] 2007. 38

[50] W. Dawoud, I. Takouna, and C. Meinel, "Infrastructure as a service security: Challenges and solutions," in *Informatics and Systems (INFOS), 2010 The 7th International Conference on*, pp. 1–8, March 2010. 38

[51] N. Paladi, C. Gehrmann, and A. Michalas, "Providing user security guarantees in public infrastructure clouds," 2016. 38, 39

[52] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, pp. 198–208, Mar 1983. 38

[53] S. Davidoff, "Cleartext passwords in linux memory," *Massachusetts institute of technology*, pp. 1–13, 2008. 39

[54] F. Rocha and M. Correia, "Lucy in the sky without diamonds: Stealing confidential data in the cloud," in *2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 129–134, June 2011. 39, 54

[55] M.-D. Nguyen, N.-T. Chau, S. Jung, and S. Jung, "A demonstration of malicious insider attacks inside cloud iaas vendor," *International Journal of Information and Education Technology*, vol. 4, no. 6, p. 483, 2014. 39, 40, 54

[56] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 668–679, ACM, 2015. 41, 42, 43, 46, 47

[57] T. Dimitriou and A. Michalas, "Multi-party trust computation in decentralized environments," in *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on*, pp. 1–5, May 2012. 41

[58] A. Michalas, T. Dimitriou, T. Giannetsos, N. Komninos, and N. Prasad, "Vulnerabilities of decentralized additive reputation systems regarding the privacy of individual votes," *Wireless Personal Communications*, vol. 66, no. 3, pp. 559–575, 2012. 41

[59] K. Yigzaw, A. Michalas, and J. Bellika, "Secure and scalable statistical computation of questionnaire data in r," *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2016. 41

[60] T. Dimitriou and A. Michalas, "Multi-party trust computation in decentralized environments in the presence of malicious adversaries," *Ad Hoc Netw.*, vol. 15, pp. 53–66, Apr. 2014. 41

[61] A. Michalas and N. Komninos, "The lord of the sense: A privacy preserving reputation system for participatory sensing applications," in *Computers and Communication (ISCC), 2014 IEEE Symposium*, pp. 1–6, IEEE, 2014. 41

[62] A. Michalas, V. A. Oleshchuk, N. Komninos, and N. R. Prasad, "Privacy-preserving scheme for mobile ad hoc networks," in *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pp. 752–757, June 2011. 41

[63] A. Michalas, M. Bakopoulos, N. Komninos, and N. R. Prasad, "Secure amp; trusted communication in emergency situations," in *Sarnoff Symposium (SARNOFF), 2012 35th IEEE*, pp. 1–5, May 2012. 41

[64] A. Biryukov, *Adaptive Chosen Plaintext and Chosen Ciphertext Attack*, pp. 21–21. Boston, MA: Springer US, 2011. 41

[65] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation.," in *NDSS*, vol. 20, p. 12, 2012. 42, 46, 47

[66] Y. Zhang, J. Katz, and C. Papamanthou, "All your queries are belong to us: The power of file-injection attacks on searchable encryption," 43, 46, 47, 48

[67] C. Liu, L. Zhu, M. Wang, and Y.-a. Tan, "Search pattern leakage in searchable encryption: Attacks and new construction," *Information Sciences*, vol. 265, pp. 176–188, 2014. 43, 47

[68] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious rams," *Journal of the ACM (JACM)*, vol. 43, no. 3, pp. 431–473, 1996. 45

[69] M. Naveed, "The fallacy of composition of oblivious ram and searchable encryption," tech. rep., Tech. rep., Cryptology ePrint Archive, Report 2015/668, 2015. 45

[70] E. Stefanov, C. Papamanthou, and E. Shi, "Practical dynamic searchable encryption with small leakage.," in *NDSS*, vol. 14, pp. 23–26, 2014. 48

[71] D. G. Murray, G. Milos, and S. Hand, "Improving xen security through disaggregation," in *Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pp. 151–160, ACM, 2008. 49, 50, 51, 54

[72] 2016. 54

[73] 2016. 54