

Methodology for Deploying Secure Applications in the Cloud

Master Thesis, July 6, 2011

Onyekwum Charles

Methodology for Deploying Secure Applications in the Cloud

THESIS

Submitted in partial fulfilment of
the requirements for the degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

TRACK INFORMATION ARCHITECTURE

by

Onyekwum Charles

Born in Lagos, Nigeria



Web Information Systems Group
Department of Software Technology
Faculty EEMCS, Delft University of
Technology
Delft, The Netherlands
<http://eemcs.tudelft.nl>



Exact Holding N.V.
Molengraaffsingel 33,
2629 JD Delft,
Netherlands
www.exact.nl/

Methodology for Deploying Secure Applications in the Cloud

Author: Onyekwum Charles
Student ID: 4031849
Email: c.o.onyekwum@student.tudelft.nl

ABSTRACT

There are challenges that lie in migrating enterprise applications into the cloud. Certain enterprises might want data to be partly hosted on premise in the enterprise and also partly hosted in the cloud, while other enterprises might their applications to be fully hosted on the cloud. Such hybrid possibilities can enable enterprises to benefit from cloud-based architectures, while honouring application performance requirements, but privacy issues and security considerations is always a problem during migration

This work highlights the need for security in the cloud; Different features offered by cloud providers are explored. Concepts like Virtual machine migration and a hardware chip that can be used as a basis for trust in the cloud are discussed. Pre-migration hints shows what needs to be done before the actual secure migration paths set-in. The migration path is based on a generalisation of the kind of threats these features are exposed to. There are arguments as to why such paths are the most secure.

It is a fact that cloud computing infrastructures enable companies to cut costs by outsourcing computations on-demand, however, clients of cloud computing services currently have no well acknowledged means of verifying the confidentiality and integrity of their data and computation. To address this problem, it was part of the migration strategy in this work to design a system that enables a cloud provider to provide an execution environment that guarantees integrity in the execution of virtual machines for its clients, but the design works in hand with a trusted third party.

Graduation Committee
Prof. dr. ir. G.J. Houben, Faculty EEMCS¹, TU Delft
Dr. ir. A.J.H. Hidders, Faculty EEMCS, TU Delft
Ir. H.J.A.M. Geers, Faculty EEMCS, TU Delft

¹ Electrical Engineering, Mathematics and Computer science

Preface

This document has been produced in the master thesis project as part of the Master program in Computer science invariant Information Architecture at Delft University of Technology (TU Delft) from 01/11/2010 until 06/07/2011. It was done in part to show how Exact, can migrate to the public cloud in the most secure way.

The completion of this graduation project has been an enormous endeavour that required continuous persistence, social interaction skills, and constant hard work. It has given me unique insight into cloud computing, especially with the aspect of migrating and securing data in the cloud, a field that is not fully mature in computer science.

Exact in the near future commence its business on a wider scale, using the public cloud as a tool to expanding its services, and I am delighted that I had the chance to show the Security Insights into a migration to the Public cloud as part my Master thesis.

This thesis could not have been completed without the guidance, assistance and support of a number of people: First of all, my immediate supervisor at the TU Delft, Jan Hidders, who always made time to discuss the thesis project and was very helpful in determining the theoretical course. My supervisor at Exact, Bart Platzbeecker, who provided all the resources I needed to be able to complete my work. And finally my direct colleagues within my project-group(*the Alpha group*), who always asked questions that made me do more research into this work , and hereby made it even more successful.

Onyekwum Charles
Delft, The Netherland
July 6, 2011

Contents

Abstract.....	5
Preface.....	6
Contents.....	8
List of Figures.....	11
List of Tables	13
Part I: Background and Context.....	15
1 Introduction.....	17
1.1 Background	17
1.2 Problem	18
1.3 Motivation.....	18
1.4 Research Goal	20
1.5 Research Approach.....	20
1.6 Research Structure.....	22
2 Security Considerations	23
2.1 Identification and Authentication.....	24
2.2 Authorization	24
2.3 Confidentiality	24
2.4 Integrity.....	25
2.5 Non-Repudiation	25
2.6 Availability	25
2.7 Hybrid Interoperability	26
2.8 Internal Applications	28
2.9 The Private Cloud	29
2.10 The Users	29
2.11 Security Area of Interest	29
Part II: The Cloud Features, Virtual Machines and TPM.....	31

3	The Cloud Features and Guarantees	33
3.1	The load Balancing	33
3.2	Auto Scaling	33
3.3	Virtual Internet Protocol Addresses	34
3.4	Virtual Private Cloud (VPC)	34
3.5	Queues	34
3.6	Storage.....	35
4	Virtual Machines and The TPM Chip.....	36
4.1	Virtualization	36
4.2	Trusted Computing and The Trusted Platform Module (TPM)	36
4.3	Virtual TPM	38
4.4	Virtual Machine Migration and Its Attacks	38
Part III: Pre-Migration.....		41
5	Pre-Migration	43
5.1	Identify the Asset to be Migrated to the Cloud	43
5.2	The Planning Phase	43
5.3	Application Profiling	44
5.4	Application Evaluation	45
5.5	Backup Data.....	46
5.6	A Full Migration	47
5.7	Gradual Migration	47
Part IV: Secure Deployment of Applications		48
6	Deploying Applications in the Cloud	50
6.1	Exact Interview Information.....	50
6.2	User Locality In the Cloud	50
6.3	The Integration and Middleware Layer	52
6.4	The Core Connectivity and Delivery Layer	55
6.5	Abstraction layer.....	56
6.6	A Firewall For Hybrid Migration.....	60
6.7	Migrating Internal applications.....	62

6.8	The Hardware layer	64
6.8.1	Performance and Key handling: NetLib	65
6.8.2	SQL Injection Attacks.....	67
6.9	The Elastic IP (EIP) feature	67
7	Design Pattern to Minimize the Insider Threat.....	73
7.1	Malicious Insiders in the Public Cloud.....	73
7.2	The Facilities layer	73
7.3	Design to Minimize Malicious Insiders	75
7.4	On Instance boot up	85
7.5	Virtual Trusted Platform Modules (vTPMs) and Exposure to attacks	86
7.6	Monitoring the Machine acting as root-of-trust	90
7.7	Migrating Virtual Machines.....	91
Part V:	Conclusion.....	94
8	Conclusion and Reflection	96
8.1	Contribution	96
8.2	Research Questions.....	96
8.3	Reflection.....	97
8.3.1	Limitations	97
8.3.2	Future Work and Final remarks.....	98
Appendices....		99
Appendix 1:	The Cloud Reference Model.....	101
Appendix 2:	The Typical Secure Cryptographic Communication	103
Appendix 3:	Useful Commands	105
Appendix 4:	Trusted Virtual Domain	108
References		110
Acronyms		118

List of Figures

Figure 1: Divergence, Transformation, Convergence	20
Figure 2: The Cloud Model Stack with a three fold approach	21
Figure 3: Challenges Anticipated in Cloud Computing.....	23
Figure 4: Cloud Computing Security Requirements	24
Figure 5: Microsoft's Approach to Threat Modeling	26
Figure 6: The Hybrid Cloud and Area of focus	28
Figure 7: Crypto Capabilities and Contents of the TPM Chip.....	37
Figure 8: Communication between TPM and Attestation service	37
Figure 9: Machine Arrangement in Clouds Datacenter	39
Figure10: A Man-in-the-Middle Attack.....	40
Figure11: Region to Zone Connectivity in the Cloud.....	51
Figure12: A Queue as a communication path in the Cloud	53
Figure 13: Transformation: Introduce a New Instance.....	54
Figure 14: Potential Back up Design for the Instance	54
Figure 15: Exposure to Snooping Attacks	55
Figure 16: An Elastic Block Store Launch	58
Figure 17: Webrole to Worker Role Communication	58
Figure 18: A Plausible Design in the Azure cloud	59
Figure 19: Firewall Connection from Private to Public cloud.....	60
Figure 20: An Hybrid Approach for Connectivity in Azure.....	62
Figure 21: Instance Connection Setup	63
Figure22: Safer strategy to migrate internal applications.....	63
Figure 23: Passing Decryption keys to a Database Instance	65
Figure 24: EIP feature.....	68
Figure 25: Hacker Attack on the Flaw of the EIP	69
Figure 26: Flowchart Migration Methodology	71
Figure 27: Bond between a Lobot agent and an aDom.....	75
Figure 28: Inside the Private Virtual Infrastructure Factory (PVIF)	75
Figure 29: Ticket-based Local Attestation scheme using MAT	76
Figure 30: Root of Trust Machine in an Isolated region.....	77
Figure 31: The Machine/ATM arrangement within a zone(s).....	78
Figure 32: Location of Root of Trust machine in Cloud.....	79
Figure 33: Unknown Machine for VM launch.....	80
Figure 34: Accessing an Instance in the Cloud.....	81
Figure 35: The Cloud Controller in the Cloud	82
Figure 36: Trust Relationship between Machines in the Cloud	83

Figure 37: Sequence of Events for an Unacknowledged Machine	84
Figure 38: A trusted Machine	85
Figure 39: The vTPM Architecture with vTPM manager	86
Figure 40: Mapping of vTPMs instances to a Hardware TPM.....	87
Figure 41: Swapped Registry Values	87
Figure 42: Inserting Dom 0 Functionalities into Dom U	89
Figure 43: Monitoring Machine A	90
Figure 44: A Migration from Machine 1 to Machine 2	91
Figure 45: Cloud Auditor Flowchart	92
Figure 46: Cloud Reference Model.....	101
Figure 47: Secure Cryptographic Communication	103
Figure 48: Trusted Virtual domains.....	108

List of Tables

Table 1: A Threat to Property mapping	26
Table 2: Mapped ‘Area of Interest’ to STRIDE	30
Table 3: The Amazon Instance types.....	45
Table 4: Table of Comparison	51
Table 5: Customer to Location of Exact users	52
Table 6: User to Privilege Mapping to Blobs	56
Table 7: The Amazon Instance Types.....	57
Table 8: Comparison of NetLib and Encrypting File System (EFS).....	64

Part I

Background and Context

1 Introduction

The aim of this master's thesis is to show how an organisation can migrate securely to the public cloud. The Organisation here refers primarily to Exact , a company which intends to do its business in the Public cloud in the near future.

Exact [73] is an organisation, founded in 1984, whose function is to serve entrepreneurial businesses with information technology by delivering business software solutions. These software solutions are aimed at enabling the integration and consolidation of all business areas into one global platform. The global platform includes processes such as Enterprise resource planning (ERP), Customer relationship management (CRM), Human resource management (HRM), project management etc, all connected together. The software solutions include but are not limited to Exact Globe, Exact synergy, Exact Online (EOL), Exact Identity service (EIS). Currently EOL and EIS are hosted on a private cloud, Rackspace. Exact Globe is a desktop application and Exact Synergy is a web application .Both are used within Exact, but are also sold to their customers.

1.1 Background

The National Institute of standards and Technology (NIST) [72] defines cloud computing as a model for enabling on-demand network access to a shared pool of configurable computing resources (i.e servers, storage, applications etc) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This rapid provision and minimal management effort can help to deliver scalable computing resources as a service with the use of Internet technology. With this, users can rent virtual machines (virtual machines is explained further in *chapter 4*) from a service provider, placing their data in the control of a third party. The data placed into the hands of the third party is one of the major security challenges that exist in the cloud, according to the cloud security alliance [64], as it could lead to sensitive data loss or leakage.

A migration [74] to the cloud logically would start from the process of firstly understanding the target platform or infrastructure and then trying to map that to where the data is currently hosted (i.e either in the private cloud or within the enterprise). The target infrastructure is the public cloud .The public cloud however offers three different services that are grouped as Platform as a service, Infrastructure as a service, and Software as a service each with unique features. Platform as a service (PaaS) provides a platform for building and running custom applications. The GoogleApp Engine and Microsoft Azure [34,74] are examples of the PaaS. Infrastructure as a service (IaaS) is an infrastructure delivered as a service. This infrastructure delivered as a service includes hardware (servers, networks, load balancers etc). Examples of IaaS providers are Amazon [11,42], Go Grid, and Flexiscale [74]. Software as a service (SaaS) is software that is deployed over the internet, but SaaS is out of scope in this work.

This thesis dwells on the security challenge when migrating data to any of the three major cloud providers: The Amazon cloud (IaaS), the Google cloud (PaaS), and the Microsoft Azure

(PaaS). With this, if any other PaaS or IaaS is chosen apart from the three mentioned above, the security mitigation strategies introduced in this work can be generalised to them

1.2 Problem

Cloud vendors deliver their services in a scalable way by sharing infrastructure [74]. Often, the underlying components that make up this infrastructure were not designed to offer strong isolation properties for a multi-tenant architecture. To address this gap, a virtualization hypervisor [6] mediates access between guest operating systems and the physical compute resources. Still, even hypervisors have exhibited flaws² that have enabled guest operating systems to gain inappropriate levels of control or influence on the underlying platform.

Data compromise [72] is another problem when migrating to the public cloud. Deletion or alteration of data records in the cloud is an example of data compromise. This threat of data compromise increases in the cloud due to the number of and interactions between risks and challenges which are either unique to cloud, or more dangerous because of the architectural or operational characteristics of the cloud environment.

Apart from data compromise, there is the problem of service hijacking where the service(s) provided by the cloud are used to eavesdrop on activities so that manipulation of data is achieved, and falsified information can be returned which can redirect clients to illegitimate sites. The users' account or service instances may then become a new base for the service hijacker. One of the major problems lays in the fact that there are different types of cloud providers none of whom would agree that some of its features can expose users to security lapses in the cloud. Instance failure [77] or corruption in the cloud for example can occur more frequently if a certain feature is chosen over another.

There is also the threat of malicious insiders[64] listed as one of the major security concerns by the cloud security alliance(CSA)[64], that is equivalent to trusting the system administrator(s) of the cloud provider that has privileged control all backend machines. This threat is amplified for consumers of cloud services by the convergence of IT services and customers under a single management domain, combined with a general lack of transparency into provider process and procedure. For example, a provider may not reveal how it grants employees access to physical and virtual assets, how it monitors these employees³, or how it analyzes and reports on policy compliance. To complicate matters, there is often little or no visibility into the hiring standards and practices for cloud employees. This kind of situation creates an attractive opportunity for an adversary within the cloud providers' domain, to perpetrate attacks such as accessing the memory of a customer's virtual machine. If the memory of a customer's virtual machine is extracted successfully by either a cloud administrator or an outside attacker, vital and sensitive information such as passwords, keys, application data and other protected data could be jeopardized.

1.3 Motivation

Designing secure migration plans, and migrating applications to a targeted cloud computing model or models is a demanding task. It requires a detailed application migration process design expertise and a deep understanding of the cloud computing model, as well as a detailed knowledge of how the applications interact with each other, and how they interact when they are

² For Hypervisor flaws, see: <http://theinvisiblethings.blogspot.com/2008/07/0wning-xen-in-vegas.html>

³ Hackers operating inside Amazon: <http://www.infoworld.com/d/cloud-computing/hackers-find-home-in-amazons-ec2-cloud-742>

moved to the cloud. It also requires to an extent some experience in integrating IT systems and cloud management. A provider may offer different services [80] in the cloud, so ensuring the security and integrity of users' information in the cloud becomes an issue as the management and ownership of the hosting platforms is removed from the control of a single facility and a single owner.

Many organizations like Exact are required to protect their data and those of their customers from compromise due to the sensitivity of the information. Before transforming an application to the cloud, they therefore need to understand the cloud features and adapt the application to run in its new environment. Hardware applications in the organisation typically separate processing and storage into individual partitions. But in the cloud, it involves virtual applications [6] where components are separated so that they can be scaled independently. Each component must be able to be served by multiple machines simultaneously so that as additional cloud resources are brought to bear, the application can take advantage of them. Mapping the operation of the hardware applications to the operation of the virtual applications in a secure way can be difficult⁴.

Migrating an application to the cloud equally has some uncertainties regarding the cloud provider. Such uncertainties include migrating data into a cloud providers cluster of machines that are being administered maliciously already discussed as part of the *Problem*. To help minimize this malicious possibility in the cloud, a consideration as part of this work would be to design user instances such that they can have an automated process to start, and be monitored, during their operation in the cloud. But this would be difficult to accomplish as most cloud providers have basic tools⁵ for automatically starting a virtual machine, although they do not address the state of what happens to the virtual machine once it has started up in their infrastructure or the security state of the infrastructure itself. To further add to the problem, virtual Machines can be duplicated and migrated to help balance load. This process is performed by copying or moving a virtual Machine state from one physical platform to another and can either be performed live, which is termed Live Migration [78]. Live migration [78,51], poses a significant challenge as an attacker can listen for migration requests by the target and intercept the image and state information from the source machine. This interception can be accomplished through spoofing, address Resolution Protocol (ARP) poisoning, Domain Name System (DNS) poisoning [27], or any other attacks that redirect traffic from the destination machine to the attacker. Since the state information is a copy of the VM's memory and processor state, the attacker can manipulate the image and machine state by implanting malicious software and then placing the compromised image on the destination server. The attacker now has access to the server's secure data stored on the Storage through the malicious software implant.

⁴ Setting up a virtual machine see : aws.amazon.com/ec2 , also see <http://www.vmware.com>

⁵ For difficulty to match hardware to Virtual appliance see: <http://priorartdatabase.com/IPCOM/000200546>

1.4 Research Goal

The combination of the problems identified in this chapter so far, the demand for a reliable Secure migration strategy has led to the formulation of the main research goal of my thesis:

"Create an approach to securely migrate an application(s) to the cloud, based on the features offered by the cloud provider"

In order to achieve the main research goal, a number of research questions have been constructed:

- (1) What are the security threats of migrating to the public cloud?
- (2) What cloud features offered by the cloud provider can expose an organisation migrating to the public cloud based on the threats in (1)?
- (3) What is a standard approach to ensure a secure migration to the cloud based on (2)?
- (4) How can an applications integrity be maintained in the cloud after the migration?

1.5 Research Approach

The approach consists generally of a sequential approach consisting of six steps. The scope of the research approach is not limited to the requirements of Exact only. Details of the activities performed are discussed in the sections indicated below. Since this work is primarily based on showing a secure way of migrating to the cloud, a *top down* approach starting from every layer in the cloud reference model[108, 63](see APPENDIX 1 for discussion of the cloud reference model) is used in combination with a horizontal ‘*Divergence ,transformation ,Convergence*’ approach[62, 75, 76] for each layer .

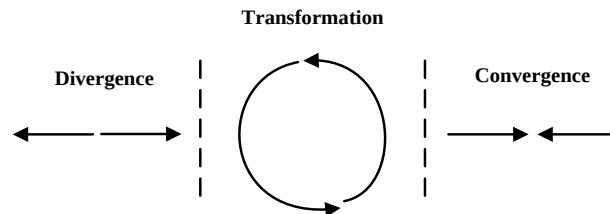


Figure 1: Divergence, transformation, convergence based on [62,75, 76]

Divergence as defined by Finger, S et al [62] is used in extending the boundary of any design situation in order to have a large enough solution search space. The search space here would refer to exploring different clouds within the layer of the cloud stack that they exist. Since there is no specific cloud type Exact has decided to migrate to. *Transformation* [62] is defined as the stage of pattern making and high-level creativity based on the information from the divergence. In this thesis transformation is aimed at narrowing down the information from the divergence,

offering a creative design to the layer of the cloud model .The objective would also be to impose upon the results of the divergent search a pattern that is precise enough to permit convergence to a single migration methodology. *Convergence* [75] would then be the end result because it is a reduction of the range of options to a single chosen methodology to deploy applications to the cloud regardless of the service (i.e IaaS or PaaS). In this work the divergence to transformation to convergence is called the *three fold approach*. Different existing features in the cloud would be mapped to every layer as shown below

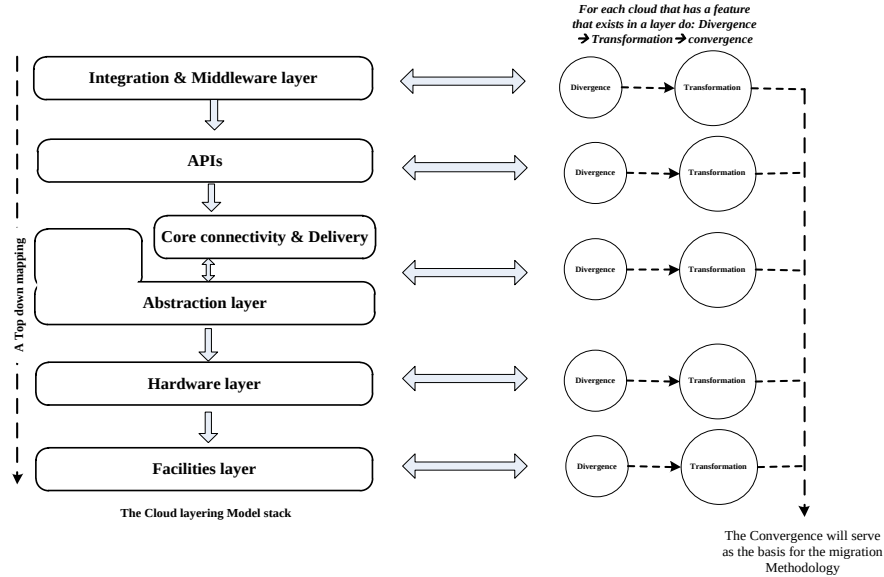


Figure 2 : The cloud model stack with a horizontal three fold approach, based on [62 ,75, 76]

The Steps highlighted below enumerate the stages of this work. They include a contextual analysis of the threats in the Public cloud, The cloud features and their components, Virtual machines and The TPM chip which is the basis for the design that is used in the chapter 7, Migration approach which contains the migration methodology, Integrity maintenance in the cloud which contains a unique design pattern to minimize the insider threat highlighted as part of the problem

Step 1: Context Analysis

Understand the security threats of migrating to the public cloud

Step 2: The Cloud

Determine the cloud features and their components

Step 3: Virtual Machines and the Trusted Chip

Introduce the Trusted platform module chip, Attestation and Virtual Machines. The introduction of this chip is used as the basis for the design in *Step 5*

Step 4: Migration approach

Determine what needs to be done before migration, and based on the *Step1*, and *step 2*, map the cloud features to the cloud model, and using the three fold approach come up with a migration strategy for applications based on security loopholes the cloud features may present

Step 5: Determine how best to maintain the integrity of an instance in the cloud after the migration

Step 6: Conclusion

1.6 Report Structure

This report is structured as follows.

Part I comprises the background and context that provide an overall picture of the topic of discussion. *Chapter 1* introduces the problem addressed by the thesis, demarcates it and formulates a research goal and approach. *Chapter 2* highlights the security threats of migrating to the cloud, and hints on the security area of interest

Part II comprises the description and explanation of the cloud features, which is discussed under *chapter 3*, while *Chapter 4* introduces and explains what live migration is and how they relate with virtual machines, it also explains extensively the trusted pair module concept

Part III contains what the organisation needs to do; it shows the pre-migration needs of the organisation. It is captioned as *chapter 5*

Part IV contains the main focus of this work. ‘Securely deploying applications’. *Chapter 6* consists of the activities undertaken to perform a secure deployment for applications in the cloud. It takes account of the features already mentioned in Part II, and anticipates everything that can go wrong. It shows which cloud features should be focused on and why. It explains how to use various features during the migration, and presents a new technique based on the TPM concept introduced in Part II, and the design in *Chapter 7* of how it all fits together in monitoring instances after they have been loaded into the cloud.

Part V comprises the conclusions and reflections that can be drawn from the thesis project. *Chapter 8* provides the contributions of this thesis project. It also provides the answers to the research questions and main research goal. Future work is also presented. The appendices complete the report and are given in the order in which they are introduced in the main text.

2 Security Considerations

Chapter 1 gave an Introduction to cloud computing, it defined the Problem and hinted on the research approach. This Chapter highlights the need for security in the cloud. It narrows down the security threats to a particular area of focus.

The National Institute of Standards and Technology(NIST) took a survey [72] and tried to rate challenges anticipated from the adoption of cloud computing and rated security with a 74.6% ranking, as shown in the figure below, which was higher than all other factors (including performance, availability etc)

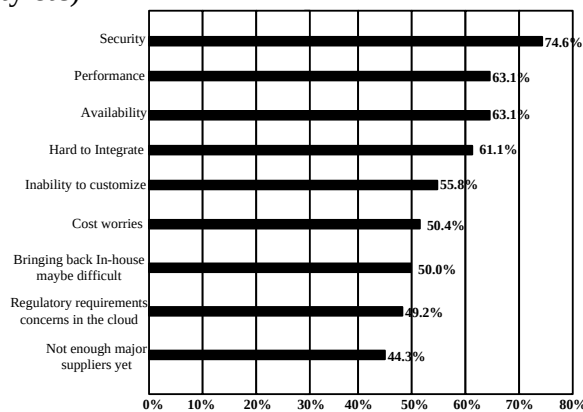


Figure 3: Challenges anticipated from adoption of cloud computing (NIST, 2009[72])

The figure above clearly shows that organizations are most worried about the surrounding of the implementation of security in cloud computing. Gartner [71], another leading information technology research and advisory company conducted an investigation regarding the information security issues that should be considered when dealing with Cloud computing. The following lists some security issues by Gartner that organizations and key decision makers, as a prerequisite, need to consider with any Cloud computing vendor [71]:

- *Privileged access*: Who has specialized or privileged access to data? Who decides about the hiring and management of administrators?
- *Regulatory compliance*: Is the cloud vendor willing to undergo external audits and/or security certifications?
- *Data location*: Does the cloud vendor allow for any control over the location of data?
- *Data segregation*: Is encryption available at all stages, and were these encryption schemes designed and tested by experienced professionals?

- *Investigative Support*: Does the vendor have the ability to investigate any inappropriate or illegal activity?
- *Data availability*: Can the cloud vendor move all their clients' data onto a different environment should the existing environment become compromised or unavailable?

The ISO 7498-2 standard [65], produced by the International Standards Organization (ISO) states that Information Security in an organization needs take both a preventive, detect and eliminate control measure to minimizing threats. In Cloud computing, the same principle can be applied, however, due to the nature of the cloud a prevent and detect mechanism might be more difficult. The Figure below shows an overview of the information security requirements coupled with the Cloud computing deployment model and delivery models set out by the International standards and organization [66]. It shows the different cloud delivery models and deployment models matched up against the information security requirements. An 'X' denotes mandatory requirements and an asterisk '*' denotes optional requirements. However, more work is needed in investigating the optimal balance required in securing Cloud computing. *Figure 4* below would be viewed in context of this thesis only as a guideline in assessing the security level.

		Cloud Delivery Models								
		Public cloud			Private Cloud			Hybrid cloud		
Information Security requirements	Authentication	X	X	*	X	X	*	*	X	*
	Authorisation	X	X	X	*	X	*	*	X	*
	Confidentiality	X	X	*	*	X	X	X	X	*
	Integrity	X	X	*	*	X	X	X	X	X
	Non-repudiation	*	X	*	*	X	*	*	*	*
	Availability	X	*	X	X	X	X	*	*	*
		IaaS	SaaS	PaaS	IaaS	SaaS	PaaS	IaaS	SaaS	PaaS

Figure 4: Cloud computing security requirements (Eloff et al [66])
X = mandatory requirements, * = optional requirements

Each of the security requirements is highlighted below in context of Cloud computing.

2.1 Identification and authentication

The Information Assurance Technology Professionals (IA) [63] states that depending on the type of cloud and the delivery model, users are established as users in the cloud based on access privileges given to them by the cloud provider. The process involves using a set of platform names and passwords issued by the provider that is unique to the enterprise, that can be used to verify individual cloud users.

2.2 Authorization

Authorization [79,81] is an important security requirement in Cloud computing since it is used to ensure integrity is maintained. Authorization exists in the back end of the cloud where the cloud facilities are; But the Information Assurance team [63] highlights that organizations are prone to significant losses from insiders with authorized access to protected information assets

2.3 Confidentiality

In Cloud computing, confidentiality [79] plays a major part especially in maintaining control over organizations' data situated across multiple distributed databases. The NIST [72,81] specifies that this is a must when employing a Public cloud due to public clouds accessibility nature. Asserting confidentiality of users' profiles and protecting their data, that is virtually accessed, allows for information security protocols to be enforced at various different layers of cloud applications.

2.4 Integrity

The integrity requirement lies in applying the due diligence within the cloud domain mainly when accessing data. i.e it poses the question of whether the data in the cloud has been tampered with or not [81]

2.5 Non-repudiation

Non-repudiation in Cloud computing is the risk that a cloud provider can refute that a particular action was or was not performed. For example the risk that an optimal cloud storage search was not performed properly. Works such as those by Jun Feng et al [116] show that non-repudiation can be obtained by applying token provisioning to data transmission within cloud applications by using digital signatures, and confirmation receipts services (i.e digital receipting of messages confirming data sent or received).

2.6 Availability

Availability is another critical information security requirements in Cloud computing. The NIST [81] defines Availability as the extent to which an organization's full set of computational resources is accessible and usable. Availability can be affected temporarily or permanently, and a loss can be partial or complete. Denial of service attacks, equipment outages, and natural disasters are all threats to availability. The concern in this work is that most downtime caused by availability is unplanned and can impact the mission of the organization.

So far, the key properties set out by the International standards organization [65], have been explored. Left out of the scene is Microsoft's security categorization of the cloud. Microsoft [82], owners of the Microsoft Azure cloud, introduced the Stride per Element Approach to threat modelling in the cloud. Their devised flow of events is shown below:

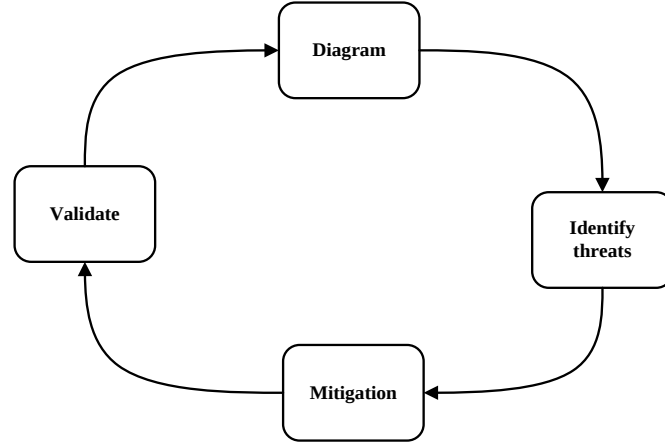


Figure 5: Microsoft’s approach to threat modelling [82]

The *Diagramming* starts the flow of events .Based on [82] the diagram event aims at using data flow diagrams that include processes, or data flows, and include trust boundaries in order to commence the model of the threat. The next element in the flow of events, *Identify threats*, is the portion that is wished to be addressed in the cloud and in this work. Microsoft classifies threats using the STRIDE approach, which is broken down into: **S**poofing, **T**ampering, **R**epudiation, **I**nformation Disclosure, **D**enial of service, **E**levation of privilege. This thesis wishes to identify the mitigation techniques in the cloud for some of these STRIDE threats. A mapping of STRIDE to the property classification by the International standards organization [65], results in the table below

Threat	Property
Spoofing	Authentication
Tampering	Integrity
Repudiation	Non repudiation
Information Disclosure	Confidentiality
Denial of service	Availability
Elevation of privilege	Authorisation

Table 1 : A Threat to property mapping [82]

The mapping is based on the fact that spoofing is the Impersonating of someone, tampering is the unauthorised modification of data in the cloud, repudiation leads to claims to not having performed an action, Information disclosure is the unauthorised exposure of information to someone not authorised to see it, Denial of service is caused by the denial or degrade of service to users, Elevation of privilege is the gain of capabilities without the proper authorisation. The security Area of interest discussed in *section 2.11* incorporates some of the STRIDE threats

2.7 Hybrid Interoperability

The hybrid cloud approach extends the private cloud model by using both local and public resources. It is used to handle unexpected loads by scaling out when the local capacity is exhausted. This is called cloudbursting [67]. Cloudbursting is an advantage because of the ability to use the private cloud and easily scale to the public cloud, but the problem with hybrid clouds is interoperability. I.e. the difficulty in interfacing different clouds .For example from the

hypervisor level up to the application programming interfaces, currently available clouds differ fundamentally [67]. At the lowest level, clouds consist of interconnected virtualized hosts using hypervisors like Xen (Amazon)[7], and Hyper-V(Azure) [7], Google on the other hand uses a different approach to its virtualisation. The hypervisor is responsible for managing the physical hardware and mapping them to several virtual machines. A virtual machine that has been created and deployed for one hypervisor might not necessarily run on a different one. Hence there is a need for a connector to act on behalf of the user to interface between different platforms. Such connectors would be known as ‘cloudconnectors’ in this work. These connectors are not provided by the cloud provider and need to be installed within the organisation that needs it. Cloud connectors already are in existence. They include Eucalyptus [68], OpenNebula [67], Citrix Open cloud bridge [69], IBMs Altocumulus [70]. Citrix Open CloudBridge and OpenNebula for example can provide an interface for the private cloud and the public cloud to behave like a single integrated network. They do this by placing the instance running in the private cloud and the instance running in the public cloud in the same subnet to enable easy cloudbursting [69]. The security area of focus of this thesis excludes the kind of cloud connector used but recommends a connector in *Chapter 6*.

This thesis narrows down the scope of security migration threats as shown in the diagram below with the broken lines:

2.9 The Private cloud: Rackspace acts as the private cloud for Exact hosting the Exact Online and Exact Identity service. Rackspace would have to interface with the cloud connector (i.e in the case of the Amazon cloud), when the need arises. This is shown in '5'. Rackspace employs Openstack⁶ virtualisation in its private cloud, this need to be taken into consideration during migration

2.10 The Users: The Users are clients of Exact that need to access Exacts application. '10' shows users accessing the Exact online software (EOL) or the Exact Identity service (EIS). Although '10' points to the private cloud, the EOL/EIS application is only hosted in the private cloud. '9' shows users accessing on premise applications.

2.11 Security Area of Interest: The security area of interest is in the public cloud, as denoted by the components within the broken line. More emphasis is laid on the assumption that mainly the data tier (This tier consists of the database servers) which contains confidential resources need to be migrated to the cloud .i.e A migration of the data tier from Rackspace to the cloud , using a cloudburst hybrid approach. Insights into security consideration is given if the application logic were to be migrated. The 'a' shows the communication from the cloud connector to an instance running in the cloud. At this point service denial attacks can take place from the cloud connector (or from Exact) to the instance, 'I' in the cloud. Service denial attacks can attack or listen for the protocol used for communication and can insert malicious traffic. 'b', 'c', 'd', 'e' show that the instance can be loaded on any datacenters (consisting of insecure machines) in the cloud. The data centres can be in different parts of the world. 'f' and 'g' show that virtual machine migration can take place inside the public cloud. This leaves room for a man-in-the middle attack [88]. Any of the datacenters can be hosting Exacts data; there is likelihood that data is tampered within the datacenters hosting it. Administrators inside the data centre can peel into the VM or even make copies of the VM's image. Attacks in the cloud can corrupt the instance running in 'I', in an attempt to steal information from the owners of the VM

Authentication is usually taken care of by the cloud provider, by allowing users create their own login access to the cloud, most of the flaws here are because these credentials are exposed to unauthorised users within the operating organisation, and as such is not be of interest in this work. Repudiation is rarely a real security threat, but is more channelled towards confirming that the cloud service performed an actual operation, this property is not be of interest. The table below shows where the security research is focused on:

⁶ Rackspace, <http://www.rackspace.com/index.php>

Security Focus Area	Mapped to STRIDE	Property
Not of interest	Spoofing	Authentication
Application Integrity	Tampering	Integrity
Not of Interest	Repudiation	Non repudiation
Information disclosure	Information Disclosure	Confidentiality
Service denial attacks, Instance corruption	Denial of service	Availability
Malicious Insiders	Elevation of privilege	Authorisation

Table 2 : Mapped 'area of Interest' to STRIDE

Part II

The Cloud Features, Virtual Machines and TPM

3 The Cloud features and Guarantees

It is essential that an organisation about to migrate to the cloud have a general idea of features that typical cloud providers employ and what each of these features offers. This chapter describes these features with their similarities and differences

3.1 The load balancing

Load balancing is the act of balancing load. In Amazon, its load balancing scheme is meant to help automatically distribute incoming application traffic across multiple instances [11] [42] . In the long run, it does this by detecting that some particular instances within a region (*a region* is an isolated location in the Amazon cloud [11]) might need to be migrated to other regions if there is too much traffic on them or within the region. In the Azure cloud, balancing of load is handled by the Azure fabric [30,83,34]. The fabric consists of users' collection of programmable Virtual machines (VM's); The programmable VM's include worker role instances and web role instances (the communication between the two instances is discussed in *chapter 6*). In Google [84,85,96], the Google Application Engine, serves a similar purpose in handling the load balancing

3.2 Auto scaling

Sanjiva Weerawarana [86] defines auto-scaling as the ability for users to scale their instances capacity up or down. Amazons auto scaling [11] fits the definition as they allow users the ability to note when a demand spike hike naturally can take place and can allow a scale up during this period to meet the demand spike and to maintain performance. Amazon also allows multiple copies of an application to run simultaneously in order to cover the volume of customer traffic [11,42]. Therefore, a user naturally only sees one Universal resource location (URL) for the application, but behind the URL there are many instances, each potentially handling customer requests. In Azure, this scaling is handled by a configuration file⁷ [30,34] that exist inside each Webrole and worker role instance that is provided. A web role instance accepts incoming traffic Hypertext Transfer Protocol (HTTP) requests via an internet information service that exists by default in the cloud, while a worker role instance cannot accept requests directly from the outside world [83]. Worker roles can their input from a web-role instance [83,34], which are mostly via a queue in windows storage. The specification placed inside the configuration file determines how instances are to scale. In Google, autoscaling is handled automatically by the application engine [85,84]

⁷ The Azure config file, <http://msdn.microsoft.com/en-us/library/ee405486.aspx>

3.3 Virtual Internet protocol addresses

In Amazon a unique type of IP address called an Elastic IP address [11] can act as a virtual IP in the cloud. Based on [42], they differ from static IP addresses because they allow users to mask instance or Zone failures (an instance corruption [77] can occur due to security lapses, a zone failure is when a machine running in an area stops functioning for unknown reasons) by programmatically remapping public IP addresses to any instance in a user's account [11]. Any Elastic IP address that is associated with the user account remains associated with that user's account until released. Users can only associate one elastic IP address with one instance at a time. When a user associates an elastic IP address with an instance, its current public IP address is released to the public IP address pool [11,42]. If the user disassociates an elastic IP address from the instance, the instance is automatically assigned a new public IP address in a short time. But this feature has a shortcoming that is elaborated in *chapter 6*. In Azure and Google however, Virtual IPs are a bit different from Elastic IPs. VIP traffic is routed through Azure's load-balancing infrastructure [30,83] and helps minimize denial of Service attacks. Google on the other hand uses a 'DoS protection service configuration' [85] in connection with an instance's virtual IP to minimize Denial of service attacks.

3.4 Virtual Private Cloud (VPC)

The Virtual private cloud [11,102] is a feature that exists only in Amazon, it is supposed to enable enterprises to connect their existing infrastructure to a set of isolated compute resources via a Private Network connection, and to extend their existing management capabilities such as security services, firewalls, and intrusion detection systems to include their AWS resources. The creation of a VPC is started by defining an IP address space. The IP addresses in this address space are private and form a network that is isolated at a packet-routing level from any other network, including the Internet.

3.5 Queues

Queues are used in all of the clouds in one way or another. In Amazon a feature called the Simple queue service (SQS) [11] is a distributed queue system that enables web service applications to quickly and reliably queue messages that one component in the application generates to be consumed by another component. The queue therefore acts as a temporary repository for messages that are awaiting processing. This enables users to decouple components of an application so they run independently, such that message management between components is eased. Any component of a distributed application can store any type of data in a fail-safe queue. The queue can also act as a buffer between the component producing and saving data, and the component receiving the data for processing. In Azure however a similar queue feature exists, and is called the Azure storage queue [34,30]. Blobs mentioned in *section 3.6*, are the form data is stored in Azure storage, but the queues' primary role is to provide a way for web role instances to communicate with Worker role instances [34,30]. For example, a user might submit a request to perform some compute-intensive task via a Web page implemented by a Windows Azure Web role. The Web role instance that receives this request can write a message into a queue describing the work to be done. A Worker role instance that's waiting on this queue can then read the message and carry out the task it specifies. Any results can be returned via another queue or handled in some other way. As an example, if a worker role needs to access a blob in order to complete a request from the Web role of a blob being written to with an HTTP GET against a Uniform Resource Identifier (URI) could resemble [30]:

http>//<storageAccount>.blob.core.windows.net/<Container>/<BlobName>

With *<storage account>* being an identifier assigned when a new storage account is created which is used to uniquely identify the blobs, *<container>* and *<Blob name>* are the names of the container and blob that the request is accessing. In Google, the App Engine controls background processing by inserting tasks into a queue called the task Queue[84,85]. The AppEngine can detect the presence of new task and can dispatch them for execution, subject to its scheduling criteria.

3.6 Storage

In Amazon the simple storage service (S3)[11,42] is the main storage for data. It provides a simple web service interface that can be used to store and retrieve any amount of data in the cloud at any time, from any location. Simple Database (SimpleDB) [11] is another service offered by Amazon that offers an alternative to using S3 which also supports relational database in the cloud. It is extensible mark-up language (XML) based, so data can be rapidly stored and easily retrieved or edited through a simple set of web service API calls. In Azure, data is stored in blobs in Azure storage [34,30]. The Blobs have associated metadata. Azure also provides a Structured Query Language database service (SQL Azure) [83,34] for relational storage. Its scope includes data insert, query, update and delete, schema creation and modification, and data access control, but SQL Azure does not support encryption of data [112]. In Google, storage is dependent on the App Engine datastore [84,85,96], which is a schemaless object datastore with a query engine, but of recent the Google App Engine provides support for SQL database[85]

4 Virtual Machines and The TPM Chip

This chapter explains what virtualization is and the components of virtualization; it also introduces the trusted platform module (TPM) chip, a hardware device that acts as the basis of the design in *chapter 7*

4.1 Virtualization

Campbell et al [6] define virtualization as the process of decoupling hardware from operating system on a physical machine. A Virtual Machine (VM) is the virtualized representation of a physical machine that is run and maintained on a host by a software virtual machine monitor or hypervisor. A hypervisor is said to be of two types⁸: Type 1 and Type 2. With Type 1 hypervisors referred to as native hypervisors which control the hardware and can monitor guest operating-systems. Type 2 hypervisors are hosted hypervisors, meaning they are software applications running within a conventional operating-system environment. Xen [7], which the Amazon cloud runs on, is an example of a Type 1 hypervisor. Xen provides full virtualization to partition the host machine into multiple VM's. Xen can also use a technique known as Para virtualization [7], where the operating system is aware of virtualization and works with the hypervisor, to improve the efficiency of operation. Hyper-V⁹, which is used by Azure, is also an example of a type 1 hypervisor. It supports isolation in terms of a partition [87]. The virtualization stack in Azure runs in the parent partition and has direct access to hardware devices. The parent partition then creates the child partitions which hosts the guest Operating systems. One concern according to Frank John Krauthheim [22] is that virtual machines vulnerabilities can be exploited to mount attacks specialized to subvert the built in defences of the guest operating systems. As all virtual machines have a standard interface and appear identical to the software running within it, all attacks can take advantage of that interface. Examples of such attacks include an adversary copying a virtual machine to maliciously run multiple copies of licensed software on multiple computers since their environments are identical.

4.2 Trusted Computing and the Trusted Platform Module (TPM)

One of the Trusted Computing Group[13,46] goal was the development of a Trusted Platform Module (TPM), a semiconductor intellectual property core or integrated circuit that conforms to the trusted platform module specification put forward by the Trusted Computing Group themselves. The Trusted Computing Group's (TCG's) specification states that Trusted computing platforms have two roots of trust, a Root of Trust for Measurement (RTM) and a Root of Trust for Reporting (RTR) [13,46]. For x86 based trusted platform onwards, the RTM is included in the BIOS boot block and the RTR needs to be in the TPM [14]. The RTM provides secure measurement of the platform and the RTR allows a verified report of the measurement through the process of attestation.

⁸<http://en.wikipedia.org/wiki/Hypervisor>

⁹<http://en.wikipedia.org/wiki/Hyper-V>

The TPM serves as the Root of Trust for Storage (RTS) providing a Storage Root Key (SRK) to protect encryption keys and other information[15,46]. The TPM also includes a Non-Volatile Random Access Memory (NVRAM) [46] for secure storage of keys and user data, and a random number generator for key and nonce generation. Measurement of the platform is accomplished by performing a Secure Hash Algorithm (SHA-1) hash on the code loaded on the platform and storing the result in the TPM’s Platform Configuration Registers (PCRs).The figure below explains more on the TPMs crypto capabilities and contents:

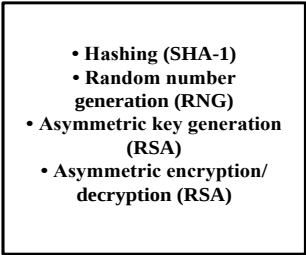


Figure 7: Crypto capabilities and contents of the TPM chip [46]

The attestation process allows clients to request a quote of the PCRs signed by the TPM and verify that the platform they are using meets their policy and configuration requirements. Clients can then determine whether they wish to use the services provided by the platform based on the attestation from the platform’s TPM. The TCG [46] explains that each TPM has an Endorsement Key (EK) and EK certificate that provides a unique identity for the TPM and validates that the TPM is legitimate. An Attestation Identity Key (AIK) is a key that contains the signature over all PCR registers; however, an AIK has no information that is uniquely identified to a single platform. This concept would be used in *chapter 7* as part of the design. The figure below shows a measurement agent querying the RTM and returning binaries to the RTR. The attestation service enables the TPM communicate with the attester [46]:

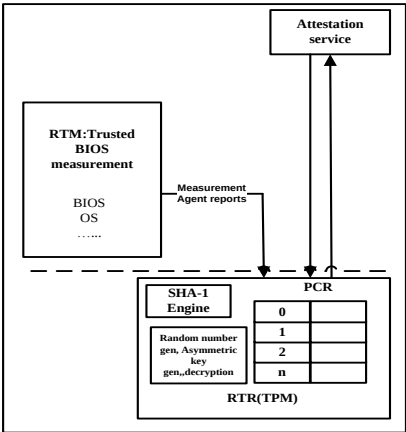


Figure 8: The communication between a TPM and an Attestation service [46]

Based on the work by the TCG [46], an instantiation of a Trusted Platform Agent needs do the following:

- 1) Obtain and supply an accurate report from the root of trust for storing and reporting integrity metrics of at least one sequence of integrity metrics in a form that prevents misrepresentation of that sequence or its source
- 2) Obtain and supply an accurate report from a Trusted Platform Measurement Store of at least one set of details describing the measurement of an integrity metric

The TPM measurements (typically, the Secure Hash Algorithm (SHA-1) of executed binaries, scripts, or configuration files used) are stored in specific TPM registers called Platform Configuration Registers (PCRs) already discussed in the second paragraph of this section. Adding a hash (m) to a PCR is called *extension* [46,47]. Where (m) stands for the binary data. It requires to use the function *TPM_Extend* (i, m) [47], which concatenates m to the current value of an i-th PCR by computing a cumulative hash. Based on these PCR values, the TPM provides a sealing functionality that binds encrypted data to the recorded configuration, and attestation, which reports the system state as recorded by the TPM to a (remote) party. For attestation, the function *TPM_Quote* [47] is employed, which presents the recorded PCR values signed by an Attestation Identity Key (AIK) of the TPM.

4.3 Virtual TPM

One problem associated with the TPM is that it works only for non-virtualized environments [48]. Most TPM implementations closely follow the TPM specification for a one to one relationship between the OS and trusted platform; therefore, the TPM has a limitation in that it can be owned by only one entity at a time. If platform virtualization is used, the TPM must be virtualized to provide full TPM functionality for each guest entity. For this reason, specifications have been developed for a virtual TPM (vTPM) [16,48,49,50,17] to provide the TPM functions for each virtual environment on the platform. There are several implementations of the VTPM; each uses different methodologies to virtualize the TPM. The technique used as part of the integrity design in *Chapter 7* incorporates the vTPM as part of its design

4.4 Virtual Machine Migration and Its Attacks

The flaws in virtual machines are inherent when the virtual machine migrates. VMs can be duplicated and migrated to help balance load. The feature that does this in the cloud is the Elastic balancer as described in *chapter3, section3.1*. This process is performed by copying or moving a VM state from one physical platform to another. Migration [78,51] poses a significant challenge to trusting a VM after migration because the migration is done while a VM is operating. Virtual machine migration commences when for example, a load balancer (*see chapter 3 for definition*) signifies that there needs to be a virtual machine migration from a machine running the VM (in this case machine 1 as seen in the figure below) to any machine that has available resources (machine 2).

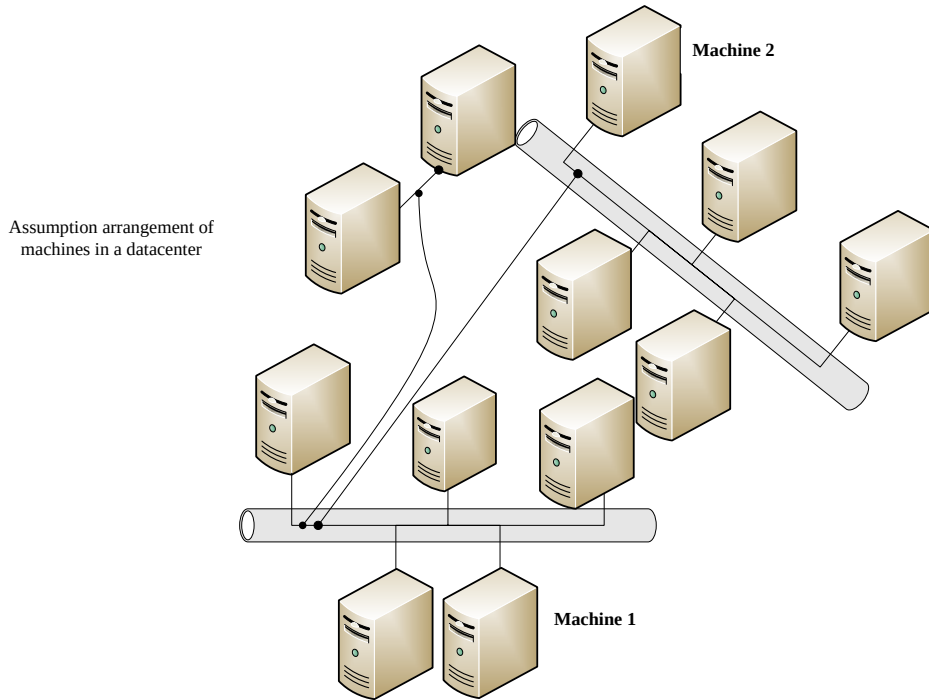


Figure 9 : Assumed distributed machine arrangement in a clouds datacenters

Based on the work by Michael R. Hines et al [104] firstly, a new Virtual machine configuration file is created on the target machine, which is machine 2, then machine 1 (being the source) initial memory state is copied to the target machine. Changed memory pages on the source virtual machine are tagged and copied to the target. The process continues until the number of changed pages is small. Then the virtual machine is paused on the source machine. The final memory state is copied from the source virtual machine to the target machine and the virtual machine is resumed on the target. But the problem now is how to trust that machine 2 doesn't compromise the users' instance, or the admin controller isn't accessing the virtual machine memory at run-time. During the migration, a man-in-the-middle attack [88] against a VM is possible but *out of scope* in this work. In this scenario, the attacker wishes to gain access to memory information of the instance. The attacker listens for provisioning or migration requests by the target and intercepts the image and state information from the source machine. This interception can be accomplished through spoofing, domain name system (DNS) poisoning [27], or any other attacks that redirect traffic from the destination machine to the attacker. Since the state information is a copy of the VM's memory and processor state, the attacker can manipulate the image and machine state by implanting malicious software [22] and then placing the compromised image on the destination server in the cloud .The figure below shows a zoom in of the Virtual machine monitor (VMM) and the VM instances on machine 1 and machine 2 and how machine 2 can be compromised by a man-in-the-middle-attack [54]

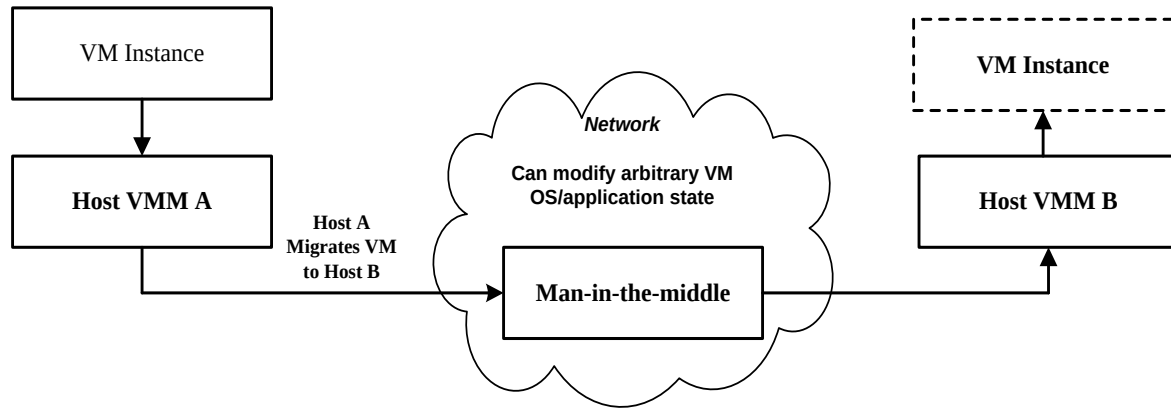


Figure 10: A man-in-the-middle attack [54]

Jon Oberheide et al [54] also states that a lack of proper access control may allow an outside attacker to arbitrarily initiate other attacks such as Incoming (i.e making guest virtual machines to be live migrated to the attacker's machine) and outgoing attacks (i.e an attacker migrating a large number of guest virtual machines to a legitimate victims virtual machine monitor causing it to overload)

Part III

Pre-Migration

5 Pre-Migration

This chapter is a discussion of what an enterprise has to keep in mind before data leaves the enterprise for the cloud. Pre-migration used in this context are a set of phases or way(s) of mapping an application used in the organisation to the cloud. The sections are based on a collection of different research papers either related directly or indirectly to Pre-migration

5.1 Identify the asset to be migrated to the Cloud

The cloud security alliance (CSA) [89] states that assets supported by the cloud fall generally into two categories:

- The Data
- Applications

The data is the raw information, and the application(s) are the logic that is dependent on the data. One way of identifying assets is to create a list of data to application mapping of the assets to be migrated

5.2 The Planning Phase

When considering migrating an application to the cloud one of the major phases would be to plan it out. Typically most organisations would make the error of focusing on only technical or implementation concerns while overlooking other stakeholder and operational concerns. In reality, they are as important as the technical concerns and must be dealt with. This statement is also supported by the work by Linthicum D. S [90]. During the planning phase, challenges such as managing the interests of some various stakeholders involved need to be considered. The answers to some of the questions by Harry Sangha¹⁰ could be used as a stakeholder organisational criterion for the planning phase:

- *Does the organization prefer low or no capital expenditure spend and rather spend more on operating expenditure (OpEx) in maintaining applications and application infrastructure?*
- *Does the organization prefer in leasing equipment or rather owning?*
- *Would there be legal consequences if an application were to reside off premises without any regulatory or geographical restrictions?*

Also, other departments that depend on the information system being migrated have to make appropriate measures to ensure their day to day operations are not hindered. It is also possible that some stakeholders could be external to the organization such as vendors to whom the operations are outsourced. At an operational level, the IT department needs to extend its view to include cloud as an extension of their on-premise infrastructure to deliver better services to business. When setting up a private cloud infrastructure within the enterprise or outside an enterprise like the way Exact does it with Rackspace for example, this could mean providing an alternative service delivery model, offering virtualized infrastructure, and associated management infrastructure to support metering, billing and service level agreement management

¹⁰ Application considerations checklist for Public Cloud Migration, see <http://www.cloudcreo.com/?p=1018>

for its enterprise applications. Similarly, when migrating to a public cloud, assessing impact on existing service level contracts and service levels have to be considered. In either case, risks can be mitigated by involving all relevant stakeholders during the planning phase. If the applications are for example outsourced and maintained by an external service provider as is with the case with Exact being dependent on Rackspace, involving them early in the process can help make the transition smoother.

5.3 Application Profiling

Hassan Chafi et al [91] defines application profiling as:

“A set of metadata elements, policies, and guidelines defined for a particular application”.

Application profiling needs to be used to measure and collect real usage data of the application(s) before it is migrated. This data helps to size the application migration to the cloud. The data collection can be done in multiple phases [91] if need be. Identification of all of the application's immediate dependencies and the identification of what other applications are dependent on the applications' dependencies. For example, application A and B need to be migrated; if both application A and application B are using the same database server (this would already have been done in the assets identification phase, *section 6.0*), this needs to be identified so that the migration plan can include a combined move or can include steps to split the dependencies. Application dependency mapping tools such as Cisco application dependency mapping service¹¹ can be used to produce server and application affinity maps, based on the observed traffic across applications, which is useful to get a sense of which applications might need to be migrated as a batch. Ideally, the application(s) data could be collected for some time to allow capture of variances in daily and weekly usage patterns. For each machine on which the application runs, useful information such as those listed below (but not limited to those listed) can be used:

- The CPU usage of the application
- The Memory usage
- Storage data the application uses
- Input/output operations per Second (IOPS)

*As a simple example, Let there be an application in an organisation which runs on an Intel Core 2 Duo E8335 with a speed of 2.54GHz, uses a memory of 3.5GB, and runs comfortably on storage data of 100GB. If the machine has an average rate of input and output operations per second (i.e based on the number of requests it sends and receives), and this application needs to be migrated to the Amazon cloud.

¹¹ Application Dependency Mapping , :see http://www.cisco.com/en/US/products/ps10437/services_segment_service_home.html

Instance Type	CPU	RAM	Storage	I/O
m1.small	1 ECU	1.7GB	160GB	Moderate
m1.large	4 ECU	7.5GB	850GB	High
c1.medium	5 ECU	1.7GB	350GB	Moderate
c1.xlarge	20 CU	7GB	1690GB	High
m2.2xlarge	13 CU	34.2GB	850GB	High
m2.4xlarge	26 CU	68.4GB	1690GB	High

Table 3 The Amazon instance types [11,42]

The Amazon instance type comparison table is shown in the table above. With one(1) Elastic compute unit (ECU) equivalent to a CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor 1[11], 2 EC's has two times that capacity. The instance type name is just the API name that Amazon uses to recognise these machines.

Therefore to *map* the example information to the Amazon cloud would mean that that organisation can go for two (2) m1.small machines. Knowing that two Machines running with memory of 1.7GB each, running each at 1.2 GHz CPU speed each equates to 3.4GB memory and can split CPU handling to run at a combined speed of 2.4 GHz, almost equating the 2.54GHz used. The storage of 160GB each is matched with the application's 100GB and a moderate I/O can equate to the applications average rate of input and output operations per second. Therefore using two (2) m1.small machines is an option. Other options also exist. Simply using one m1.large instance meets and surpasses all the criteria the application needs (4ECU, 850GB storage, High I/O acceptance). Hence an organisation can begin to map the cost of using two m1.small instance machines, against using one m1.large instance machine etc.

In addition to machine-level statistics, the usage data can also be used to build automated tests for the application to make sure of the same or an improved level of service after the application is migrated. The machine data, along with application usage data, can also provide an initial estimate of the costs of the cloud resources. After all necessary dependencies are gathered, static data can then be moved to the cloud storage. Although this movement of data is out of scope in this work, the cloud offer different wizards that can make this happen. For example: Amazon uses the Import/Export Wizard[11], with Azure users can use the Azure data sync or service bus[83] to load data, in Google the Import or Export datastore via Comma separated value(CSV)[85] files is possible.

5.4 Application Evaluation

After Application profiling, the candidate application needs to be further evaluated to make sure that it is feasible to migrate and also to prepare for the migration process itself. Linthicum [90] listed properties that can useful to evaluate an application, amongst them is: 'The tight coupling that may exist between data and application, or between applications'. The tighter the coupling between data and application(s), or between a layer(s) in the application, the more the difficulty of migrating them to the cloud. Since the majority of enterprise applications are built using multiple tiers, decoupling and re-coupling them in the cloud can be very difficult if a tight coupling exists in a particular layer(s), e.g. the logic layer. The Cloud security alliance however looks at evaluation from a different perspective. Some of the questions proposed by the Cloud security alliance as part of their evaluation phase include [89] :

-How can the organisation be harmed if the asset (data/application) became widely public and widely distributed?

- How can the organization be harmed if an employee of the cloud provider accessed the asset?
- How would the organization be harmed if the process or function failed to provide expected results?
- How would the organization be affected if the asset were unavailable for a period of time?
- Can the public cloud platform run existing on-premises applications without change?
- Does the application require high-scalability and high availability?
- Does the application require new resources to be deployed or removed easily over time as application or business needs change?

Essentially the questions posed above by the Cloud security alliance(CSA) are channeled towards assessing evaluation from a confidentiality, integrity, availability and change requirements perspective for the asset; and how those are affected if all or part of the asset(s) are migrated to the cloud .If the answers to the questions by the CSA have a more positive outcome then secure migration guarantees are higher .With the questions aside, if an application usage pattern is predictable and the kind of coupling mentioned by Linthicum [90] is well known, it would be possible to migrate application tiers, or modules within a tier, separately. For example, in a web application, static content can be migrated to a content delivery network provider to allow parts of a website to load more quickly. In other cases, bandwidth restrictions might prevent tiers from being separated, and all layers of the application are migrated to a cloud. In either case, each layer and its major distributed components modules should be evaluated separately for how it should be sized and migrated to a cloud. Another issue to consider during the evaluation phase is some applications are used from a single geographic location, while others might be used from multiple locations or even worldwide. For example, consumer-facing Internet applications often are used in multiple regions of the world in an unpredictable manner. Since enterprises also have multiple corporate locations, they might benefit from allowing applications to be distributed closer to points of access. Therefore applications need to be evaluated for their closest points of access before migration

5.5 Backup data

After the application evaluation the next step is to do a Back up [89], which refers to making copies of data so that these additional copies may be used to restore the original if data gets corrupted or lost in the deployment process or while in the cloud. Every organisation already has some kind of backup mechanism, and with migrating to the cloud it is recommended that this is done before moving data to the cloud. Backup of data includes local storage, using a remote backup storage etc. Many different techniques have been developed to optimize the backup procedure. These include optimizations for dealing with open files and live data sources such as compression, encryption, and *deduplication* (a special data compression technique) [92]. In this thesis, it is recommended that encryption be done before moving data. This would minimize the risk associated with putting control of personal or sensitive data in the hands of a third party so that its contents cannot be viewed without access to a key that can decrypt the data. NetLib [93] is recommended for encryption if using the Amazon cloud (*see chapter 6, section 6.8*)

Migration scenarios

5.6 A Full migration: A full migration to the clouds means moving all tiers (web, application and database servers) of an organisation to the cloud [94], such that all of organisations functionality now exists in the cloud, and the tiers that exist in the cloud are not dependent or communicate with any tier(s) or layer in the organisation. The advantage of using this approach is the cost effectiveness and the ease of migration since no tiers need to be communicated with in the organisation. Amazon [94] terms this approach the *forklift* or ‘pick it all up at once’ approach. Stateless applications, tightly coupled applications, or self-contained applications better suit this approach because the coupling and interaction would all exist in the cloud.

But the disadvantage is that an organisation is now fully dependent on hosting its services on platforms logical components that has no complete guarantee on the future of its services, thereby making a standby rollback mechanism a must.

5.7 Gradual Migration:

A gradual migration is more complex than a full migration. A gradual migration would entail moving a tier(s) to the cloud gradually in order to auto scale to the cloud only when the need arises. It more or less is a hybrid approach (*already discussed in chapter 2*) to migration which consists of taking some parts of an application and moving them to the cloud while leaving other parts of the application in place. The difficulty of gradual migration is dependent on the coupling of components amongst the tiers, as stated by Linthicum [90]. This is also supported by the National Institute of Standards and Technology (NIST) which stated that [81]:

“A gradual migration can only be successful if the current binding architecture of an organisation can be broken down such that well defined interfaces can be migrated once at a time to the cloud to resemble its current state”.

For example an organisation that hosts a web application wants to migrate only its database tier to the cloud. If there are static content, this can be migrated to a content delivery network provider (such as Amazon) to allow parts of the website to load more quickly, only considering that there is no tight coupling within another layer in the organisation. If it happens that only the application logic were to be moved to the cloud, a cloud like Amazon [11] provides a set of publicly available Amazon Machine Interfaces (AMIs) that are pre-bundled with SQL Server Express, Internet Information Services (IIS) and ASP.NET that an organisation (e.g. Exact runs its application on ASP.NET and IIS as a front end server) can bundle its application. A strategy suggested by Amazon [94] is to design, architect and build temporary ‘wrappers’ to enable communication between parts residing in the organisation and those that reside in the cloud. These wrappers can be made cloud-aware and asynchronous (e.g. using SQS queues described in *Chapter 3, section 3.5*) so that they are resilient to changing internet latencies.

Part IV

Secure Deployment of Applications

6 Deploying Applications Securely

This chapter discusses the migration proper, it ends with a flowchart summarising a set of steps. It contains information received from Exact and puts together a section by section technique and arguments of migrating to the cloud. The first two sections in this chapter do not belong to any specific layer in the cloud model. The first section ‘Exact Interview information’ only gives an idea of the amount and type of information available from Exact. The rest of the sections are based on the cloud model.

6.1 Exact Interview Information

Exact has gone through the pre-migration stage in the previous chapter; Exact is considering a hybrid approach to migrating to the public cloud. They have greater than 15,000 users and currently have more than three Terabyte (3TB) of database. The number of users and the size of their data keeps growing steadily.

The following are therefore noted:

- The Exact Online software is currently exposed to three countries: The Netherlands, Belgium and Luxembourg. The largest amount of users is located in The Netherlands. Belgium is only a fraction, and Luxembourg has the smallest users

- The Information technology (IT) department is the only department with access to the database both in Exact (on premise apps) and in Rackspace (EOL/EIS), but only limited people within the department have access to this data.

6.2 User Locality in the Cloud

As part of the interview information collected above, Exact is currently exposed to three countries: The Netherlands, Belgium, and Luxembourg. User locality has no place in the cloud model but is discussed here based on the Interview information given, and as part of the problem considers the consequences of further expanding its applications to more countries in the future.

In order to commence the divergence in the three fold approach, let the Amazon cloud firstly be introduced. The Amazon cloud provides the ability to place instances in multiple locations [42,11]. The control given to users to map instances in chosen locations would be used as part of the design for malicious insiders in the next chapter. Locations are composed of availability Zones as stated in [43,11], and availability zones can communicate amongst themselves. Availability zones exist in a region. Regions are dispersed and located in separate geographic areas [43,42]. Availability zones are distinct locations within a Region that are engineered to be isolated from failures in other availability zones and are supposed to provide network connectivity to other availability zones in the same Region. The idea to keep in mind during the *Transformation* is that if instances were to be placed in separate Regions, then applications can be closer to specific customers and hence legal requirements can be met, which satisfies the CSA requirement of knowing where customers’ data resides

The figure below shows a representation of Amazons Regions and Zones.

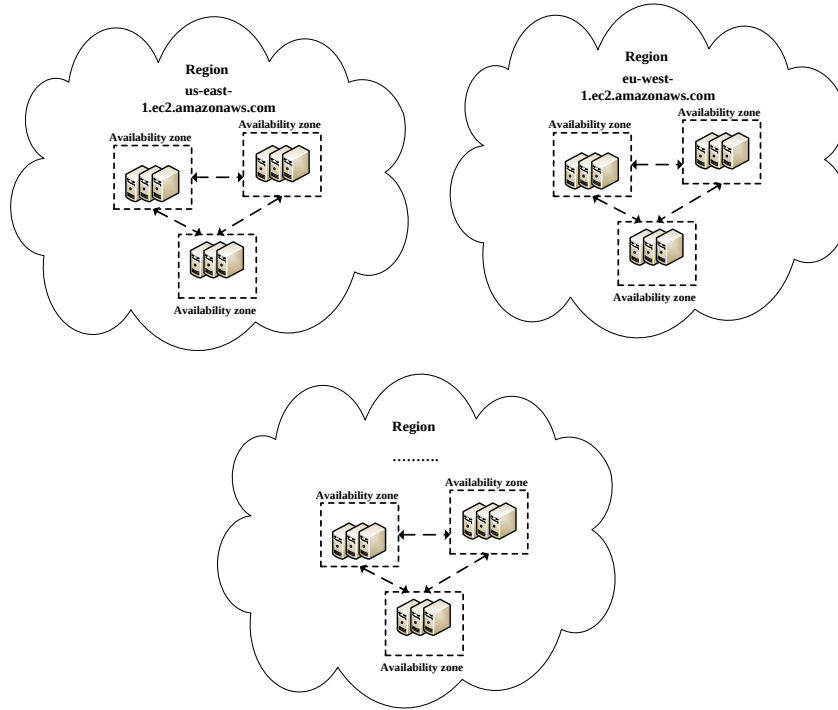


Figure 11: Region to Zone connectivity structure in the cloud [43] [11]

The `ec2.eu-west-1.amazonaws.com` and `ec2.us-west-1.amazonaws.com` shown above are endpoint calls [11,43] to the regions. A *Transformation* point to note is that all public clouds have datacenters similar to the region concept like that of Amazon, but Amazon offers users the choice of which region to place their data. The table below shows a comparison of certain features including data control in Amazon, Google and Azure [101]

	Amazon	Google	Azure
Development Environment	Simple APIs to access Virtual infrastructure and libraries	Google plug-in for Eclipse, Google AppEngine SDK	IIS/Windows based-Visual studio environment
Computing and scaling	Based on Amazon machine image (AMI) (Linux or Windows support)	Not based on a Virtual machine approach like Amazon and Microsoft: Task Queue API to schedule background work, AppEngine Cron service, quotas and limitations including CPU limits	Scales through instances of windows Virtual machines similar to Amazon, Number of Images needed is calculated through web role and worker role
Control	EC2 environment controlled by customer: Geographic location available	No control over location of data	No control over location of data

Table 4: Table of comparison, Tory Harris [101]

To map Exact to the Amazon cloud would mean that its customers in Europe needs to use the EU (Ireland) Region instance and make calls to the `ec2.eu-west-1.amazonaws.com` [43] endpoint. However, In the future if Exact has customers in the western part of America, calls to the `ec2.us-west-1.amazonaws.com` endpoint have to be made. Not mentioned is if no endpoint is explicitly specified, the US-East Region endpoint is used by default [42]. (APPENDIX 3 briefly shows how to make a change to the default region if using the command line)

Divergence aims at expanding the boundary of the design situation in order to have a large enough solution, this would mean exploring how this is done in both Azure and Google. The Azure and Google cloud don't give users the option of which region their data is to be stored or instance can be launched, based on Torry Harris [101], hence the location to user mapping cannot affect the deployment if either of these clouds were chosen. This can pose a very serious issue for clients that need their data to meet region or locality requirements

Transformation: Based on the divergence, it can be seen that mapping users to particular regions is dependent on how the cloud operates. If the cloud has no guarantee of where users' data is placed a mapping of user to location is meaningless. No general design can be made from the divergence as the mapping is either supported or not supported by the cloud infrastructure. Therefore the idea is to ensure that locality guarantees exist or take the risk and use the cloud knowing that data locality problems can be an issue in the future. If the cloud can support cloud-data to region mapping, then a customer to location structure is useful. The table below shows an example of what the customer to location structure of Exact could resemble.

Customer	Location
Customer 1	Belgium
Customer 2	Netherlands
Customer 3	Belgium
Customer 4	Luxembourg
Customer 5	Netherlands
Customer 6	Netherlands
Customer 7	Netherlands
Customer n	

Table 5: Customer to location of Exact users

6.3 The Integration and Middleware layer

In this layer messaging services such as queues exist. The Problem in this layer is that there is no single way of handling queues in the cloud. The *divergence* starts with the Amazon cloud. The idea behind this layer is to find the best way of handling queuing features so as to minimize instance failure and at the same time increase efficiency. In the future there is a possibility that Exact might use an application that places its requests on queues in the enterprise. A function known as the simple queue service (SQS)[11,95], which was briefly introduced in *chapter 3* exists in Amazon. As explained, SQS uses a message queuing service that enables asynchronous message-based communication between distributed components of an application. This communication is depicted below¹² :

¹² see Amazon simple queue service, <http://aws.amazon.com/sqs/>

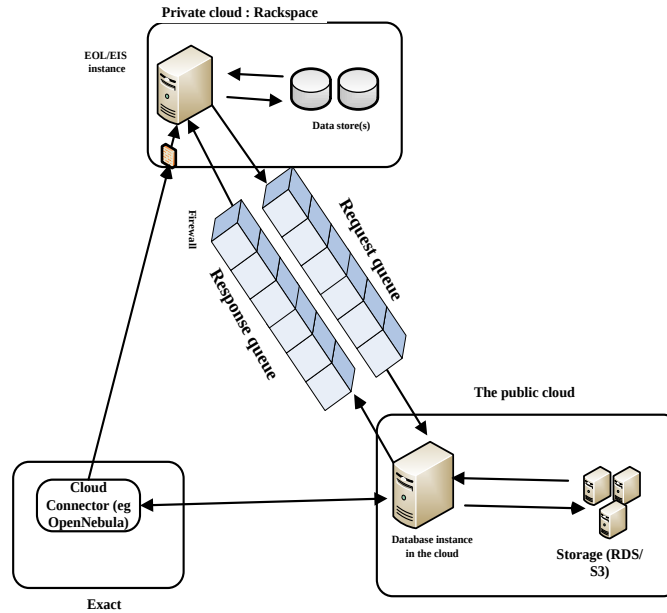


Figure 12: A Queue as a communication path in the Cloud [11,95]

The assumption is that in the private cloud Open Nebula [67] is used as the cloud connector inside Exact to act as an interface between the EOL/EIS instance in the private cloud and the database instance in the public cloud. This setup can handle applications that query the database very extensively. But if the private cloud and the public cloud were using the same virtualisation (e.g. a private cloud running on Xen virtualisation (Amazon) and a public cloud running on Xen virtualisation (Amazon)), then a direct call would be possible and there is no need for a cloud connector. The asynchronous message-based communication is such that the queue can both act as a response queue, and as a request queue. Such that every user request results in a message being queued into the request queue. The message in the queue as explained in [95] can then contain the application processing operation to be performed. The Server, running in an EC2 database instance, reads messages from the queue, and processes the request. Messages that are stored in the SQS queue remain in the queue until explicitly deleted [95]. From a *Transformation* view it's logical to see that this can handle an instance corruption Problem, stated as one of the security areas of interest in *Chapter 2 section 2.11*) This is because since messages remain in the queue until deleted if an instance failure occurs, it can easily recover from it. The *Google cloud* uses a Task Queue [85,84]. It allows for defining any number of individual queues by providing a queue name. The task queue uses token buckets to control the rate of task execution. Each named queue has a token bucket that holds a certain number of tokens. Therefore each time an application executes a task, it is said to use a token [84,85]. An application continues processing tasks in the queue until the queue's bucket runs out of tokens. The AppEngine refills the bucket with new tokens continuously based on the 'rate' that is specified for the queue.

Based on this, a *Transformation* is made to serve as part of the solution: Instead of allowing SQS to act as a backup start up for an instance that gets corrupted, another instance can act as a preventer of complete processing (instance) failure in the cloud. During the period when an EOL application instance crashes, before its recovery users notice a delay [95] if they were formally processing data on it. A preventive design is based on the knowledge exposed in the *divergence*. If another instance(s)

were introduced to bolster the instance while it processes, if the instance currently handling the load crashes another instance can continue.

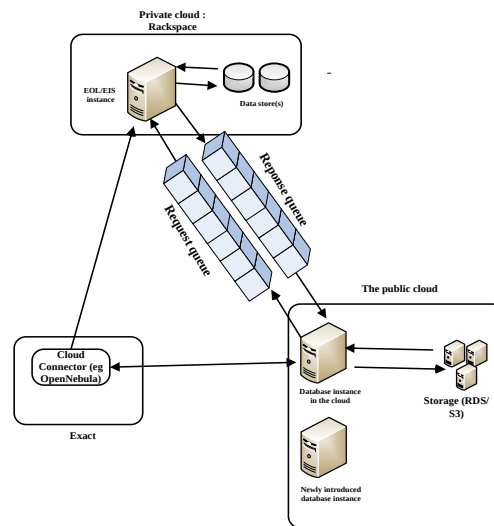


Figure 13: A Transformation: Introduce a new instance

The newly introduced queue can act as a bolster to the other instance. Another backup instance design option might be to introduce another SQS queue (but this would have to be paid for) and attach the already existing database instance as the input to the queue, and a new database instance to the receiving end of the queue.

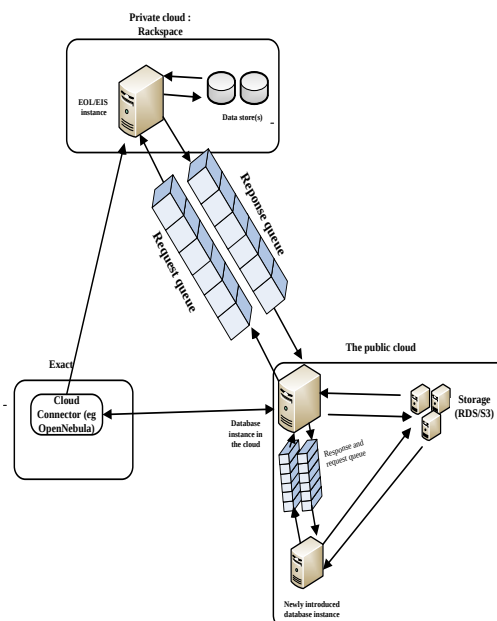


Figure 14: Potential back up design for the instance

The new database instance can act as both a support for the existing instance in case of a failure and might be able to help load balance. With the *Google* cloud, there has to be some sort of

mapping between the bucket size and rate based on that applications needs and resources. A mapping can make a difference since the AppEngine refills the bucket with new tokens continuously based on the 'rate' that is specified for the queue, it can somehow lead to bursts of processing and consumption of system resources that could be competing with user-serving requests.

6.4 The Core connectivity and Delivery layer

Any platform as a service provider (for example Google and Azure) gives users a platform to access its data by a programmer. The problem in this layer is that potential cloud users such as Exact are exposed to certain harmful protocols by default. *The divergence* into connectivity and delivery commences with the Google and Azure clouds .The Google cloud allows a *URL fetch* service [84,96,85] to access resources over the web, and to communicate with other hosts using HTTP by default. In Azure however there are three options of how to access the data in Azure storage:

The first option is to have a single key for the storage account and allow users access. The problem with using this as stated in [30,34] is that anyone who has this key can do whatever they want to the data stored, and as such is not a recommended option. The second option is that users can read and access the data (blob) in a public container. The problem with this is that anyone can read the blob or data, and setting security privileges to every blob or data and associating it to a user account would be problematic and can be prone to errors. The third option is with the use of Shared Access Signature [34,30] . Shared access signatures are recommended in this work because they can use tokens to give a set of rights against a blob or blob container that would not be set to public. The only problem is that the applications are responsible for generating the tokens, and hence are also responsible for their security. The default token state [34,30,83] for URLs to access resources is the HTTP. A transformation point to note is that there is a common default fetch service between Google and Azure: Shared access signatures in Azure by default uses HTTP, the URL fetch service in Google uses HTTP by default.

The Transformation: HTTP is not secure either in its use in the Google cloud or in Azure because snooping [97] can occur during the URL request. The figure below shows a picture of a user accessing Exact.com being exposed to a snooping attack based on the work by Timothy D. Morgan [97]

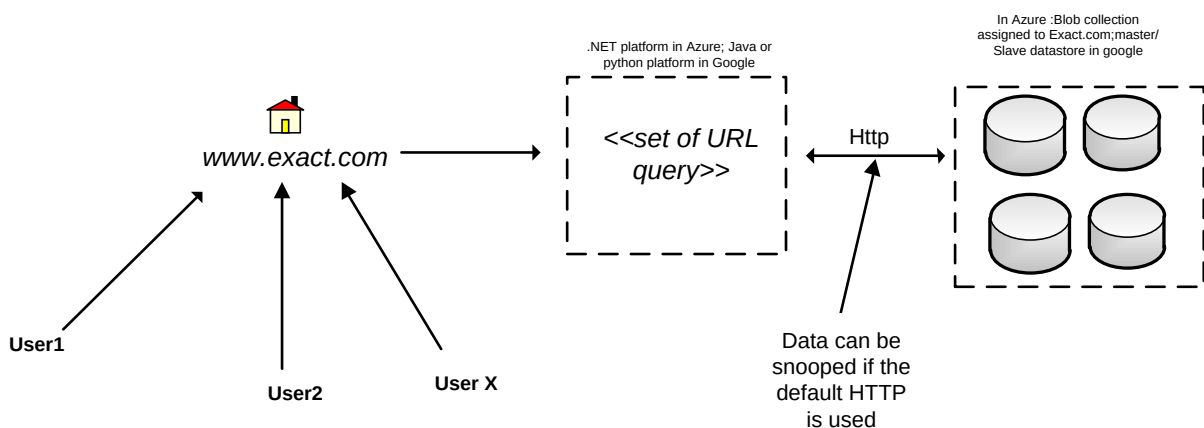


Figure 15: Exposure to snooping attacks [97]

The figure above shows the possibilities of a snooping attack occurring. Other kinds of attacks that can occur are the HTTP request smuggling [97]. HTTP request smuggling is a general class of attacks which is used to either poison web caches or hijack user credentials through proxies. Therefore HTTPS must always be used, as it's more secure and more trusted than HTTP. This also needs to be applied to the Google cloud or to any platform as a service cloud that needs to use URL to access resources. Also to minimize the threats of these kinds of attacks the use of the most restrictive Access control list (ACL) mapped to specific users would make a big difference. For example such mapping of users to data in the table below is one scenario

User	Access privileges
User 1	Only allowed to access certain blobs
User 2	Can be given a timeout period when access to a blob expires
User 3	Can be given full access to blobs

Table 6: User to privilege mapping to blobs

By applying the kind of mapping in the table to the codes, users can get restricted to certain resources. Users that no longer need access to the Exact application need to be removed, so that a user such as User 3 does not continuously have full access even after they don't exist in the system. The table above is equally supported with the *Principle of least privilege*¹³ which requires that in a particular abstraction layer of a computing environment, every module (such as a process, a user or a program on the basis of the layer being considered) must be able to access only such information and resources that are necessary for its legitimate purpose

6.5 Abstraction layer

In this layer, the instances and the instances' security state exists. The Problem in this layer is that there are a varieties of things that can go wrong with an instances' state in the public cloud. Chapter 4 briefly highlighted live Migration as an example. Stuart hacking et al [51] showed what typically occurs in a live migration scenario, where the VM monitor copies all the VM memory and then begins to re-send pages which have been re-written. He also showed the possibility of the VM detecting that the memory pages are being re-written faster than it can send; once this happens the virtual machine monitor freezes the VM and send the final changed memory and system state. According to [51] this act of re-writing memory is known as page dirtying. With this it means that for large applications or applications which are memory intensive it can be the case that the monitor cannot keep up with the rate of page dirtying. The Virtual machine can get frozen too early and for an extended period of time. This period of inactivity often results in errors such as network timeouts and memory corruption [51]. In serious cases it can crash the application or server. Therefore if a large database instance is running lots of requests there is more likely to be have VM failure if there is going to be a live migration. Hence a *divergence* into how the instance size is handled in the various clouds are looked into.

In Amazon, the choice of instance varies with the way size and the speed of the computation of the enterprises application in question. Users are free to choose any instance they want. Amazon matches its instance size to CPU as the table shows below [42,11]

¹³ see http://en.wikipedia.org/wiki/Principle_of_least_privilege

Instance Type	CPU	RAM	Storage	Platform	I/O
m1.small	1 ECU	1.7GB	160GB	32-bit	Moderate
m1.large	4 ECU	7.5GB	850GB	64-bit	High
c1.medium	5 ECU	1.7GB	350GB	32-bit	Moderate
c1.xlarge	20 ECU	7GB	1690GB	64-bit	High
m2.2xlarge	13 ECU	34.2GB	850GB	64-bit	High
m2.4xlarge	26 ECU	68.4GB	1690GB	64-bit	High

Table 7: The Amazon instance types [11,42]

In both Azure and the Google cloud however, they are PaaS providers, and so the concept of instance size is different. Azure¹⁴ allows users to set the size of the instance either programmable or by manually changing a config file from small to medium to large. But, the recommendation still remains to start from small and auto-scale because the hyper-V operation is very similar to the way the Xen Hypervisor operates. Azure also does VM migration¹⁵ and as such is open to the same instance corruption; But this is not the case for the Google cloud which is automatically handled [85] by the Application Engine. Users have no config file to determine the size of an instance. Based on Google's documentation [85], the Google Application Engine monitors the number of requests waiting in each instance's queue, and doesn't use the instances size. If it detects that queues for an application are getting too long due to increased load, it automatically creates a new instance of the application to handle that load, without regard to its size. It scales instances in reverse when request volumes decrease. In this way, App Engine ensures that all application's current instances are being used to optimal efficiency, but there is no way of telling if and when an instance can get corrupted in Google.

The Convergence here is if using an IaaS like Amazon the choice of either the m1.small or c1.medium instance suits Stuart hacking et al [51] recommendation, and if using Azure the small or medium option needs to be chosen in the configuration file.

Backing up Instances

It's possible that in the cloud an instance can fail¹⁶ for reasons such as administrator errors from the machine the virtual machine is running on or from unknown errors (too much traffic etc.) that could lead to an instance crashing. *The convergence* starts from the Amazon cloud. There is a feature in the Amazon cloud called the Elastic block store [11,42]. EBS is briefly enumerated in *chapter 3*. If for example an database instance experiences failure of any sort, an EBS volume can automatically detach with the data intact. It can then be possible to reattach this EBS volume to a new instance and quickly recover from this failure, as shown below [11]

¹⁴ Azure Config file, <http://msdn.microsoft.com/en-us/library/ee405486.aspx>

¹⁵ VM migration in Azure, <http://www.hyper-v.nu/blogs/hans/?tag=live-migration>

¹⁶ EC2 Instance failure: <http://blog.rightscale.com/2008/02/02/top-reasons-amazon-ec2-instances-disappear/>

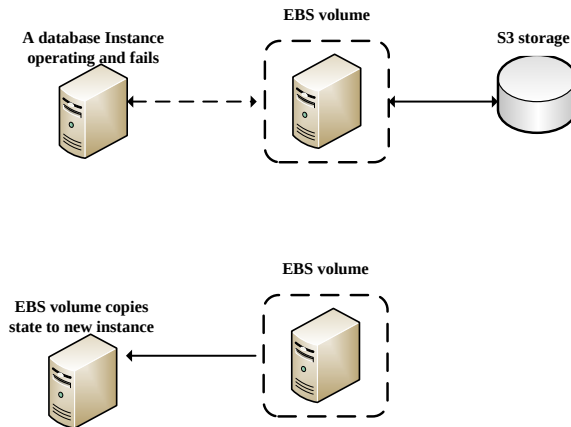


Figure 16: EBS launch[11]

The Figure above shows an instance that is operating and fails for some unknown reason. The second figure shows the EBS volume copying its state after a new instance is launched. In summary, to recover, the EBS volume is detached from the instance, launches a new instance, and attaches the EBS volume to the new instance. In *Azure* however there is a different approach to recovering from failure. Failure recovery is dependent on the programmer. Azure allows up to five¹⁷ storage accounts [83] to be assigned to one Azure subscription, and each account can accommodate up to 100 terabytes of data. Since there are a limitless number of web roles to worker role communication, any kind of architecture that works for the user can be used. By default, role instances deployed to Windows Azure run under full trust [30,83]. But there is another option of Partial trust. Partial trust delineates a restricted Code access security (CAS) environment, i.e it can allow programmers to restrict users to certain resources [83]. An approach would be to use the simplest communication connection recommended by Azure, which is the use of full trust with an instance acting as a web-role directly connected to a worker role which is connected to the storage. The problem with this as shown in the figure below is that if the worker role instance is jeopardized for some unknown reason, all storage can get corrupted, and users won't be completely utilising the advantage of being able to have separate storage accounts

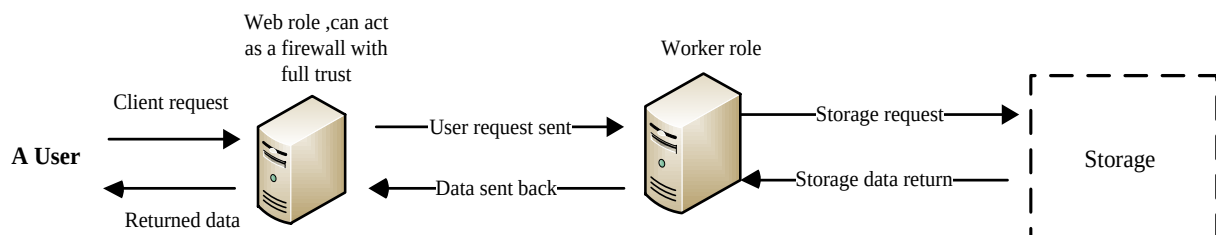


Figure 17: Webrole to worker role communication [83]

¹⁷ Azure storage : <http://blogs.msdn.com/b/jimoneil/archive/2010/08/06/azure-home-part-3-azure-storage.aspx>

A *transformation* approach for the Azure cloud would then be to utilize these storage accounts and make instances point to them in code. This creates a firewall with web role instances and filter with unknown or untrusted requests to a different storage account .The design below shows this:

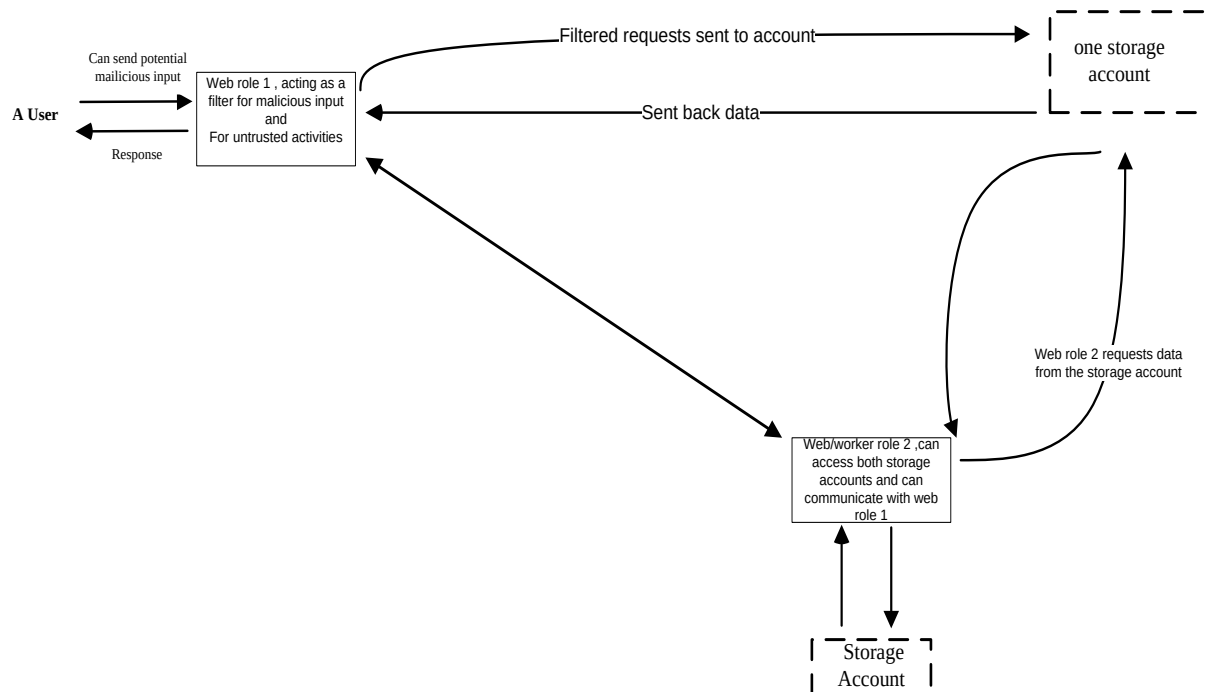


Figure 18: A plausible design in the Azure cloud

*Let there be two different storage accounts accessible by two different Webrole instances as shown above. In the figure above, there is now an instance that can receive untrusted requests, called web role 1 which has access to only one storage account, and should not be trusted to do anything but send incoming requests and can log them to that storage account. Another web role instance, web role 2, can then be able to access both its own storage account and the account of web role 1, but the setup is not exposed to potentially-malicious, inputs. Web role 2 can therefore process and filter requests from the untrusted role via web role 1's storage account, Webrole 2 therefore shouldn't fully trust that storage, therefore allowing for the safety of the bulk data stored in the storage only accessible to web role 2. Therefore if there is any compromise whatsoever, only the storage account accessible to Webrole 2 gets compromised.

In the Google cloud however, there are no worker roles or web role instances. But there is a feature that might be helpful. The feature is the 'Always on'[85] .Always on is a premium feature which reserves three instances of an application, and never turns them off, even if the application has no traffic at the time, it's always on stand by. This can solve the backup of the instance that is needed. This feature is recommended here in order to mitigate the impact of loading requests on applications that have small or variable amounts of traffic. Also in the event that an 'always on' instance dies accidentally, the Application Engine automatically restarts the instance [85]

Therefore, If using the Amazon cloud, back up instances with the EBS. If using the Google cloud uses the Always on feature, but if using Azure, maximize the limit of storage accounts to

create a firewall-like back up in case the instance gets corrupted as shown in the proposed design in figure18

6.6 A firewall for Hybrid migration (Abstraction layer)

This section simply discusses the problem of virtual firewalling instances in the cloud. It takes into consideration the fact that Exact has both data and EOL application in Rackspace, which is a private cloud and will need to cloud burst into the public cloud.

A *convergence* into how Amazon secures each instance in the cloud is by using a security group feature [11] such that a group for the web servers would have port 80 (HTTP) and or port 443 (HTTPS) open to the Internet. A group for application servers would have port 8000 (application specific) accessible only to the web server group. A group for the database servers would have port 1433 (MS SQL) open. But this setup is for a full migration to the cloud and only applies if all the tiers are to exist in the cloud [11]. If considering a hybrid approach where only the data is migrated to the public cloud, there needs to be a database instance in the public cloud that would cloudburst from the private cloud (Rackspace) when necessary. Rackspace uses the openstack [98] operating system as its virtualisation. The Amazon cloud uses Xen. The job of a cloud connector is to make the application instance running in Rackspace see the database instance running in the public cloud as being in the same subnet, so that a cloudburst can take place. Since the MS SQL port number is 1433, this needs to be left open to the EIS/EOL instance running in the private cloud to cloudburst. This is shown in the figure below.

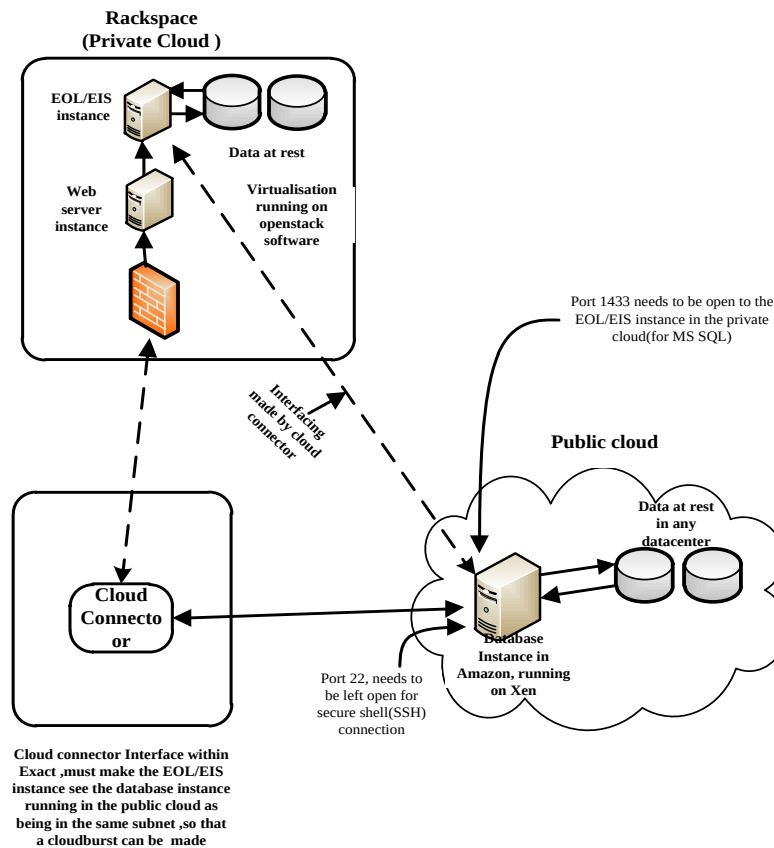


Figure 19: Firewall connection from the private to the public cloud based on the work by Dr. Martin Schader [99]

The cloud connector is the interface that would connect both clouds, making the instance running in the cloud seem like it's in the private cloud. Therefore users that want to access the EOL/EIS application stills need to go through the firewall of Rackspace, the application logic takes place within Rackspace, but some parts of the data would reside in the public cloud. The fetching of the data needs to be done in the public cloud. In the event that there needs to be a Secure Shell Protocol(SSH) connection from Exact into the public cloud, *Port 22* (SSH port number) needs to be left open on the database instance.

The recommended cloud Connector to use for this is OpenNebula [67]. The recommendation is based on the cloud connectors' ability to introduce its own proprietary infrastructure that can support different clouds [99] (for example vSphere (VMware Infrastructure)). Other cloud connectors like Eucalyptus, Nimbus either introduce or emulate others [99]

Dr. Martin Schader [99] states that OpenNebula:

“Differentiates itself from the other platforms in the sense that it has been designed to federate existing technologies; It has a flexible component-based structure and includes several drivers for information management (monitoring), image and storage management “

OpenNebula supports both Xen and OpenStack [99] .Therefore with its support for EC2, OpenNebula can be configured to use Amazon's infrastructure for cloud bursting from Rackspace or use other EC2 compatible systems (such as Eucalyptus or Nimbus).

Currently no cloud connector supports an Azure to Rackspace connection or a Google to Rackspace connection for cloudbursting, but a new EU-funded VENUS-C project¹⁸, is aimed at developing, testing and deploying interoperability between Windows Azure and OpenNebula. Therefore the most visible approach to migrating to the Google or Azure cloud would be to only run instances in the public cloud when the need arises, but the data in the cloud needs to be an up to date version. One approach in *Azure* would be to put an 'active dataset' in Azure storage , but keep the full master repository on premise and by applying a service such as the Azure Service bus[30,34,100] or the Azure data sync[30,100]to keep the data up to date. The figure below shows this:

¹⁸ VENUS-C project: <http://blog.opennebula.org/?author=5&paged=2>

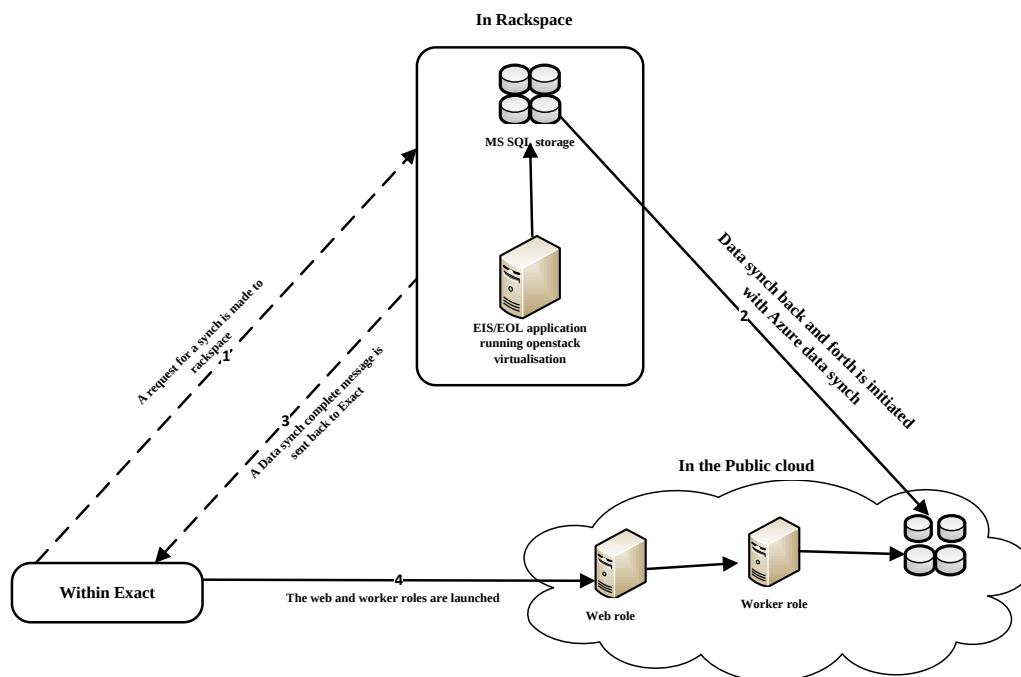


Figure 20: An hybrid approach for connectivity in Azure

Since Azure data synchronisation can provide bi-directional data synchronization and data management capabilities, it can be used to share data between multiple SQL Azure databases and between on-premises and SQL Azure databases in the cloud. It is to be noted that the firewall design in Section 6.5.2 can be used here, to bolster the setup in *figure 20*. In Google however, the data synchronisation feature doesn't exist. Instead, there is a possibility to use the bulk loader [85] to either upload or download data to the datastore. The only criteria is that the data uploaded to storage must be in a .CSV format [85]

6.7 Migrating Internal applications

The discussion in this section solves the Problem of migrating internal applications within Exact to the cloud while trying to maintain the same amount of segregated control that exists in the organisation. For example Exact Synergy is a web application that is used only by Exact, while Exact Globe is an on premise application not accessible to the public. There is a possibility in the near future to migrate data for both of the above mentioned applications to the cloud. The discussion will only consider it from an Amazon perspective as the Information given in *section 6.6* applies to Azure and Google. The only difference is data synchronisation is done from within Exact, since that database resides there.

The problem with migrating internal applications is the ability to maintain the same amount of control and security privileges within an organisation as in the cloud. If Using the Amazon cloud, one way to do this if not running all the applications in the cloud is to run the on premise applications within the internal network[11,42], but only the database is hosted inside the public cloud, and as such only the database instance exists in the cloud as shown below

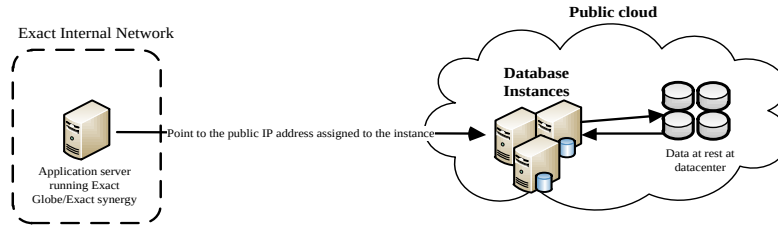


Figure 21: The Instance connection setup [11]

Here, *no cloud connector* would be needed since Exact isn't running any virtualisation software. But the application running the on-premise applications in Exact must be made to point to the public-IP address assigned to the database instance launched in the cloud. Also, port 1433 needs to be left open to communicate to the application servers. With the setup above, the Secure Shell (SSH) port needs to be left open so that internal users within Exact can access the data internally. *But in* a scenario where there are different internal data belonging to different applications that should be migrated to the cloud, it can become very untidy to manage control to these instances as users in Exact could have different kinds of security accesses to different applications. A better and more secure approach would be to use a private virtual feature, such as the virtual private cloud (VPC) [102] [11]

Since VPC is based on a Virtual private network connection established from Exact to the cloud provider, users within Exact can launch instances into the VPC's subnets [102,11]; based on [102]with the appropriate security policy, these instances appear on the enterprises' existing network. VPC traffic bound for the Internet can be routed over the VPN to Exacts network, where it can be examined by pre-existing network security services, such as their firewalls and intrusion detection systems, before exiting their network perimeter to the Internet.

The figure below shows one solution with using a VPC with two subnets. Each subnet represent an set of instances connected together that meet security requirements for a single applications database and is only accessible to users in Exact that meet that subnets' requirements (which is basically users that have access to a particular IP range that corresponds to the subnet)

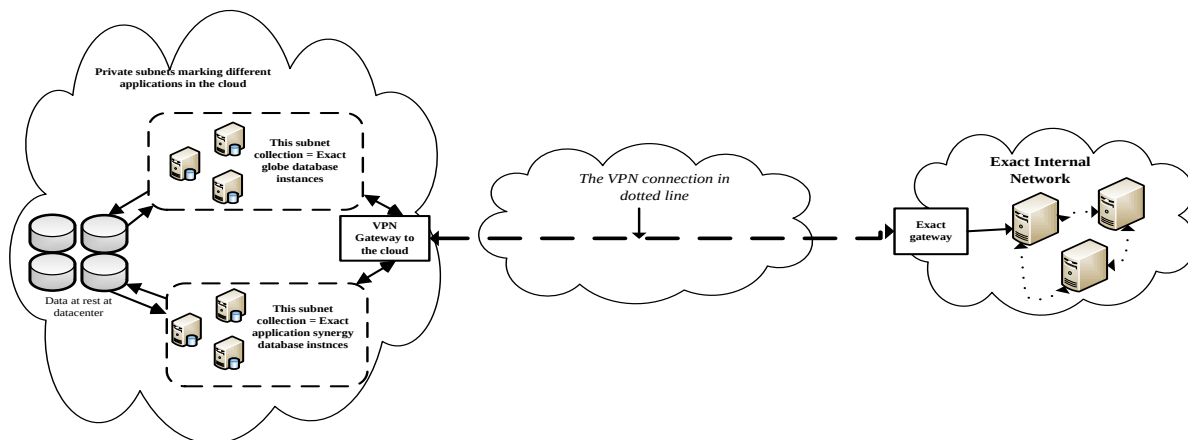


Figure 22: Safer strategy for migrating internal applications using a Private virtual feature [11]

The VPN gateway can be seen on the cloud side of the VPN connection, and the customer gateway can be seen on the customer side of the VPN connection, which in this case is Exact. Since a VPC is created by first defining its IP address space, the IP addresses in this address

space are private (match that from Exact) and form a subnet that is isolated at a packet-routing level from any other subnet. This makes it easier to meet *security requirements* with internal applications' data. For example, with this setup Exact globe, and the Exact synergy can be launched such that only users that match the IP address of the subnets are allowed to access the data; This lets separate the isolated resources in the VPC based on security and operational requirements.

6.8 The Hardware layer

Confidentiality of data needs to be achieved in the public cloud. So far this thesis has hinted on the need for encryption in the cloud, but hasn't recommended any encrypting software as of yet. The problem is that there are different varieties of encryption software. Exact has over 3TB of data, and the size of the database keeps growing. The choice of any encryption software must be able to match the size of their database, and should give different flexible options of storing and sending the decryption keys to the public cloud.

A *convergence* starting from Amazon shows that decryption keys can be passed to an instance before it is launched [11]. NetLib [93] is recommended for securing storage in and out of the cloud. The recommendation is based on NetLib's ability to achieve whole database encryption on the fly [93]. i.e it supports dynamic encryption and decryption. Data is never decrypted on disk, but only in memory as requested by the MS SQL Server. NetLib equally has support for a wide range of algorithms, they include: Advanced Encryption Standard (AES) with up to 256 bits, Data Encryption Standard (DES), Triple DES (3DES), Triple DES (3DES), Blowfish (up to 256 bits). The table below shows a comparison of NetLib supported features and the Encrypting File System (EFS) [93]

Supported feature	NetLib	EFS	Comparison view
Operating Systems supported	WinNT, Win2K, WinXP, Win95, Win98, WinME	Only supports Win2K, WinXP	NetLib supports more operating systems
Files copied to non-NTFS media can be automatically encrypted	Yes	No	Backups can be automatically encrypted with NetLib even if a non-NTFS media were used
Can integrate into an install script	Yes	No	Easy to bundle with an application. EFS must be deployed by end user.
Can be used to encrypt individual columns	Yes	No	The column encryption option needs to be on for NetLib to support individual column encryption

Table 8: Comparison of NetLib and Encrypting File System (EFS) [93]

6.8.1 Performance and Key handling: NetLib

Decryption keys from NetLib can be sent to an Instance's boot partition (in Amazon for example) only when the Instance needs to Start-up, so that it can decrypt the encrypted files in the cloud, as shown in the *figure* below. With respect to performance, NetLib allows whole database encryption, regardless of the size of the database and shows negligible impact in most multiprocessor servers, and for very large data sets its performance still remains negligibly the same [93]. So for a 3TB database, such as that of Exact, it should not be a problem. It leaves very little *Memory footprints* (i.e it takes up small memory), and uses a small sized non paged portion. The Data keys used for encryption and decryption can be stored flexibly with a variety of methods. These methods include [93]: using a strongly encrypted file (called a *profile*) on the local drive, on CD, or USB drive (but user need to insert the profile file wherever it may be to start the MS SQL Server, the disk can be removed once started), Keys can also be stored in a profile on a remote machine (referred to as a proxy location). The disadvantage of this is if the proxy machine is not found, the SQL Server cannot be started. Keys can be embedded into an application with an API call if necessary

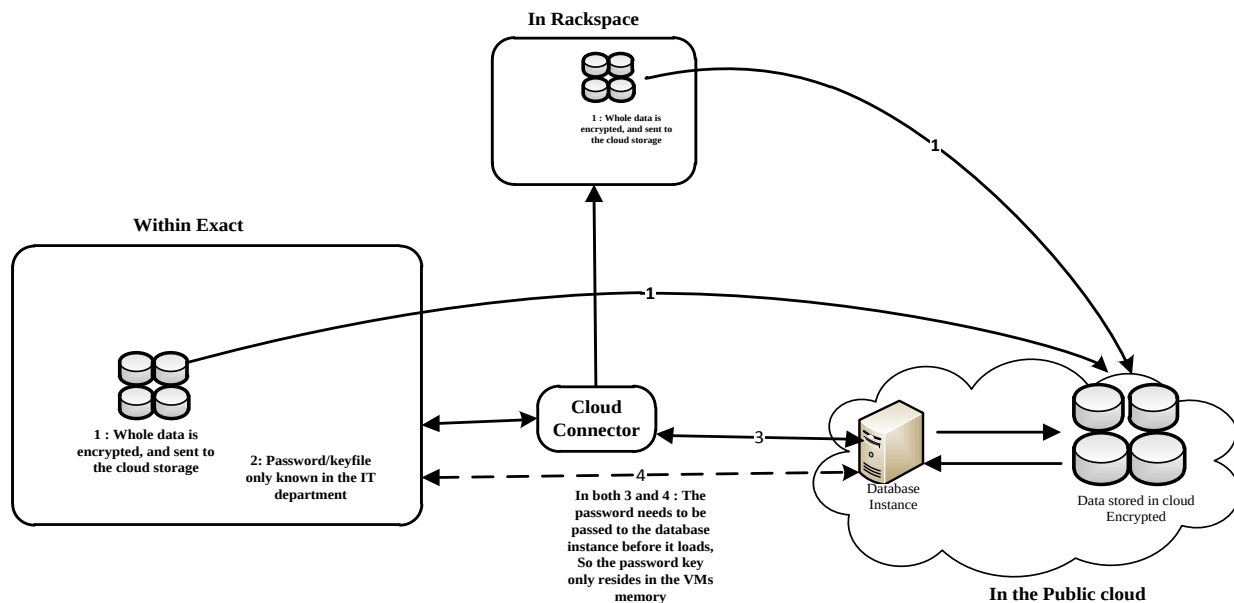


Figure 23: Passing decryption keys to a database instance

Azure on the other hand allows up to five storage accounts to be assigned to one Azure subscription [83]. Azure SQL is similar to S3 in Amazon, but it does not support encryption of data yet [112]. The Azure storage supports encryption of data, but only allows non relational data, meaning that data to be sent to the Azure storage has to be in non relational form. With encrypting data in the Azure cloud, there is a core .NET library which allows developers to integrate a .NET Cryptographic Service Providers (CSPs) within Windows Azure [33]. Therefore a developer can use this .NET CSPs to easily implement encryption, hashing, and key management functionality for stored or transmitted data. Microsoft [83] further states that there is an:

"RNG Crypto Service Provider class which can be used to generate random numbers sufficient to seed the high level of entropy required for strong cryptography"

This CSP exposes users to an array of cryptographic hash functionality including MD5 and SHA-2 to verify data correctness, create and validate digital signatures, and create non-identifiable tokens in place of sensitive data. The next PaaS to *converge* to is the Google cloud. In Google, the AppEngine includes a custom version of the Python Cryptography Toolkit, also known as *PyCrypto* [85], which can be called to do the encryption and decryption. *Keyczar toolkit* [85] can be used for encryption and decryption if using the Java platform

The Transformation: The *transformation* here is to note that since Google gives each user ten (10) applications for free [85], there is an ability to leverage these 10 applications in any way and use this advantageously to secure data. It is possible to separate the data in the cloud into sensitive portions. For example store some data in one Google AppEngine application with some other data in another Google AppEngine then combine those two data sources programmatically (e.g. using an Rich Internet Application (RIA) client) or any other method suitable). Therefore anyone who manages to gain access to one of the data stores would not have complete information. One data source can have some details of the data, while another data source can have other details. The program environment needs to know how to use encryption on the data elements to tie the two data sources together. In Amazon separating data like this would not be possible because there is no way of calling the different data sources. Also, there is a problem of where to store the encryption keys in *Google AppEngine*. One way of storing the encryption keys is in a key store file [85] (a database file that can contain both public keys and private keys.) Any Intruder trying to get at the key needs to know the location of the key store, the key store's password, the alias name for the stored encrypted key, as all these attributes are needed to access a key store file. To make it a bit harder, it might be better to try to spread around the key store name, its password, the alias, etc. between configuration files and source code.

In *Azure* the same problem lies in key storage and persistence. But based on the work by Richard J. Dudley et al[103], the Azure Storage services can be utilised as a storage key library. Cryptographic keys could be stored in a text file through the blob service API in a *key container*[103] (i.e storing ones own library in within the windows storage services, since only the users that have the storage keys can access the blob service assigned), this is based on the fact that each CSP has a key database[85] that can be used to store persistent cryptographic keys. Each key database contains one or more key containers, each of which contains all the key pairs belonging to a specific user. Azure gives each key container a unique name, which applications provide to the CryptoAPI function when acquiring the context to the key container, and so this might be a good place to store the keys.

In the Event that a PaaS provider does not provide an encryption algorithm it is recommended that triple DES(3DES) be used based on Microsoft's advice[115], which banned the use of the DES encryption algorithm [113], because it has been broken [114], and there have been no proven attacks on 3DES as yet

In order to protect cryptography keys, it is also proposed that:

- *Keys be replaced regularly*
- *Keys be made available only to the people who need to have access to them*

6.8.2 SQL Injection

SQL injection attacks are one of the most common and dangerous web application security issues on the Internet, and so is a big concern in the cloud. SQL injection vulnerabilities occur when an application takes user content data and uses it to construct SQL (Structured Query Language) statements without first properly validating or sanitizing that content [26]. SQL injection attacks take advantage of SQL injection vulnerabilities to steal sensitive data from the database, modify or destroy the stolen data, execute administrative commands on the database, or in some cases take control of the whole machine. SQL Injection attacks have gained popularity and effectiveness because this type of attack can effectively compromise data layers despite the existence of firewalls and intrusion detection systems.

According to breach.com [26], SQL injection attacks were the most common type of security incident resulting in security breaches in 2008 .Known high-profile SQL injection attacks^{19 20} have caused serious damages in the past. One way to minimize these kinds of attacks are to ensure on code that all input to the application from a user, a component, or another program are validated properly. This helps to ensure that the input is free from characters that cause SQL injection attacks. It not only helps to protect against SQL injection attacks but also other attacks, such as XSS attacks[26] and buffer overflows.

6.9 The Elastic IP (EIP) feature

This section is dedicated to a feature in Amazon called the Elastic IP [11,42]. The problem is that such features as these are by default on . This section shows that the use of EIP makes it easier for an attacker within the cloud network to initiate a *domain name service (DNS) cache poisoning* attack [55] if it is associated as the web server instance launched in the cloud. By showing that this feature is unsafe , features that associate a user's instance permanently to an IP address would have to be avoided.

Amazon [11] states that EIPs can be associated to a user's instance permanently in the cloud. A DNS cache poisoning attack is a security or data integrity compromise in the Domain Name System (DNS) server. The compromise occurs when data is introduced into a DNS name server's cache database that did not originate from authoritative DNS sources. It may be a deliberate attempt of a maliciously crafted attack from within the cloud provider on a name server as described by Tom Olzak [55]. Knowing that the cloud provider assigned the EIP to the client, and the EIP is always going to be used by the client every time a user's web server instance is launched, an attack on a DNS server with this EIP address would be fatal if it happens. The figure below shows users assessing *exact.com*, but as a web server instance which is known by its EIP: a.b.c.d

¹⁹ Known SQL injection attack 1 : <http://www.ftc.gov/os/caselist/0523148/0523148complaint.pdf>.

²⁰ SQL injection attack 2 : http://www.scmagazineus.com/Phishing-scams-may-target-Ameritrade-breach-victims/article/35699/?DCMP=EMC-SCUS_Newswire

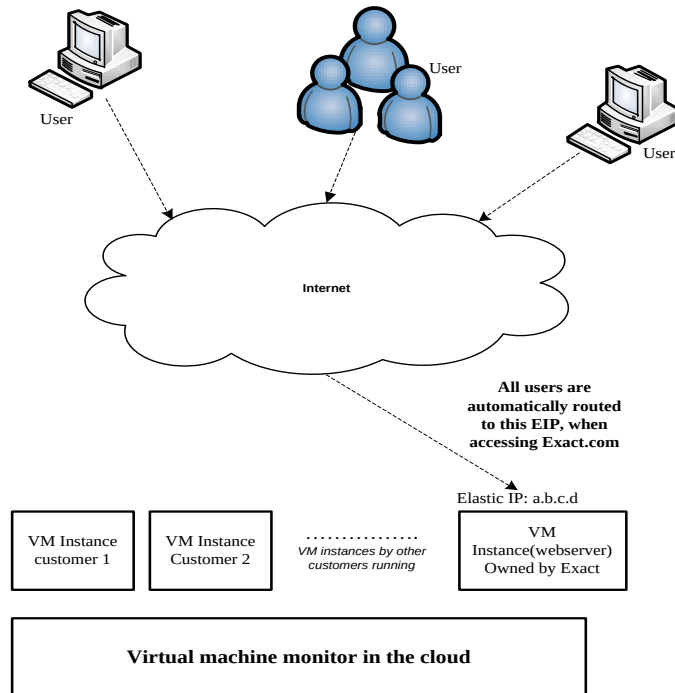


Figure 24: EIP feature in the cloud [11]

Next, a malicious hacker, knowing that this EIP is always be used by the User in the cloud manages to instigate a permanent attack on the DNS server. The DNS server then becomes poisoned, and returns an incorrect IP address, diverting traffic to another computer associated with the IP address of the hacker. The figure below shows the before and after of a DNS server being hacked.

Hence forth every time a user launches its web server instance on that EIP, it is always manipulated to point to the malicious IP, which is controlled by a machine owned by the hacker.

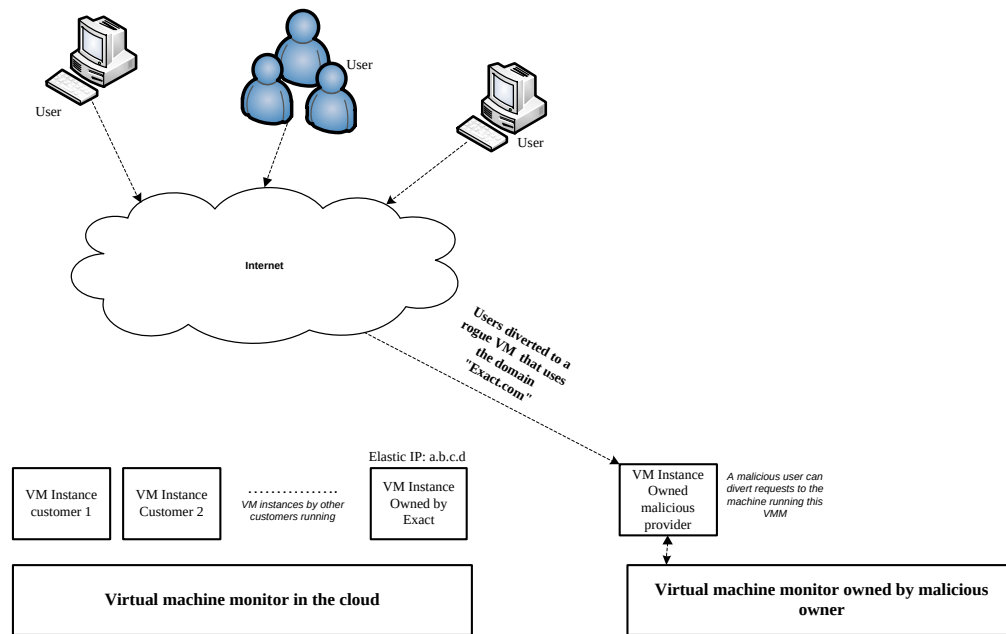


Figure 25: Hacker attack on the flaw of the EIP based on the work by Tom Olzak [55]

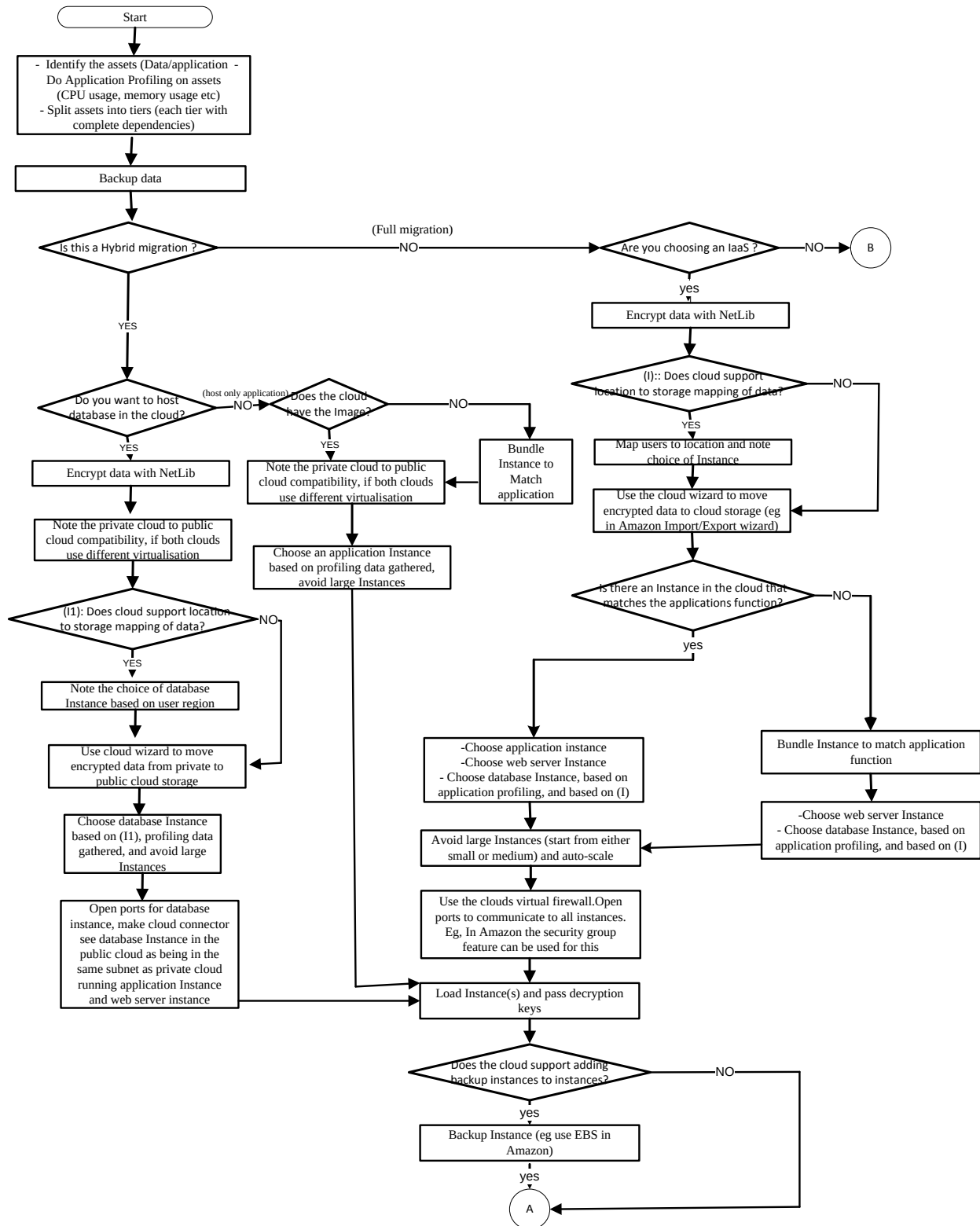
This kind of attack can equally be caused when the domain name system security extensions (DNSSEC)²¹ which is supposed to secure certain kinds of information provided by the Domain Name System (DNS) as used on Internet Protocol (IP) networks have been hijacked by a malicious user. It is supposed to act as a set of extensions to DNS which provides to DNS clients origin authentication of DNS data.

One way of minimizing this threat is by always disassociating an Elastic IP address from the instance every time a new instance is booted up, by doing this the instance is automatically assigned a new public IP address thereby limiting the probability that a malicious cloud manager targets a particular public IP address (EIP) that it knows to belong to one of its customers.

There are features in the cloud that can help automatically minimize Denial of service .In Google a feature known as the ‘DoS Protection Service Configuration’[85] can protect an application from running out of quota when subjected to DoS attacks or similar events with App Engine Denial of Service Protection Service. It is also possible to update the DoS configuration when an upload of the application is made to the App Engine. In Azure however, Azure load balancing partially mitigates denial of service attacks from the internet and internal networks [83]. But this mitigation is done in conjunction with the developer defining an appropriate service definition VM instance count *scale-out*. Windows Azure VM’s are accessible via a public IP address (just as is the case with Amazon). The Public IP is routed through Windows Azures load-balancing infrastructure. Azure can monitor and detect manually initiated denial of Service attacks and removes offending Virtual Machines from the network

²¹ The DNSSEC information page, <http://www.pir.org/dnssec>

Flowchart Migration Methodology



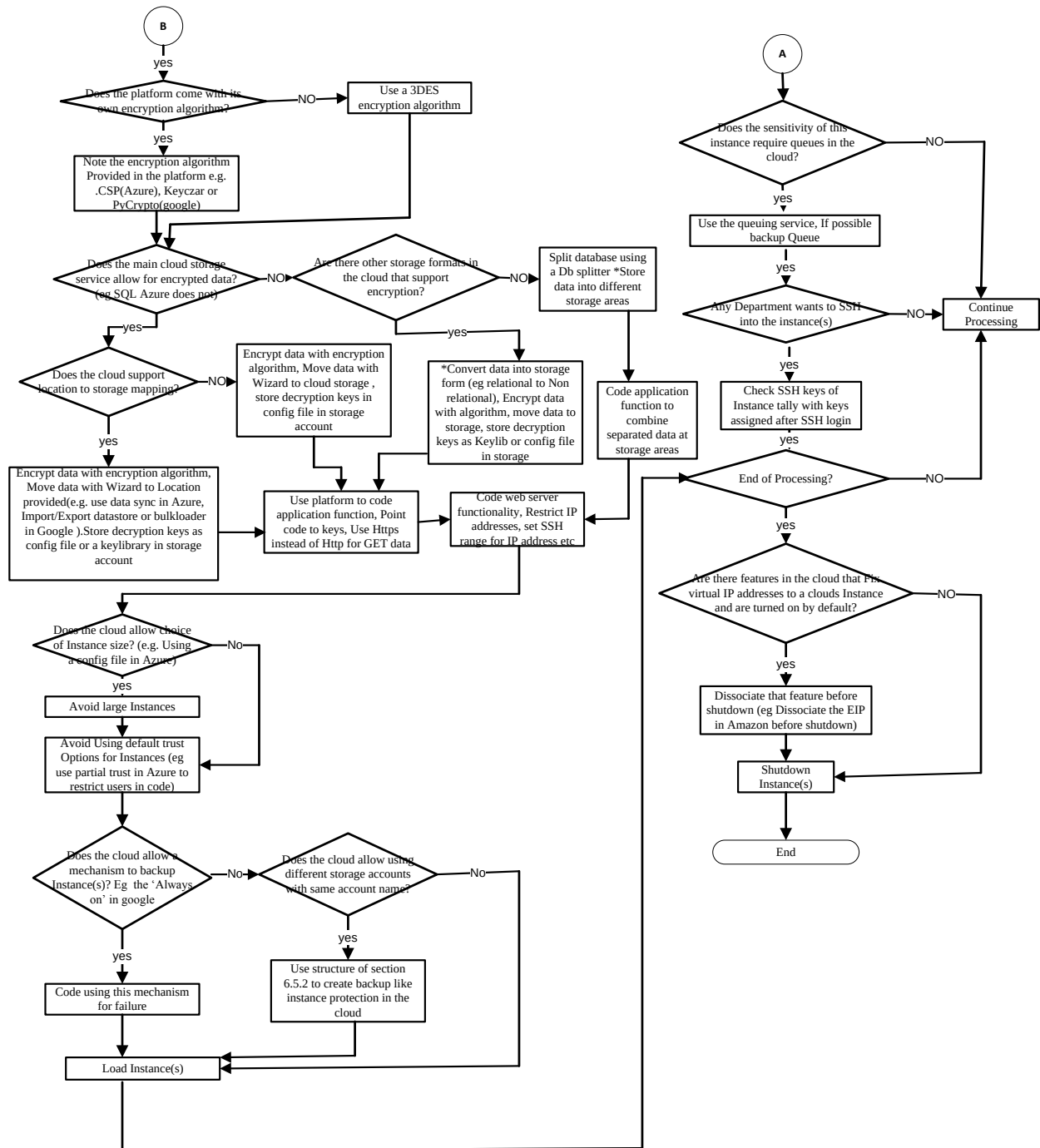


Figure 26: Flowchart Migration Methodology

In the flowchart above all the processes have been discussed in different areas of this work so far. The squares show the processes, the decision making and branching show 'if and else' with Yes and No. The circles link to other parts of the flow. For example Processes such as *'Identify the assets (Data/application), Do Application Profiling on assets (CPU usage, memory usage etc) etc'*, *'Backup data'* can be found in the beginning sections of Pre-migration, in Chapter 5. The reason for the choice to 'Encrypt data with NetLib' can be found in Section 6.8. The discussions on why to *'Avoid large Instances'* is discussed in section 6.5. The process *'Use the queuing service, If possible backup Queue'* is discussed in section 6.3. The idea behind the process *'Dissociate that feature before shutdown (e.g. dissociate the EIP in Amazon before shutdown)'* can be found in Section 6.9. Section 6.8 discusses these processes: 'Use a 3DES encryption algorithm', 'Convert data into storage form (e.g. relational to Non-relational), Encrypt data with algorithm, move data to storage, store decryption keys as Keylib or config file in storage', 'Note the encryption algorithm Provided in the platform e.g. .CSP (Azure), Keyczar or PyCrypto (Google)', 'Split database using a Db splitter and Store data into different storage areas'. Section 6.6 discusses the process *'Open ports for database instance, make cloud connector see database Instance in the public cloud as being in the same subnet as private cloud running application Instance and web server instance'*, Section 6.5.2 and section 6.3 shed more light on *'Backup Instance (e.g. use EBS in Amazon)'* etc

7 Design Pattern to Minimize the Insider Threat

This chapter Introduces the design pattern to ensure a customer virtual machine is loaded and monitored properly. It is based on the TPM chip described in *chapter 4*, and at different stages compares the design to similar existing works. It is somewhat dedicated to the *facilities layer* in the cloud model .

7.1 Malicious insiders in the Public Cloud

One of the major security areas of interest in this work is with malicious insiders [64] listed as one of the major security concerns by the cloud security alliance (CSA)[64]. The Problem in this chapter is coming up with and showing that the final design would minimize the effect of malicious insiders if implemented.

Malicious insiders exist in the *physical or facilities layer* in the cloud layer and are classified by the CSA as users in the service provider's organization with approved access or privilege to the cloud information systems that is motivated to compromise information confidentiality, integrity, and or availability. For example, a provider may not reveal how it monitors employees or grants access to physical and virtual assets. Depending on the access granted, an insider could collect confidential data or even gain control of the cloud services with little or no risk of detection. P. Barham et al [7] shows how a malicious administrator can surreptitiously attack a VM in the cloud using his higher privileged access to inspect memory, monitor VM communications, and perform suspend and reboot attacks. Xenaccess [7] for example can allow a system administrator to run a user level process in Dom 0 that can directly access the content of a VM's memory at run time.

7.2 The Facilities layer

The physical layers in the cloud are the machines that host the instances. They are where the data centres exist, and as such are where the malicious insiders reside. As mentioned in *section 6.2* under 'user Locality', Azure and Google don't give users the option of controlling the geographic location of users' data in the cloud, but Amazon allows this. The design would be based on the ability to place data in a particular zone(s) based on Amazon .This section would start with related work for the design and proceed to the design itself

Extensive discussion of Virtual machine monitors and their properties can be found in the work by Goldberg [1,2] and more work can be found in Disco [3] and VMware [4,5]. Chen [18] in his work argues for routine and extensive use of VMMs for security purposes. Ben Smyth et al [20] designed a direct anonymous attestation scheme (DAAS) that draws upon techniques developed for group signatures, and credential systems to attest to a platform. The approach can be seen as a group signature scheme without the ability to revoke anonymity, with an additional mechanism to detect rogue members. In broad terms the host contacts an issuer and requests membership to a group. If the issuer wishes to accept the request, it grants the host or TPM an attestation identity credential. Tal Garfinkel et al [19] designed an architecture called *Terra* that provides a simple and flexible programming model that is meant to allow application designers

to build secure applications in the same way they would on a dedicated closed platform. Terra realizes this union with a trusted virtual machine monitor (TVMM), that is, a high-assurance virtual machine monitor that partitions a single tamper-resistant, platform into multiple isolated virtual machines. Also another design similar to the TVMM is the concept of the trusted virtual domain that is explained in *APPENDIX 6*. Tal Garfinkel et al [19] further explains that the TVMM can ensure that existing applications and operating systems can run in both open and closed boxes. Applications running inside a closed-box VM can tailor their software stacks to their security requirements. A single VM running in a closed box can also be attested to by an attestation firm to check for how trusted the VM properties are [19].

Ronald Toegl and Michael Hutter [21] designed an attestation token architecture which is based on a commodity smart phone. They defined kiosks as machines that customers would want to authenticate. For example a point-of-sales terminal in a shop that can collect users' Pins according to them can't be trusted as it can also collect their PIN together with the information on the magnetic stripe of their credit card for later frauds. The same holds true for other types of Automatic Teller Machines (ATMs) and payment terminals. Ronald Toegl [21] explains that when a user encounters a room with several terminals kiosks available, they have no means to establish the trustworthiness of a single device. Therefore he might choose a malicious kiosk computer for his task, unmeaningfully exposing private information to an attacker. In their design [21], the user is equipped with a smart phone which serves as an attestation token, they termed this smart phone a Mobile attestation Token (MAT). Their design was dwelt on a Verification server (VS) and configuring policies according to their preferences and needs, this warrants that there be a public-key certificate and the URL of the Verification server be transferred to the MAT. With this, the kiosk can be sent a link of the VS's URL so that it can authenticate itself to the VS, and a message is sent to the MAT of the user on how authentic the kiosk is. This work serves as part of the basis of the design in this thesis. Also, more related work can be found in [23,24,25,29,31,32] [35]

Frank John Krauthheim [22] proposed a method for remote authenticating a platforms authenticity. They assumed that the security is shared between the client and operator. They used a Private virtual infrastructure factory (PVI) to serve as a controller, root authority, and policy decision point for the entire interaction between the service provider and the client. Their PVIF design in reality is supposed to be an application running on a dedicated trusted server that has built in hardware to accelerate cryptographic operations and to provide some degree of randomisation. Hence as part of the basis for the design in this thesis, the concept of using this sort of technique as a means of assuring security are inherited, but instead this machine is placed in the service provider's domain. Also in Frank Johns' design, an agent called the 'lobot' which is a VM appliance that acts as an agent to locate trustworthy platforms, is coupled with every launched application (every virtual application was termed aDom) on the virtual machine; they termed the combination of the application and lobot the virtual server domain (vsDom). The figure below shows a picture of the binding definition of their lobot and the application VM [22]

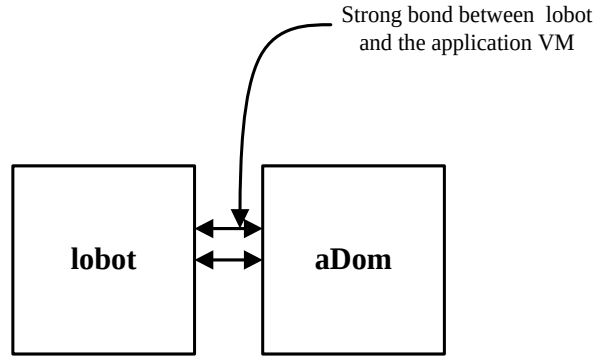


Figure 27: Bond between a Lobot agent and an aDom [22])

Their system was designed such that after lobot verifies the target environment the aDom would then get launched on that machine. The PVIF initiates this. Being the component where all the components of the PVI are provisioned, the PVIF is supposed to be responsible for virtual TPM (vTPM) key generation and certificate generation. Since vTPMs do not have valid endorsement keys, an EK must be generated for each vTPM to trust the operations of the VTPM. This factory is also used to maintain the master images for the application server's images, Lobot images and holds keys and certificates. The figure below shows what their PVIF typically would hold [22]

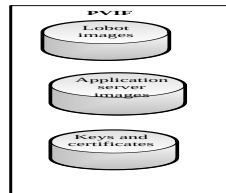


Figure 28: Inside the Private virtual infrastructure factory (PVIF) [22]

But their design is *unrealistic* when compared with how the cloud currently operates. Application images are not directly controlled by both the cloud provider and the user, they are the property of the cloud provider. Placing keys within an isolated region in the cloud can be risky if the PVIF gets compromised. Using an agent like lobot, to act as a mobile agent adds to the *unrealisticness* in a cloud environment.

7.3 Design to Minimize Malicious Insiders

To ensure that an organizations' virtual machine is launched on a trusted machine in the cloud, and to ensure that the administrator cannot direct it to a chosen machine in the cloud before the VM is launched, the work by Ronald Toegl et al [21] is used as a reference

A summary of the design by Ronald Toegl and Michael Hutter [21] is discussed below

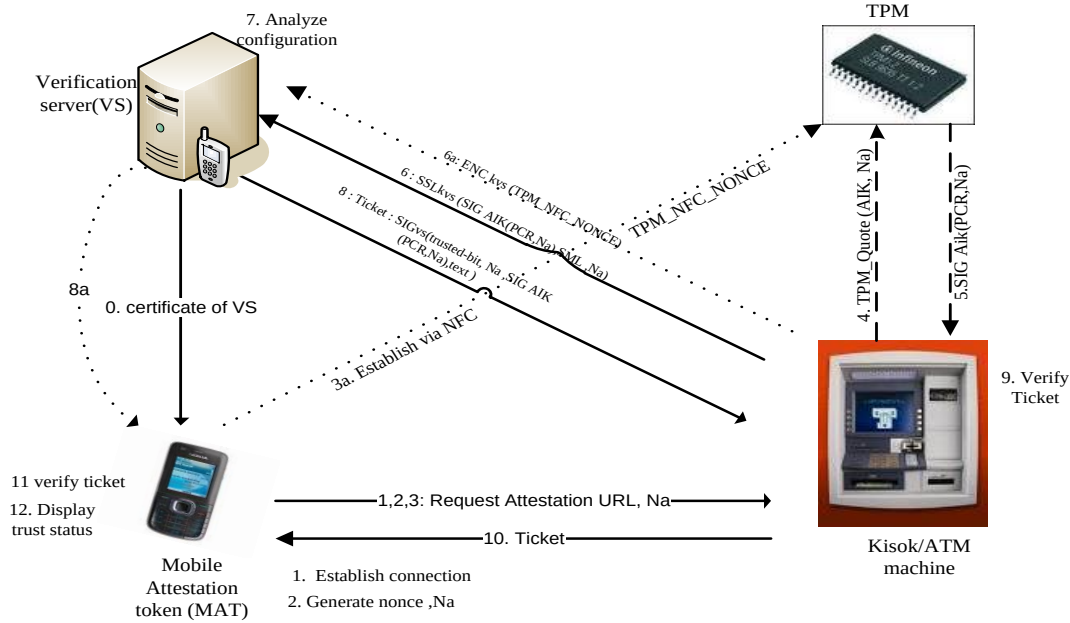


Figure 29: Ticket-based Local Attestation scheme using MAT [21]

The explanation of the steps in the *figure* above can be found in [21], but since it is the basis for the design in this work, there is a need to discuss the steps for clarity in future sections.

In step ‘0’ the user chooses the VS that is deemed trusted and configures it with policies according to her preferences and needs. The public-key certificates and the URL of the VS is then transferred to the MAT. In ‘1’, the MAT has encountered an ATM machine/Kiosk that is public and untrusted and connects to the attestation service of the Kiosk/ATM. As part of ‘2’ the MAT generates a nonce N_a to protect against replay attacks. In ‘3’ the first message to the kiosk is then sent. The user initiates the attestation with her MAT, transmitting the URL of the verification server along with N_a . *‘3a’ can be ignored as it’s an optional way of connecting with an electromagnetic near field communication (EFC) with the mobile phone for a more secure connection to the TPM directly. In ‘4’ the kiosk then gather a quote of its recorded system state from the TPM. In ‘5’ the quote, SigAIK (PCR, N_a), the signature over all PCR registers under an Attestation Identity Key AIK of the TPM, is then returned to the kiosk, containing the nonce of the MAT. In ‘6’ the kiosk then needs to establish a secure connection to the VS via the URL provided by the MAT. It transmits the quote together with the AIK certificate and the stored measurement log (SML) which documents the chain-of-trust of the kiosk in detail,. *just like ‘3a’, ‘6a’ can be ignored as they serve the same purpose. In ‘7’ , after the VS receives the complete information, it can then analyze the quote and decide the trustworthiness according to the detailed requirements of the user, using its local or other available Known-Good-Value services or property extraction modules. It also validates that AIK is indeed a TPM protected key and asserts that the PrivacyCA did not revoke the certificate. ‘8’ shows that a trust decision has been made, the VS assembles and returns a ticket to the kiosk: SigVS (trusted, N_a , SigAIK (PCR, N_a), text). It contains a binary trust decisions (the trusted-bit) and a free text for additional messages. In ‘8a’ optionally this message could have been sent to the users phone .In ‘9’ the ticket is validated on the VS before being passed on to the MAT. This allows a kiosk operator to gather statistics on how much acceptance the offered configuration finds. In ‘10’ the unmodified

ticket is passed on to the MAT. Step 11 shows the MAT verifying the signature using the pre-installed VS certificate and also that Na is from the same session. In 12, If this succeeds, it can then proceed to finally display the result to the user.

****Let there be a machine, called Machine A, in the cloud running the attestation process on behalf of the verification server or on behalf of a third trusted party as related to the work by Ronald Toegl [21]. This machine is also similar to the function as the PVIF [22], being the root of trust for both the platform and the machines that the VM is migrated or provisioned on.**

****Let it be that Machine A resides in an isolated region within the cloud providers' domain. A diagram of the machine is shown below:**

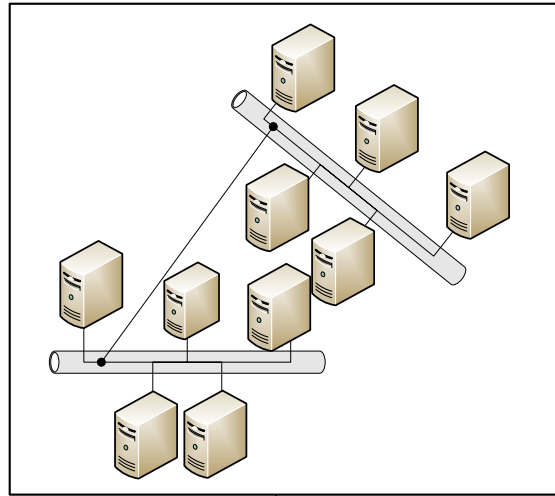


Figure 30: Machine A in an isolated region based on [22,21]
(The dotted lines around machine A, show that it is in an isolated region)

Within the cloud and in a datacenter, machines are multi-linked in an unknown structure (*as was also assumed in chapter 5*). The assumption of placing Machine A in an isolated area within the datacenter is directly related to the assumption made by Frank John Krautheim [22] where their PVIF was placed in an isolated area

****Let the collection of untrusted machines inside the cloud be the same as the untrusted Kiosk or ATM machine (in this case a collection of untrusted ATM machines/kiosks), based on [21]**

Zone 1 (untrusted Machine collection in this zone = untrusted ATM/Kiosk collection)



Low latency connectivity between zones

Zone 2 (untrusted Machine collection in this zone = untrusted ATM/Kiosk collection)

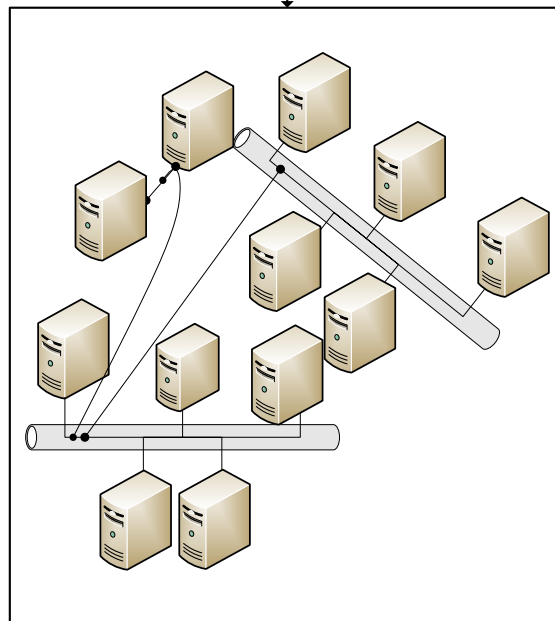
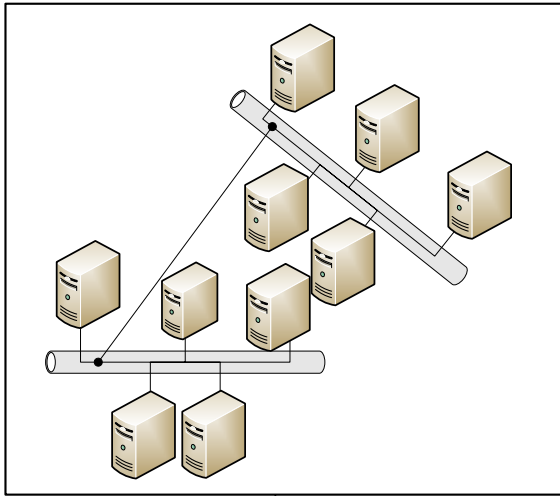


Figure 31: Assumed Machine/ATM arrangement within a zone(s) in the EU regions' data centre

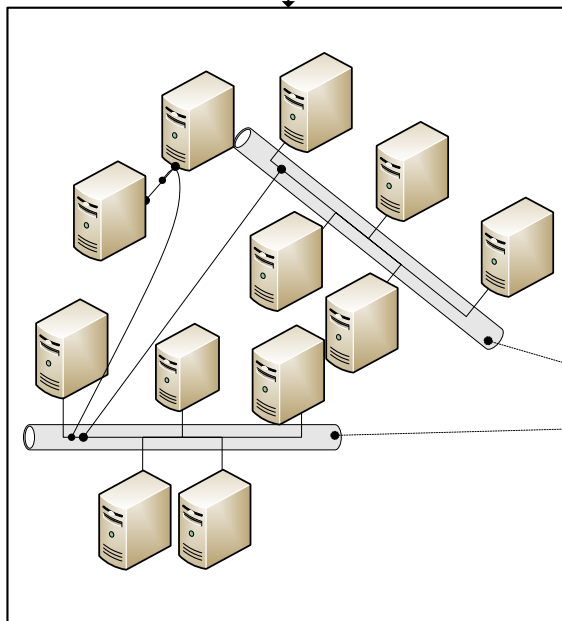
The placement of machine A needs to be such that it establishes a connection within a zone in the datacenter, but needs to be in an isolated region within that zone. The figure below shows this isolation:

Zone 1 (untrusted Machine collection in this zone = untrusted ATM/Kiosk collection)

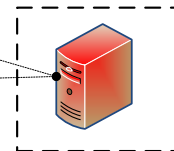


Low latency connectivity between zones

Zone 2 (untrusted Machine collection in this zone = untrusted ATM/Kiosk collection)



Machine A= Mini-verification server(VS)/PVIF/ machine acting on behalf of TTP



Machine A, in an isolated area within zone 2

Figure 32: Location of root of trust machine in Cloud (based on the work of Frank John Krautheim [22])

Here, Machine A is in isolated region within Zone 2. It can communicate with the machines of Zone 2, and through the low latency links can communicate with zone 1 and any other zones within the region[11,42].

Next, the design is based on the fact that, before launching a VM, Exact does not know which physical machine the VM is assigned, and, among the components of the service, only trusts machine A (just like a user with an MAT[21] only trusts the verification server(VS)) which can communicate with any zone within a region

**Let Exact or an organization, which are the users be the same as the mobile attestation token [21] as they do not trust any machine their VM is loaded on

If Exact has a signed up and has an account with the cloud, which is the assumption henceforth; they have access to a list of available VM images [11]. After the VM image is chosen it mustn't be launched as they don't know which machine the Instance is launched on. The diagram below shows more:

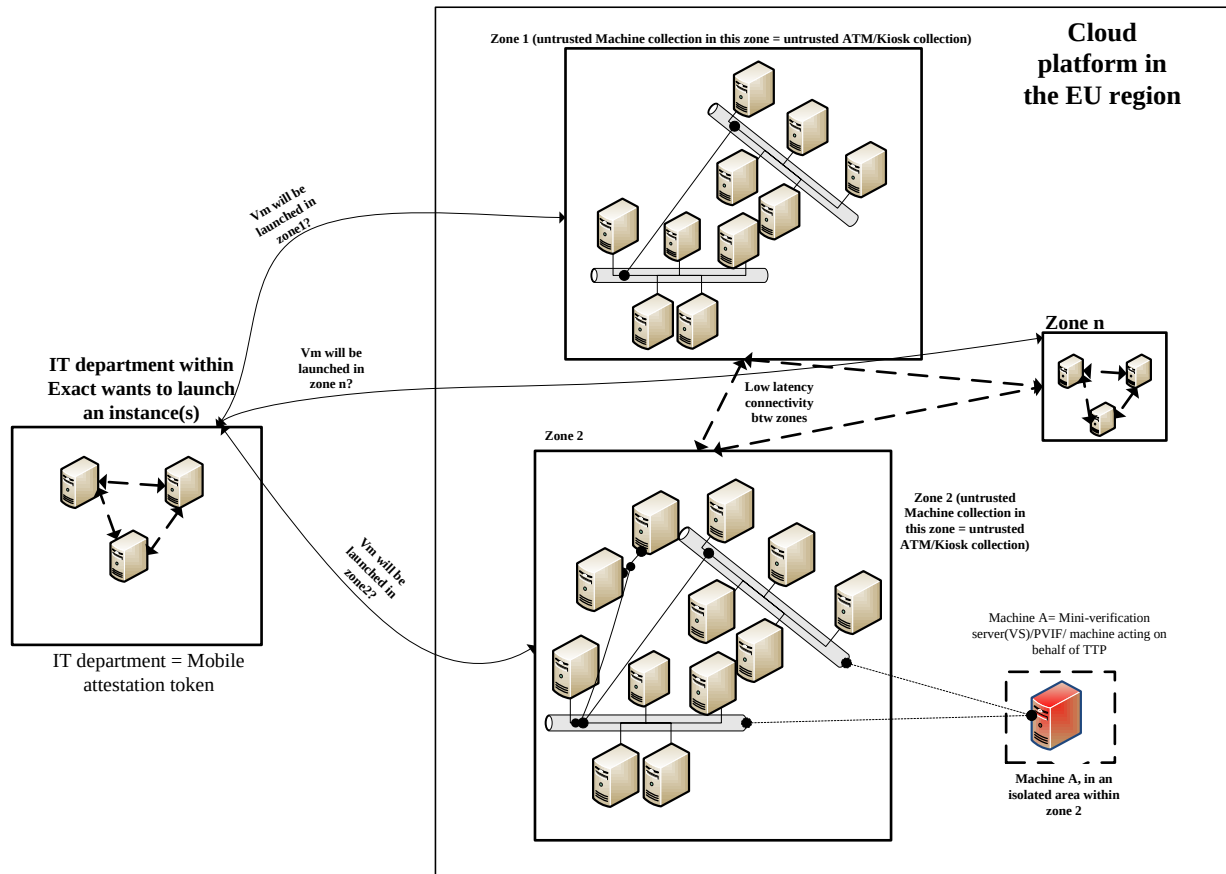


Figure 33: Unknown machine where the VM is launched

The communication commenced in [21] (see *step '2' in figure28*) where the MAT had to generate a Nonce to send to the kiosk (which was the untrusted machine), But in the cloud is that users can select instances (AMI) to be launched and a device like the MAT exists which can send a nonce to the cloud (**Let this nonce be called N_1). Hence let it be that Exact has selected an AMI to be launched as shown in the *figure* below. If the Instance is sent to the cloud once chosen without encrypting its state, it is the same as using the ATM machine[21] without knowing if it is trusted or not. Therefore the state of the Instance needs to be encrypted.

The concept of session keys [22] and public keys [22] is introduced next. The Simplest kind of communication between two entities is shown in *APPENDIX 2* between Alice and Bob. A Session key was used to initiate message conversations. Also encrypting a message with a public key ensures that only an entity that has the private key of the public key can view the

message.(see APPENDIX 2) .This same concept was used by Frank John Krautheim [22]. A session key can contain other keys (such as the public keys of other entities), used to initiate a conversation

Therefore Exact has to generate a session key, **Let this be called S_{VM} . The session key, S_{VM} , needs to contain the public key of Machine A, so that only Machine A can decrypt the message (see the communication between Alex and Bob in APPENDIX 2) when it is sent to the cloud. As seen in APPENDIX 2, encrypting the session key, S_{VM} , with Machine A's public key ensures that only Machine A can access it. (and Machine A servers the same function as the verification server in [21] and is only authorized a machine if it decides that machines quote is trustworthy, see step '7' of figure 28 above).

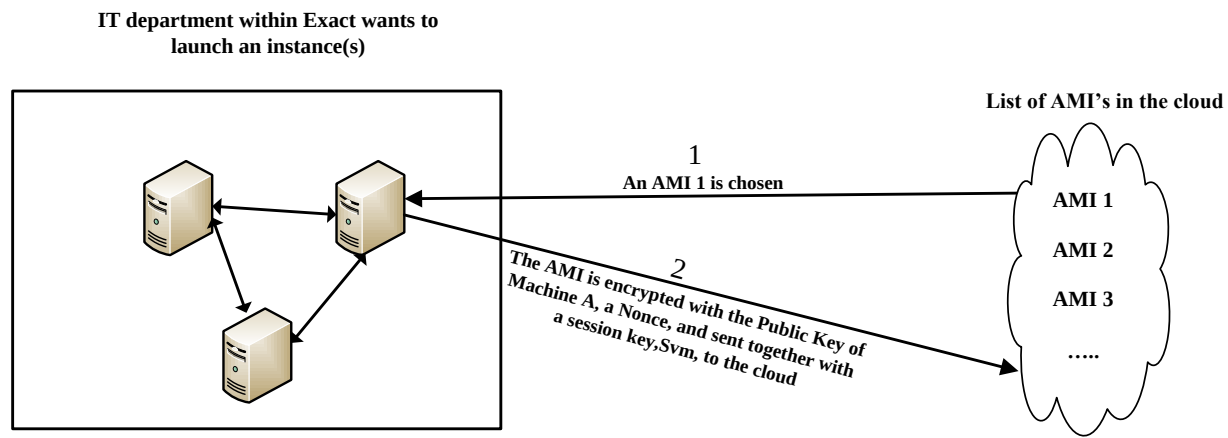


Figure 34: Accessing an Instance in the cloud [11,42]

Upon receiving the request to launch a VM, the cloud administrator designates a machine [111] from one of the zones to host the VM. **Let it be assumed that this request is forwarded to machine 1 which is in Zone 2. This is illustrated in the figure below

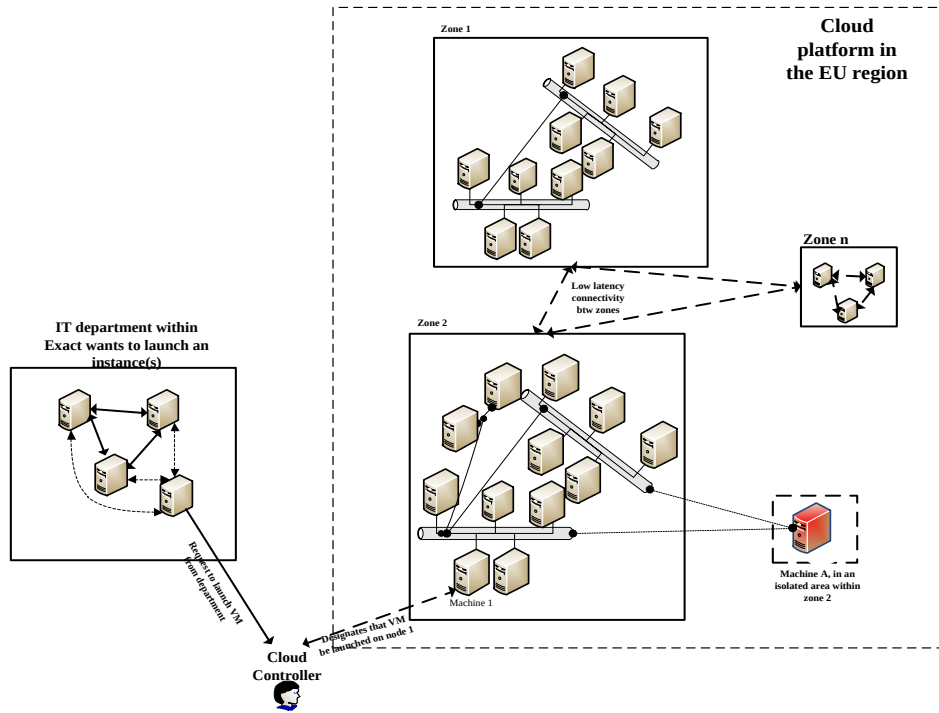


Figure 35 : The Cloud controller in the cloud [111]

Since Machine 1 needs to access the encrypted state of the VM in order to boot the VM, it has to send some sort of message to Machine A which is the only machine that can decrypt the public key before the instance can be launched. Therefore Machine A and Machine 1 need to establish a trust relationship before Machine A can decrypt the instance state to ensure that Machine 1 is not malicious. In [21] the URL (see step ‘3’ in figure 28) of the verification server was sent along with the Nonce from the MAT to the kiosk, and was used for contacting the verification server. The assumption in this work is that the URL of Machine A is either known by Machine 1 or was sent along with the nonce like in [21]. *Another approach* might be to assume that an IP address instead of a URL of machine A is known by all machines (including Machine 1), since IP addresses more uniquely identify machines in a network. Hence Machine 1 can use either the URL [21] or an IP address to communicate with machine A. From here the sequence of events that take place are the same as what happens in [21] step ‘4’, step ‘5’ and step ‘6’ as seen in figure 28 .i.e Machine 1(which is the kiosk) gathers a quote of its recorded system state from the TPM. The quote, SigAIK (PCR, N_1), the signature over all PCR registers under an attestation Identity Key AIK of the TPM, is then returned to Machine 1 , containing the nonce generated earlier. (The quote function is explained in chapter 4, section 4.2 as TPM_Quote) .Then Machine 1 needs to establish a secure connection to Machine A (acting as the VS) via the URL or IP address already discussed. Similar to [21] Machine 1 then transmits the quote together with the AIK certificate and the stored measurement log (SML) which documents the chain-of-trust of Machine 1 in detail, based on [21] .Also transmitted from machine 1 is the session key that contains : $S_{VM} \{ \{Instance\} Public \text{ key of Machine A} \}$.The brackets are similar to those seen in the work by Frank John Krauthheim[22], it denotes that the session key, S_{VM} , holds an instance that is encrypted with the Public key of Machine A .

In *step '7'* as seen in *figure 28* , the same sequence occurs next after Machine A (being the VS) receives the complete information, it can then analyze the quote and decide the trustworthiness according to information it received, using its local or other available Known-Good-Value services or property extraction modules[21]. It also validates that the AIK it received from Machine 1 is indeed a TPM protected key (i.e by evaluating the endorsement key, EK, of the TPM discussed in chapter 4). In *step '8'* as seen in *figure 28*, the same sequence occurs as a trust decision needs to be made, Machine A (the VS) decrypts (assembles) and returns not a ticket to the Machine 1(the kiosk), but decrypts the Instance state with its private key and returns the decrypted instance to Machine 1 for boot up of the instance, represented as [21]:

Machine 1(SigAIK (PCR, N_1), Instance). This is based on the output of [21] which shows: SigVS (trusted, N_a , SigAIK (PCR, N_a). As an *add-on* to the design, Machine A can store the URL or IP address of Machine 1 in a file within itself signifying that Machine 1 has a trusted state and its IP address or URL can be used for future communication. The sequence of events is shown in the diagram below:

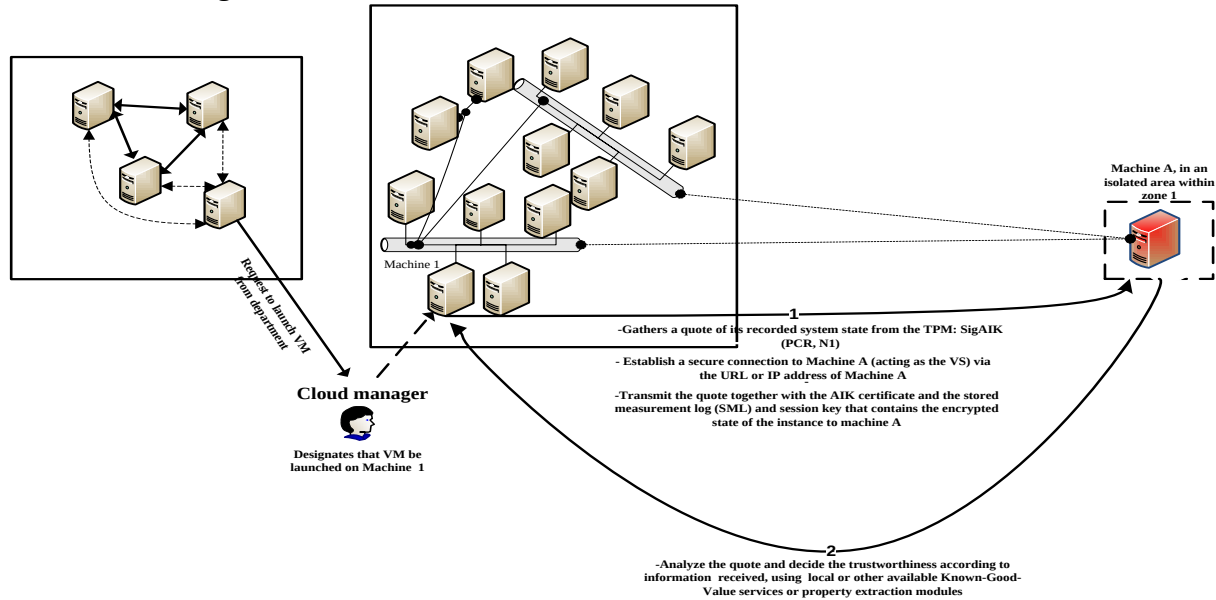


Figure 36: A trust relationship between Machines in the cloud based on Ronald Toegl [21]
(1 in the fig above= Equivalent to the combination step 4, step 5 and step 6 in figure 28 , 2 in the fig above = Equivalent to the step 7 in figure 28)

If the corresponding quote state of machine 1 was not found to be authentic or trustworthy by machine A's corresponding analysis (In *step '7'* as seen in *figure 28*) for example, then the request to decrypt the Instance state should then have been denied automatically. *One approach* after this denial takes place would be for Machine 1 to request that the cloud administrator fetch a new machine as it cannot decrypt the Instance state. The figure below shows this option:

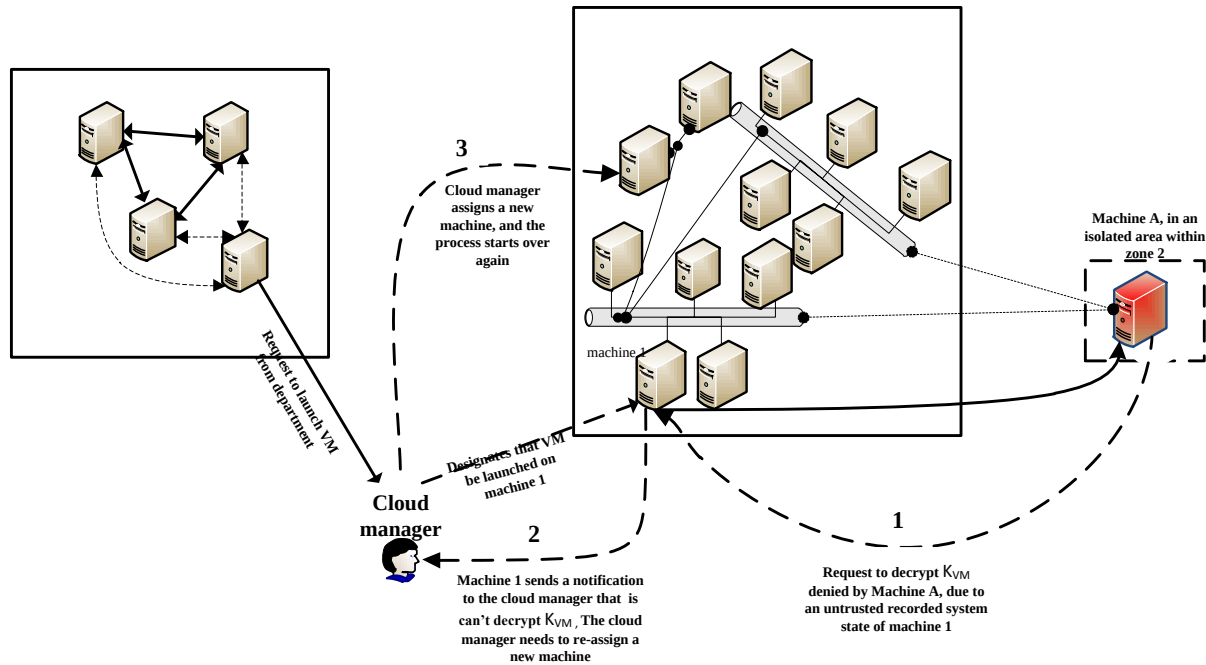


Figure 37: Sequence of events for an unacknowledged machine based on the work by Ronald Toegl [21]

‘1’: Shows that the request to decrypt S_{VM} has been denied by machine A, (this is the same as the verification server in [21] not trusting the recorded system state of the kiosk) a message is sent back to machine 1. *Another approach* might be that this denial (of machine 1) be sent back directly to the cloud manager from Machine A. The ‘2’: Shows that machine 1 receives this notification about its inability to launch the Instance and sends the S_{VM} state the way it is to the cloud manager. (‘1’ and ‘2’ might have been combined into a single step so that Machine A only send a message to the cloud manager). The cloud manager as seen in ‘3’, would then assign a new machine launch the VM, but since the instance is still decrypted, the whole trust communication process begins with authenticating with Machine A. Else if, Machine A trusts Machine 1 then: Machine 1($SigAIK(PCR, N_1)$, Instance) takes place and the state of the instance is sent back Machine 1, as discussed above. Machine 1 is now able to boot the Instance knowing that the machine is secure, This is similar to a ticket (see step ‘9’ in figure 28) being sent to the kiosk in [21] after a trust-decision has been made verifying the state of the kiosk, which in this case is Machine 1. The diagram shows this event of a trust.

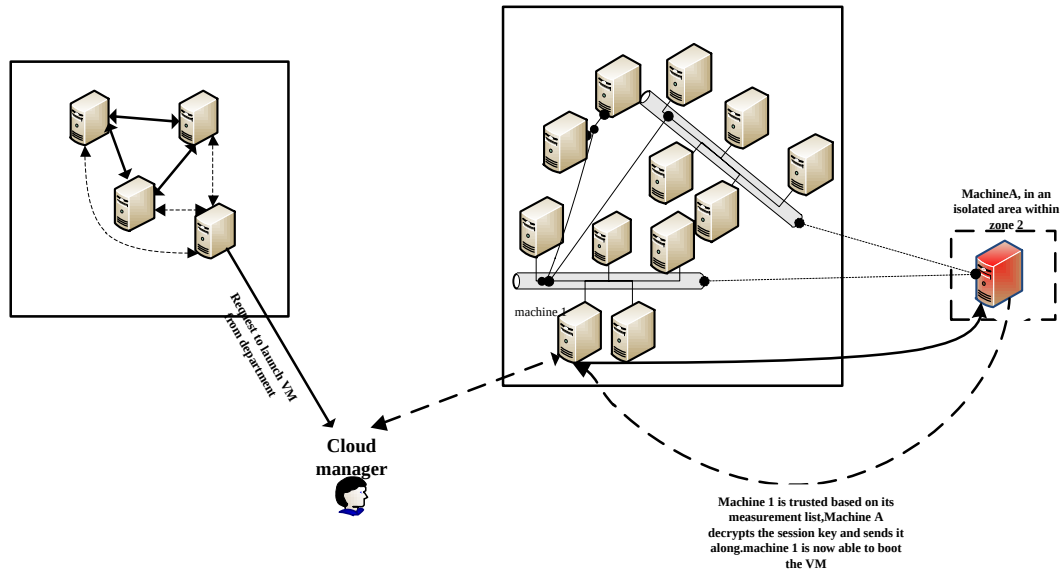


Figure 38: A Trusted Machine

*The dotted line from machine A to the machine 1 shows that Machine A trusts Machine 1, using its Known-Good-Value services or property extraction modules (similar to *step 7* in [21]). machine A decrypts the instance state sent by machine 1 and sends it back machine 1, machine 1 can now boot the Instance successfully, and is considered trusted.

In summary the design has shown that as part of the design there needs to be a secure location in the datacenter where a machine acting on behalf of a trusted party can be placed such that it is securely isolated from the main cloud infrastructure but can still communicate with other machines, this assumption exists in the work by Frank [22], as the PVIF was assumed to have been placed in an isolated area. Machine A's placement in the network needs to be able such that it can protect itself from attacks from the internet including viruses, and any other forms of attacks. All machines in the cloud need to have the TPM embedded within them for an attestation to take place. The user trying to load an instance needs to know the Public key of Machine A to encrypt the instance state. The user needs to know either the URL or IP address of Machine A in the cloud, as discussed in [21]. A summary of the function of Machine A is found in *Figure 45*

7.4 On Instance boot up

Once an Instance boots up, they generate a random SSH server keys [107]. With this key generated after a VM is launched, it can be used authenticate the VM later on if an Exact Internal user decides to SSH into the VM. This can be used as a good security check. This check is added in the Flowchart in *Figure 45*

From here the concept of monitoring the VM machine after it boots is introduced. The assumption from here on is that the VM is booted up and is running. It is necessary to show how currently TPMs can be of assistance in monitoring VM (and not just the hardware or machine itself)

7.5 Virtual Trusted Platform Modules (vTPMs) and their Exposure to attacks

Virtual machines can be monitored after they have been launched using virtual TPMs (vTPM) [55,57]. This section firstly describe work related to the internal structure of vTPMs, it then shows the current attacks machines that run virtual trusted modules are exposed to.

Stefan, Ramón and Kenneth [55] designed a virtual TPM facility in software that provides TPM functionality to virtual machines. The virtual Trusted Platform module support on Xen has also been described by Berger et al [55]; in their architecture, there is a root service which can spawn and manage new child instances. This work uses the default design based on the work by Stefan Berger et al [16], Nihal A.D' Cunha et al [17] , and supported by Xen Users Manual [105] which assumes that all Virtual TPM instances are implemented by one dedicated service (the vTPM manager) hosted in Xen domain 0. This service is meant to direct the TPM requests to the right instance in a multi-threaded manner. The states and secret data of all virtual TPM are all controlled by this central service, which is responsible for protecting the secret data. This service is equally responsible for forwarding the reply to the right caller. The vTPM architecture is shown below based on [17,16]

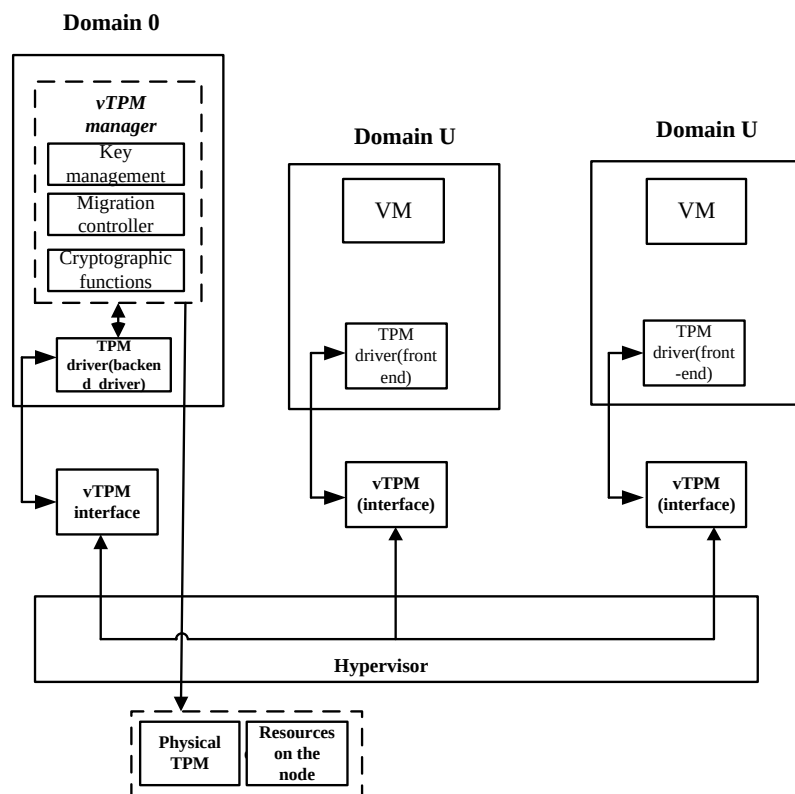


Figure 39: The vTPM architecture with vTPM manager [16,17]

The Internal communication of the architecture above as described by Stefan Berger et al[16] explains that the interface gives each domain the impression that it is accessing its own exclusive TPM, as if it were a hardware TPM; This is done with the aid of virtual devices which are provided with two collaborating drivers as shown in the figure above: the *frontend device driver*, runs in Domain-U, and the *backend device driver*, which runs in Dom 0 with access to the real

device hardware .The frontend exports a character device used for communicating with the instances. The backend driver provides a single interface where there are threads waiting for requests from the different domains that have a corresponding frontend [17]. The manager can request the state of all VM instances from its driver. The states are returned by the back-end driver, which identifies each state by its vEK (the vEKs are generated by the key management). Stefan Berger et al [16] further explains that the managers overall task is for the creation of instances, deletion, suspension and migration of instances. The migration controller shown in the figure above handles this migration requests in the manager.

All vPCRs from each instance are sent to the vTPM manager on it's request. **The problem* with this is that the manager (which is under the control of the administrator) , now has to match each instance to its vPCR (based on their vEK).In order to extend the chain of chain of trust, based on the work by Stefan Berger et al[16], the cryptographic function now has to link each instance to its underlying TCB by mapping the lower PCRs of the real TPM to the lower PCRs of a vTPM (as shown in the arrow that links the physical TPM with the manager) ;This is supposed to enable the vTPM to include the configuration of the underlying hypervisor platform during attestation

A malicious administrator who has control of Dom 0 can instantiate an attack(s) such as Software attacks, Fault generation[106] such that it causes a wrong match of the vTPM instance and vPCR to be attached to the lower PCRs of the TPM. An attestation from a Third party with these readings wrong matches shows an incorrect VM configuration.

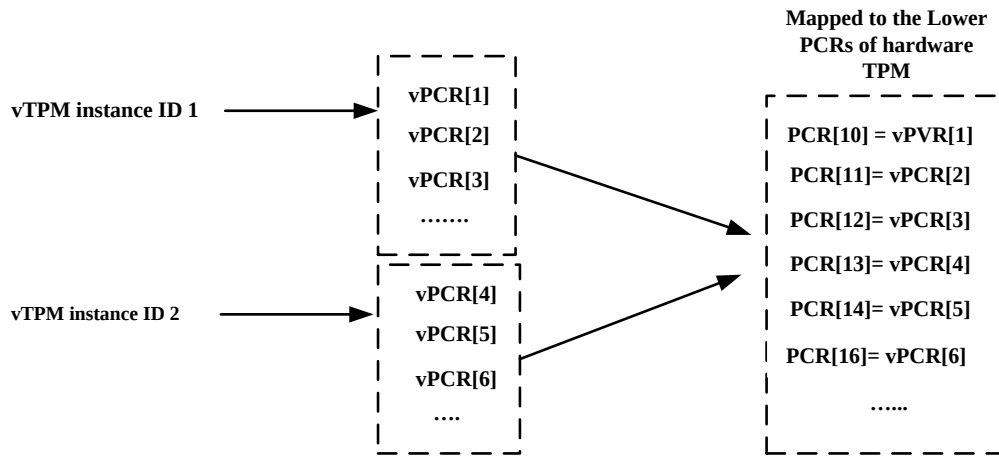


Figure 40: mapping of vTPMs instances to a hardware TPM, Stefan Berger et al [16]

The diagram above shows the mapping that occurs with vPCRs and the TPM [16]. Registry values are saved in the lower PCRs of the TPM, such that during an attestation the values in particular registers are known by the attesters. The assumption that they [16] made is for every VM there is a fixed number of positions it can have its values stored into the TPM.

A malicious attacker might want to pass codes into the manager to mask a VM attack, and map a wrong vPCR instance to the wrong PCR (based on the attack types by Sergei P. Skorobogatov [106]), such that the TPM chip gets the wrong values as shown below:

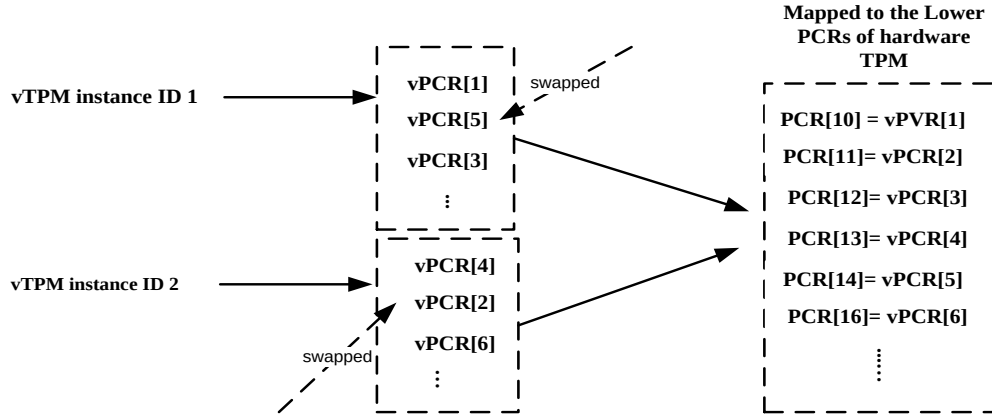


Figure 41: Swapped registry values [106,16]

The diagram above shows a malicious administrator swapping registry values between vPCR[5] and vPCR [2].The TPM only reserves PCR [0]...PCR [2] for one VM instance. So automatically PCR [1] having a corrupted registry state would deceive the attester into believing that the binary value for that VM is vPCR [2], this value could stand for the binary state of the application after loading, or binary state of a VM's memory etc.

To minimize the kind of attack in *figure 40* above, it is proposed that functionalities such as the cryptographic controller, Key management from the vTPM central manager running in Dom 0 exist in every loaded Virtual machine running in Dom U such that the only responsibility of the controller would be to migrate instances when the need arises . The diagram below shows this:

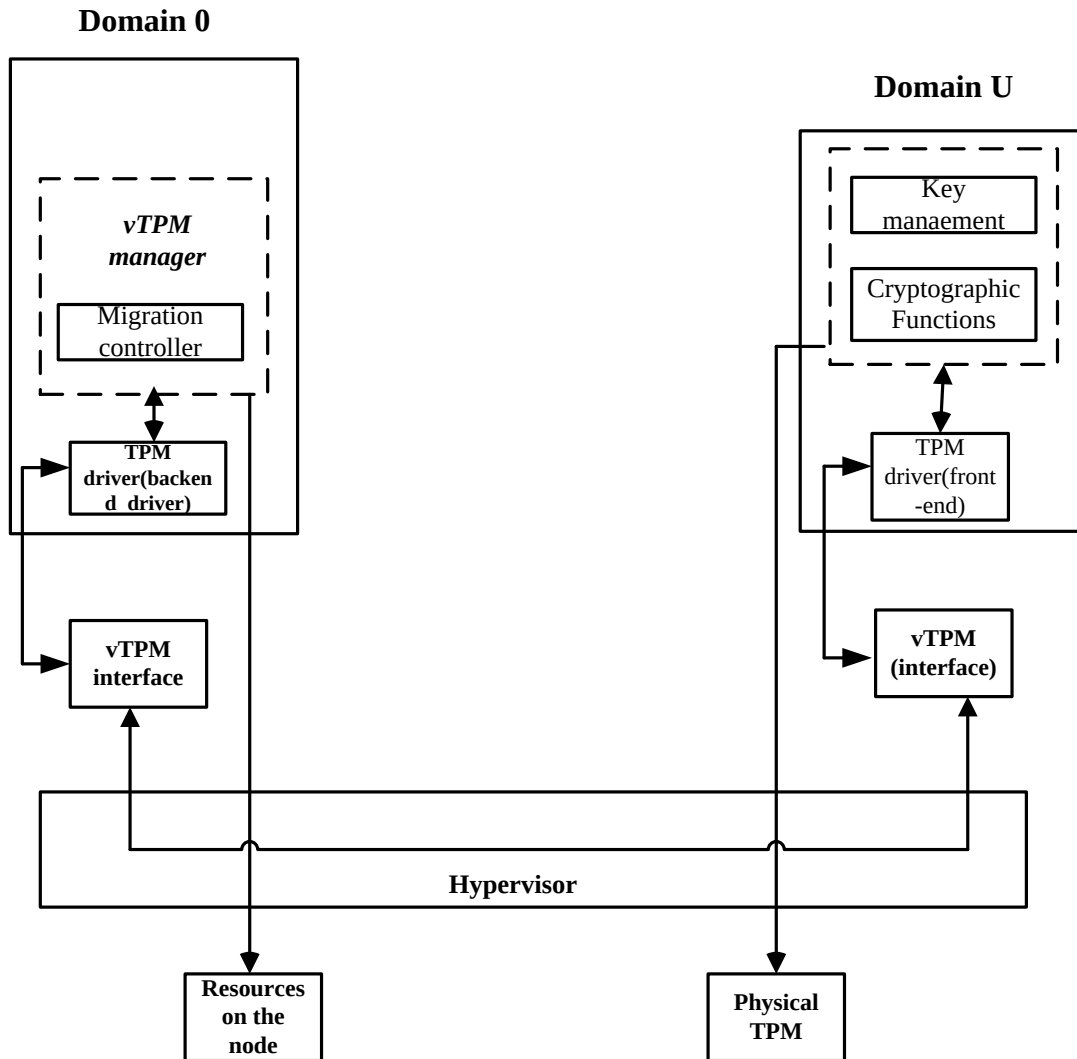


Figure 42: Inserting Dom 0 functionalities into Dom U

-In Dom U: The key management, cryptographic, were inserted as part of the operation of Domain U.

-In Dom 0: The manager only now has the migration controller function and is connected to the resource allocator of the machine. But the link between the Dom 0 and Dom U's (i.e. via the vTPM interfaces) is used to send migration requests to any VM based on the vTPM ID

With this proposed design, every VM loaded in Dom U can generate its own key and can merge its state directly to the physical TPM. Also since the key management function now resides inside the Dom U, it must now be able to generate a vEK for its vTPM by asking the physical TPM chip. So the VM must be able to request that a key be created for each vTPM, the key management now delegates the request directly to the hardware TPM. By abstracting the state of the keys and the cryptographic functionality away from a central service originally controlled by the Dom 0 (such as the central vTPM manager), certain details of key pair

generated and the kind of attack discussed in *section 6.14* can be safeguarded against and avoided.

7.6 Monitoring the Machine acting as root-of-trust

So far, the design has not incorporated any mechanism that can help monitor the state of machine A. Machine A is assumed to be isolated in the cloud, and this isolation might provide security to the machine, but the auditors might need to guarantee its state is secure. For this to be, it is proposed that a secure channel from the third party constantly attest to the state of machine A. This might be easy if Machine A has a TPM running in it. The Auditor can confirm the state of Machine A through the TPM. This can be accomplished based on the work by Steffen Schulz (2009) [41]. The communication between Machine A and the auditor can be done using a secure IPSec link as described in [41]. Machine A in contrast with Franks [22] design does not hold any application images. Users in the cloud would have no control on this machine. Also, Machine A should only be a dummy machine (with no monitor or screen, it should only be configurable remotely (i.e) by the auditing team only.

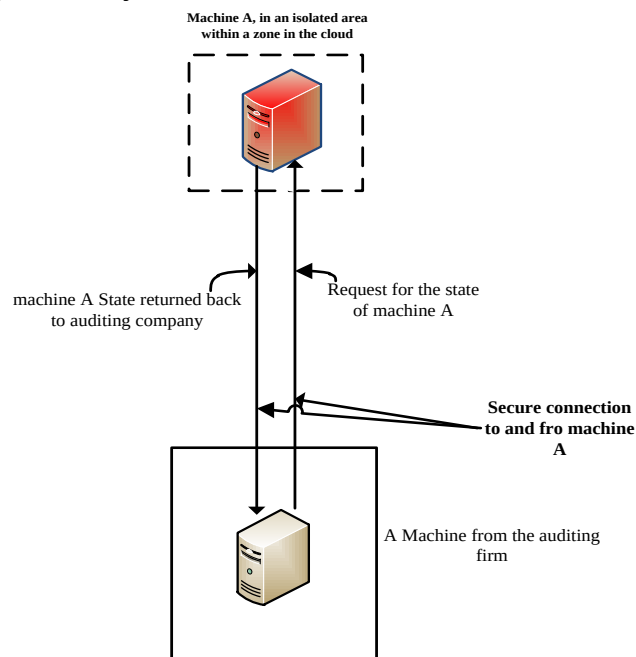


Figure 43: Monitoring Machine A based on Steffen Schulz (2009) [41]

The figure above is part of the entire design that ensures that machine A has not been modified or tampered with. By constantly checking the state of machine A through a secure link (IPSec for example [41]) the state of Machine A is always monitored

So far, the design has shown what needs to be done for an instance to check for malicious Machines, How Machine A can be monitored and, but nothing has been said so far of what needs to happen if the customer's VM is to be migrated to another machine

7.7 Migrating Virtual Machines

A live migration is the process of moving an instance or virtual machine state from one physical machine or platform to another [78,51]. *Chapter 4, section 4.4* explains the process in more detail. The security challenge in this section lies in the process of trusting this new machine if a users instance were placed on it. *One approach* is to use the work by Ronald Toegl [21], and the design in *section 6.12* Such that whenever a migration is to take place machine A can serve as the root-of-trust to that machine. For example ** Let it be that a VM state is to be migrated from machine 1 to machine 2. Machine 1 is running the instance and has been attested to by Machine A. If the VM state is to be transferred from Machine 1 to machine 2, Machine 1 can communicate with Machine A to assess the state of machine 2 before the VM transfer state begins. This is similar to the work done by Frank [22]. The figure below shows this:

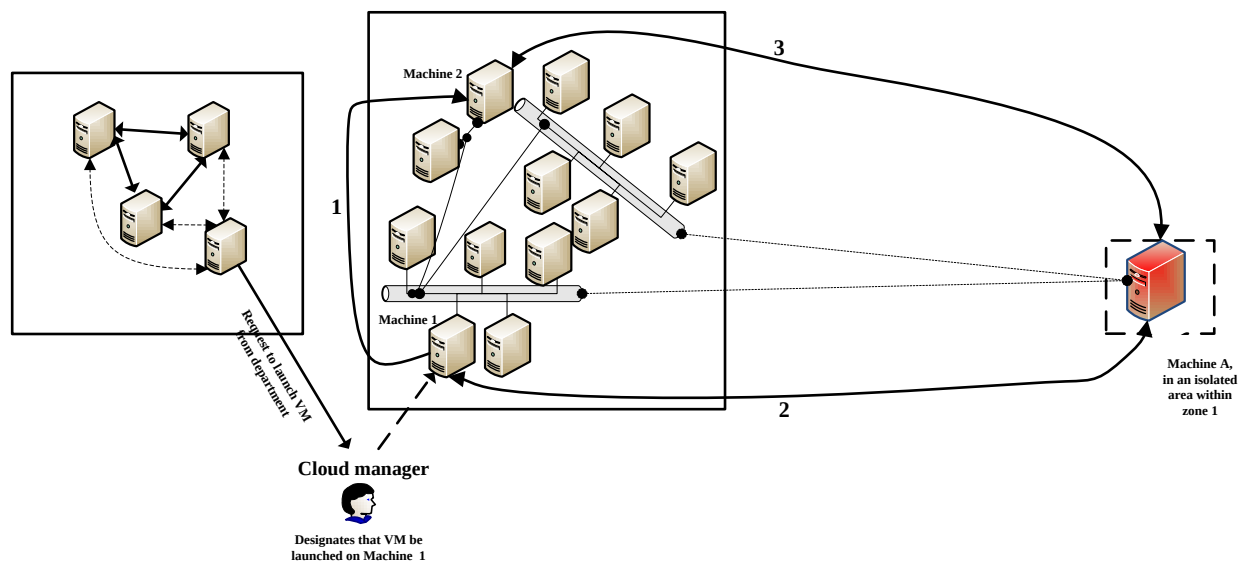


Figure 44: A migration from Machine 1 to Machine 2 based on frank [22]

In the figure above, the '1' shows that there needs to be a migration from machine 1 to machine 2. In '2' Machine 1 can send a query to machine A to certify that Machine 2 is trusted. '3' shows the communication between Machine A and Machine 2 based on *step 4, step 5, step 6* of *figure 28* (also see [21]), but this time the URL or IP address of Machine 2 could be sent to Machine A through Machine 1. The bidirectional arrow in the diagram above shows that communication is two way.

Sequence of Events for Machine Acting as Root-of-Trust

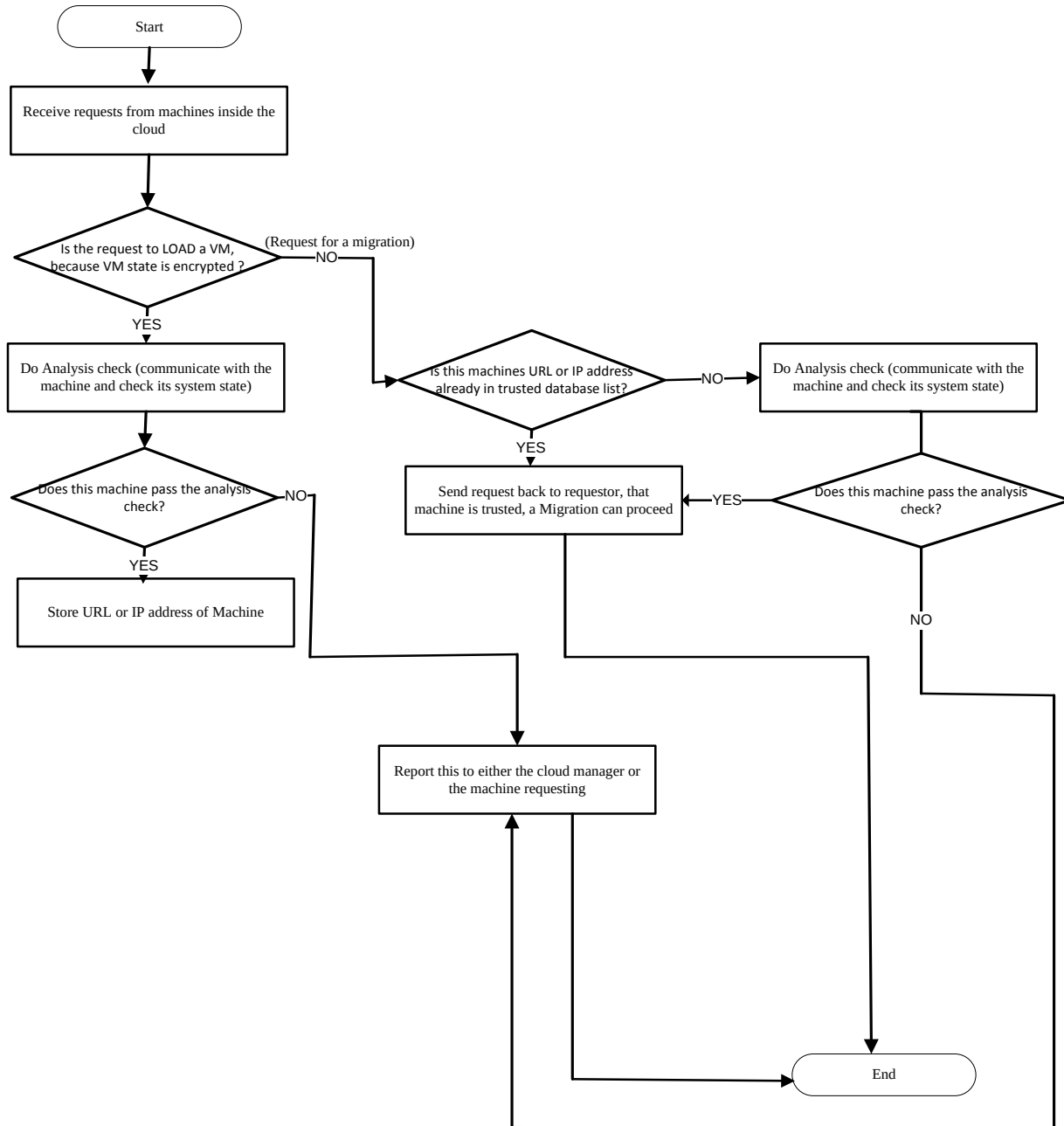


Figure 45: Machine acting as root of trust

The flowchart above shows a summary of the basic function of the machine acting as the root of trust (Machine A). This design is elaborated more in various sections of this chapter, and it explains this flow in greater detail, including the function of the organisation that needs its instance secured, the operation of other machines in the cloud and, the operation of the cloud manager

Part V

Conclusion and Reflection

8 Conclusion and Reflection

8.1 Contribution

The main contribution of this thesis shows an overview of what an organisation needs do to migrate its applications to the cloud securely. Furthermore two different designs are provided: A migration methodology design and a design pattern for organisations to ensure the security state of their instance(s) is not compromised by the cloud provider while running in the cloud. Exact received a report of how to migrate securely to the public cloud either from a private cloud or from within Exact. They are now more knowledgeable in the area of cloud computing. This deliverable helps support Exacts' decisions in the event that they choose to migrate to the public cloud

The main research goal of this thesis was:

"Create an approach to securely migrate an application(s) to the cloud, based on the features offered by the cloud provider"

This thesis project addressed the research goal by highlighting the areas where the security threats exist in the cloud. It showed a mapping between cloud features in different clouds. It explained the concept of virtual machines and their attacks in the cloud. Subsequently a secure approach was created based on a transformation of each cloud layer in the cloud stack as can be seen in the sections of Chapter 6 and the Flowchart there

The migration strategy in *chapter 6* showed that security in the cloud is not only dependent on the cloud user, but also requires that the cloud provider provides transparency in its operation. Another design in *chapter 7* showed what a cloud user needs to do if it wants to monitor its instances based on the assumption that trusted pair modules (TPM) are running in the cloud providers' platform. The literature review consisted of an extensive examination that has shown research papers on cloud features in various clouds, Pre-migration standards, Security loopholes and threats, trusted computing and attestation.

8.2 Research Questions

The research questions in *Chapter 1* have been answered during the course of this thesis.

1 *What are the security threats of migrating to the public cloud?*

Chapter 2 has shown based on the International Standards Organization (ISO) that threats are categorized into Authentication, Integrity, Non repudiation, Confidentiality, Availability and authorization. In the cloud, the threats were mapped to this categorization, with the threat of Malicious Insiders specified as one of the most *risky* by the cloud security alliance, other security threats identified were integrity in stored data, Information disclosure, Instance corruption, Service denial attacks

2. *What cloud feature(s) offered by the cloud provider can expose an organisation migrating to the public cloud to the threats in (1)?*

Chapter 6 has shown that the features in the cloud that Fix virtual IP addresses to a clouds Instance can lead to service denial attacks. An example of such feature is the Elastic IP feature .It

was shown that they need to be disassociated from the instance before shutdown. The size of an Instance can lead to a VM corruption in the cloud, exposing a user to Instance corruption i.e If a large instance is running lots of requests there is more likely to be have VM failure if there is going to be a live migration. It was also shown that using the default HTTP URL fetch in Platform services can lead to snooping attacks. SQL injection attacks are more vulnerable in a Platform as a service since they are dependent on the quality of the codes written by the programmer .

New approaches to backing up Instances were introduced in *section 6.5.2* and *section 6.3*, to minimize the instance corruption occurrence in the cloud. *Section 6.8* showed that clouds storage might not support encryption, exposing a user's data to unethical disclosure of information. Splitting the data into different data stores and combining the data sources programmatically was an approach to solving this problem. NetLib was recommended for encrypting data sent to storage, and the choice of a 3DES encryption algorithm was recommended for a platform that does not provide any encryption algorithm .The best ways of storing decryption keys for PaaS was also highlighted in *section 6.8*

3. What is a standard approach to ensure a secure migration to the cloud based on (2)?

The approach in the form of a flowchart can be seen in *Chapter 6* , and is based on a top down *transformation* layer comparison of the cloud stack model, its operation in different clouds, and *convergence* based on research papers

4. How can an applications integrity be maintained in the cloud after the migration?

Integrity refers indirectly to controlling the effect of malicious insiders, and hence to the design in *section 7.3* and *section 7.5* in *Chapter 7*. Integrity in the cloud can be maintained by encrypting the state of an instance with the public key of a trusted machine and forwarding it the cloud. The Trusted machine can then designate which machine to load an instance. Also, it was shown that by moving certain Dom 0 functions into Dom U can isolate the strength of insiders and can go along way in minimizing the malicious insider threat

8.3 Reflection

This final section of the master thesis reports reflects upon the work performed, describing the Limitations, applicability of the results, possibilities for future work and a number of final remarks.

8.3.1 Limitations

There are quite some limitations in this work. Not taken into consideration is zero down time effect during migration. The consequences of a full migration could lead to an interruption of the services to users, but a hybrid migration when properly planned could reduce the likelihood of there being a noticeable down time since services are still running in a private cloud. Other limitations are channelled towards the design in *Chapter 7* .Since this design was not implemented, some lapses not considered might show up if it were implemented.

Also, using a single machine to act as the root-of-trust if jeopardized can be disastrous to both the parties (the user and the auditors) that are dependent on it. Isolation of this machine was one key to solving this, but a total isolation of this machine in the cloud could prove difficult to

achieve in reality, as the machine would still remain physically in the cloud providers' domain. Another limitation is if a company has its customers in different regions, then there must be another machine in that region acting as the root of trust. There must then be a need for all machines acting as the root-of-trust in all regions in the cloud to communicate between one another, which was not put into consideration as part of this work. In reality a design of this scale would be very costly and could affect the overall performance of machines, as they have to be constantly communicating with the root-of-trust machine, and the cloud manager. This constant communication can cause delays and unforeseeable errors in the cloud and during the life-time of an instance.

8.3.2 Future Work and Final remarks

Cloud computing has the potential to revolutionize the way companies operate and do their business. With this new technology come new risks and threats that need to be thoroughly understood and analyzed if an organization is to migrate securely to the cloud. The opportunity exists to understand these risks and threats and build in the security to circumvent the risks and threats early in the migration and deployment phase. This work has shown the steps to migrating to the cloud in order to circumvent these risks. Although only three clouds were used, in the future more features in newer cloud infrastructure can bring newer threats, hence creating a gap between this work and the future clouds. This work should therefore be used only as a baseline framework in characterising future security threats in the next generation of cloud infrastructure .

This work has also shown how to build integrity into a clouds infrastructure, but much work remains to have a viable solution if it to be generalised and implemented on different cloud platforms and on a wider scale. The assumption that was made was all machines in the cloud need to have the TPM installed. Beyond the basic features of the machine that acts as the root-of-trust in the cloud, the additional functionality such as continuous monitoring of each instance on the machine its been loaded on, monitoring the state of the instance after a live migration etc would need to be further looked into.

A number of details still remain to be worked out regarding examining and understanding which properties are required and useful for pre measurement as part of the VM and which properties are effective in determining the configuration of a platform. Areas of research that might be interesting could include how to apply some techniques and research described in this thesis to a Software as a Service model.

Appendices

1 The Cloud Reference Model

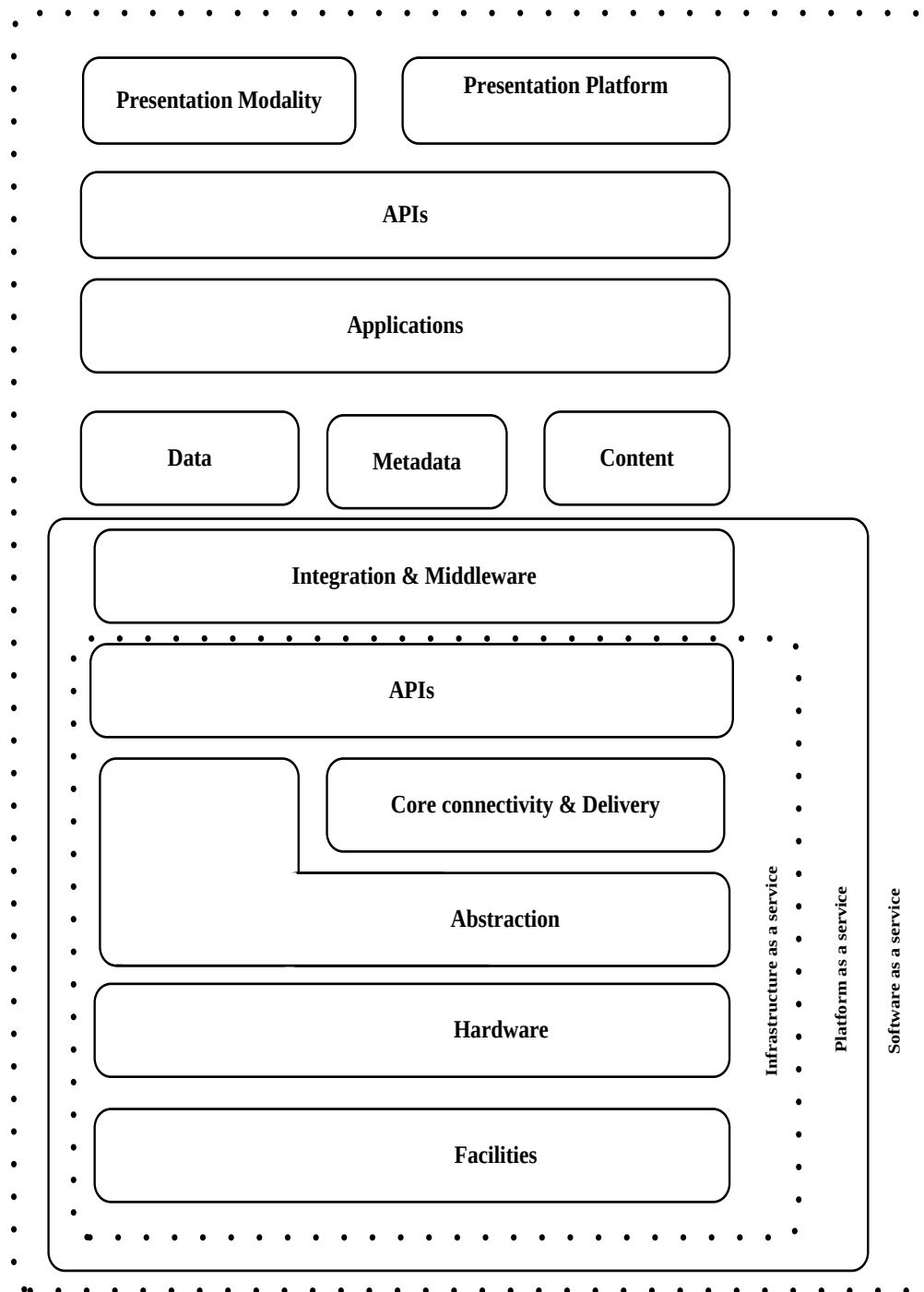


Figure 46 Cloud reference model [108,63]

The Cloud reference model shown in the *figure above* is based on the National Institute of standard and Technology (NIST) [108] and it shows the visualization of the boundaries of an *aaS (as a software) definition, how and where a particular Cloud service fits, and also how the discrete *aaS models align and interact with one another. Each *aaS layer can be seen as a self-contained stack of integrated functionality with Infrastructure as a service (IaaS) providing the foundation, and the other two models platform as a service (PaaS) and software as a service (SaaS) in turn are built upon it.

Each of the abstract layers in the reference model represents elements which when combined, comprise the services offerings in each class [108]. IaaS includes the entire infrastructure resource stack from the facilities to the hardware platforms that reside in them. IaaS incorporates the capability to abstract resources as well as deliver physical and logical connectivity to those resources. It also provides a set of API's which allow for management and other forms of interaction with the infrastructure by the consumer of the service. PaaS sits atop IaaS and adds an additional layer of integration with application development frameworks, middleware capabilities and functions such as database, messaging, and queuing that can allow developers to build applications which are coupled to the platform and whose programming languages and tools are supported by the stack. Google's AppEngine is an example of a PaaS. SaaS is built upon the underlying IaaS and PaaS stacks and provides a self-contained operating environment used to deliver the entire user experience including the content, how it is presented, and the application and management capabilities.

2 Typical Secure Cryptographic Communication

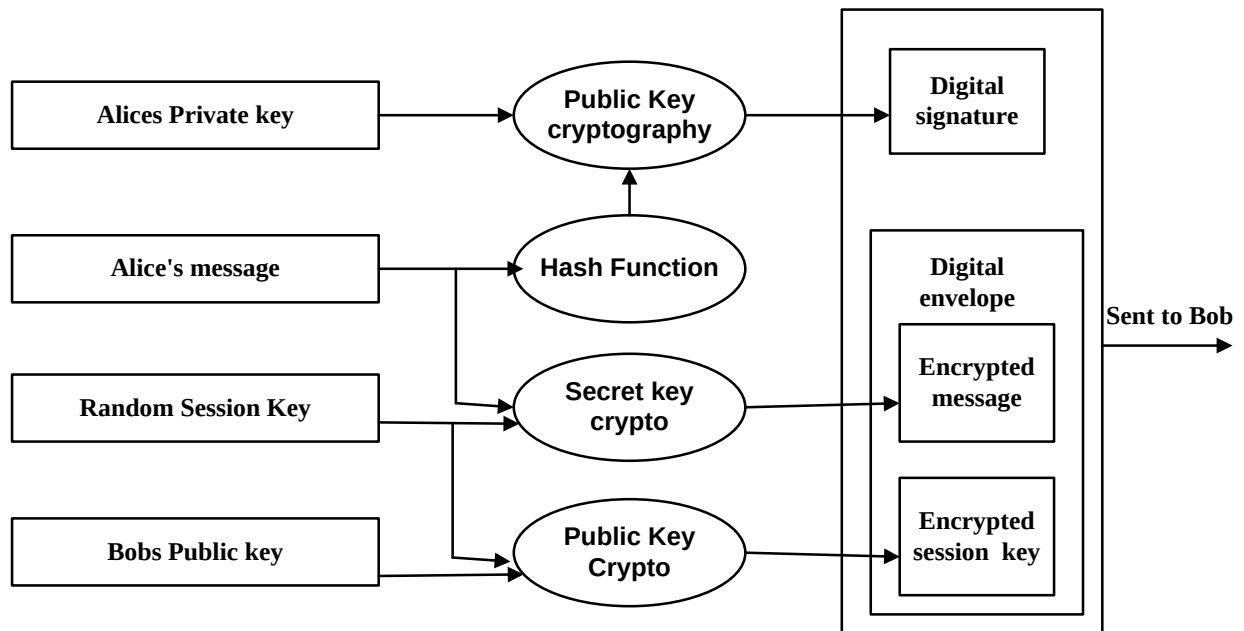


Figure 47: Secure cryptographic communication, Gary Kessler [109]

Gary Kessler [109] shows how a hybrid cryptographic scheme combines functions to form a secure transmission comprising a *digital signature* and *digital envelope*. In the example above, the sender of the message is Alice and the receiver is Bob.

The digital envelope comprises an encrypted message and an encrypted session key. *Alice* uses secret key cryptography to encrypt her message using the *session key*, which she generates at random with each session. Alice then encrypts the session key using Bob's public key. The encrypted message and encrypted session key together form the digital envelope. Upon receipt, Bob recovers the session secret key using his private key and then decrypts the encrypted message. The digital signature is formed in two steps. First, Alice computes the hash value of her message; next, she encrypts the hash value with her private key. Upon receipt of the digital signature, Bob recovers the hash value calculated by Alice by decrypting the digital signature with Alice's public key. Bob can then apply the hash function to Alice's original message, which he has already decrypted (see previous paragraph). If the resultant hash value is not the same as the value supplied by Alice, then Bob knows that the message has been altered; if the hash values are the same, Bob should believe that the message he received is identical to the one that Alice sent.

This scheme provides non repudiation since it proves that Alice sent the message; if the hash value recovered by Bob using Alice's public key proves that the message has not been altered, then only Alice could have created the digital signature. Bob also has proof that he is the intended receiver; if he can correctly decrypt the message, then he must have correctly decrypted the session key meaning that his is the correct private key.

3 Useful Commands

The commands in this section are Amazon based, and are useful if the command line tool were used. For a full list of these commands see [110]

To create a security group

PROMPT> *ec2-add-group groupname -d "group description"*

The output would resemble:

GROUP webservers My Web Server Group

To add a Security Group Rule

The command needs to be entered:

PROMPT> *ec2-authorize group [-P protocol] (-p port_range | -t icmp_type_code)
[-u source_group_user ...] [-o source_group ...] [-s source_subnet ...]*

For example, to authorize access from all IP addresses:

PROMPT> *ec2-authorize default -p 80*
PERMISSION default ALLOWS tcp 80 FROM CIDR 0.0.0.0/0

For example, to authorize access from specific IP addresses:

PROMPT> *ec2-authorize -p 22 -s <your_ip_address>/32*
PERMISSION default ALLOWS tcp 22 22 FROM CIDR <your_ip_address>/32

To lock an instance

1. The current value for the attribute can be viewed with:

PROMPT> *ec2-describe-instance-attribute instance_id*

The output would resemble:

disableApiTermination i-87ad5eec false

2. The Instance can be locked with

PROMPT> *ec2-modify-instance-attribute instance_id*

The output would resemble:

disableApiTermination i-87ad5eec true

To unlock an instance

An instance can be unlocked with:

PROMPT> *ec2-modify-instance-attribute instance_id*

The output would resemble:

DisableApiTermination i-87ad5eec false

Authorizing access to an instance

1. The ec2-authorize command is needed to allow all IP addresses access the instance through the port 80 (public web) IP address.

PROMPT> *ec2-authorize default -p 80*

GROUP default

PERMISSION default ALLOWS tcp 80 80 FROM CIDR 0.0.0.0/0

2. The public IP address of the local machine needs to be found and kept

3. The ec2-authorize command needs to be entered to open port 22 (SSH port) to that IP address.

PROMPT> *ec2-authorize default -p 22 -s your_ip_address/32*

GROUP default

PERMISSION default ALLOWS tcp 22 22 FROM CIDR your_ip_address/32

This command only allows access from the IP address. If the IP address is dynamic, then it is necessary to use this command each time it changes [110]. To allow additional IP address ranges, it is necessary to use this command for each range.

To Change the Region

By default, EC2 tools use the Eastern United States Region (us-east-1) with the ec2.us-east-1.amazonaws.com service endpoint URL.

To specify a different Region, firstly the regions can be viewed then the service endpoint needs to be set

1. To view available Regions:

PROMPT> **ec2-describe-regions**

The output would resemble:

REGION ap-southeast-1 ec2.ap-southeast-1.amazonaws.com

REGION eu-west-1 ec2.eu-west-1.amazonaws.com

REGION us-east-1 ec2.us-east-1.amazonaws.com

2. To change the service endpoint, the EC2_URL environment variable needs to be set :

C:\> **set EC2_URL=https://<service_endpoint>**

To launch an instance

1. The ec2-run-instance command is used

PROMPT> **ec2-run-instances ami-b232d0db -k gsg-keypair**

The output would resemble

RESERVATION r-f25e6f9a 999988887777 default

INSTANCE i-85b435ee ami-b232d0db pending gsg-keypair

0 m1.small 2010-03-30T08:01:36+0000 us-east-1a

2. The instance ID in the second field needs to be looked for and written down.

3. To see the launch status of the instance.

PROMPT> ***ec2-describe-instances i-85b435ee***

The output would resemble:

RESERVATION r-f25e6f9a 999988887777 default
INSTANCE i-85b435ee ami-b232d0db ec2-67-202-28-13.compute-1.amazonaws.com
domU-12-31-39-00-78-93.compute-1.internal running gsg-keypair 0 m1.small 2010-
03-30T08:01:36+0000 us-east-1a aki-94c527fd ari-96c527ff monitoring-disabled
-----Entire output deleted-----

4 Trusted Virtual Domains

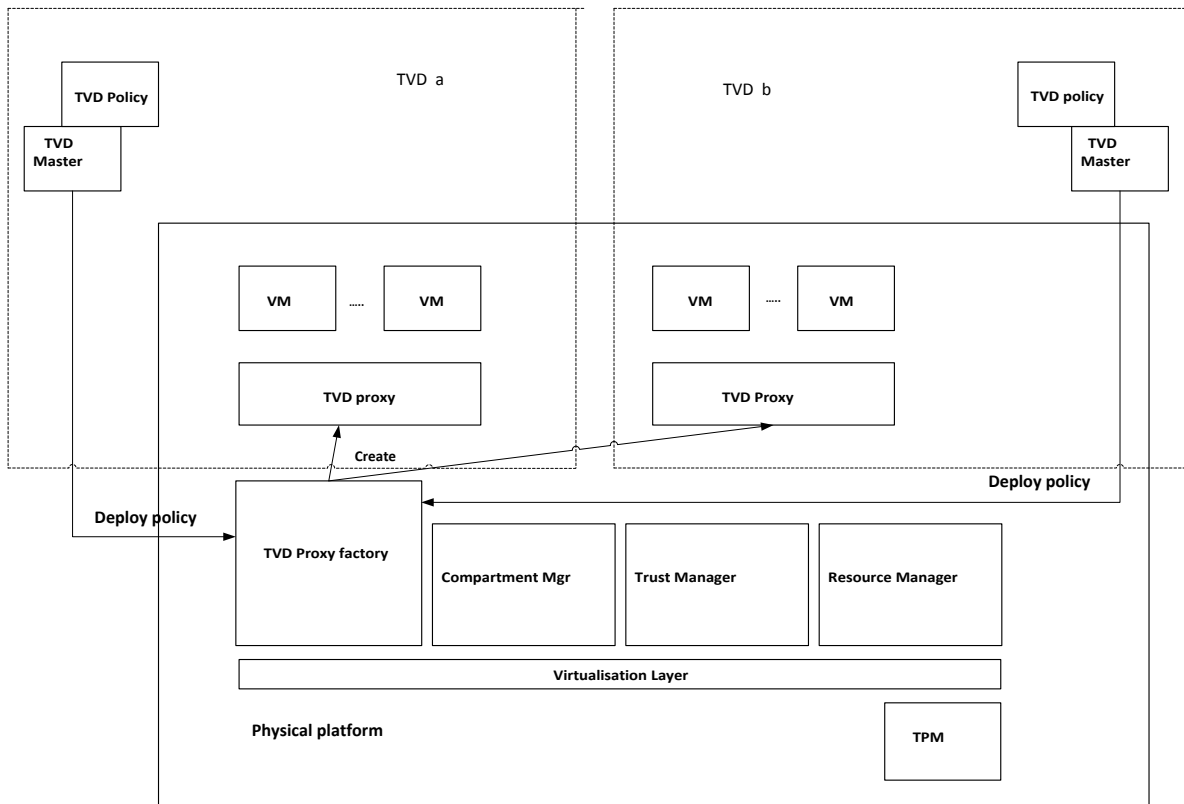


Figure 48: Trusted Virtual domains [60]

A method for ensuring centralised security with a virtual machine on different platforms or machines is with the use of a trusted virtual domain [60]. The trusted virtual data centre (TVDC)[60] helps to address the need for strong isolation and integrity guarantees in virtualized cloud computing environments. A Trusted virtual domain acts like a security domain that uniformly enforces an isolation policy across its members. The policy specifies which Virtual machines can access which resources and which Virtual machine can communicate with each other. In terms of security this goes a long way. The implementation of the TVD centre can be found in [60]. For integrity guarantees, the TVDC can leverage a virtualized root of trust to determine the identity and integrity of the software loaded on Virtual Machines (this root of trust is also based on a having a trusted computing base, which in turn is dependent on the strength of a TPM). By assigning a TVD Master and complementing it by TVD Proxies; one instance of such a proxy is then running on each physical platform hosting a TVD and then represents the local copy of the TVD Master. The TVD Master then stores and enforces the corresponding policy (TVD Policy) for admission of physical platforms, whereas TVD Proxy enforces the

policy for admission of Virtual machines. The TVD Master is supposed to be like a logical service and could be implemented either on a single server or in distributed way, as shown above

Each platform (as shown above) has one TVD Proxy for each TVD. So before a Virtual machine running on the TVD Platform can join a TVD, the corresponding TVD Proxy has to be instantiated on the platform. During this instantiation, TVD Master deploys the TVD Policy to a local TVD Proxy. After deployment, TVD Proxy enforces the admission of Virtual machines to the TVD locally on the respective platform. With the compartment manager being responsible for managing virtual machines and their attributes[110], The Trust manager providing abstraction of the underlying trusted computing functionality in hardware and the Resource manager providing access to basic virtualised resources like networking and storage. More work on TVDs can be found in [61]

References

- [1] R. Goldberg, *Architectural Principles for Virtual Computer Systems*. PhD thesis, Harvard University, 1972.
- [2] R. Goldberg, *Survey of virtual machine research*. IEEE Computer Magazine, 7: pg 34–45, June 1974.
- [3] E. Bugnion, S. Devine, and M. Rosenblum. Disco: *Running commodity operating systems on Scalable multiprocessors*. In Proceeding of 16th ACM Symp. Operating Sys. Principles, Oct. 1997.
- [4] J. Sugerman, G. Venkitachalam, and B. Lim. *Virtualizing I/O devices on VMware Workstation's hosted virtual machine monitor*. In Proc. 2001 Ann. USENIX Tech. Conf., Boston, MA, USA, June 2001.
- [5] C. A. Waldspurger. *Memory resource management in VMware ESX Server*. In Proc. 2002 Symp. Operating Sys. Design and Implementation, December 2002
- [6] S. Campbell, and M. Jeronimo, *Applied Virtualization Technology*, Hillsboro, Intel Press, 2006.
- [7] P. Barham, B. Dragovic, K. Fraser, et al., *Xen and the Art of Virtualization*, ACM SIGOPS Operating Systems Review, vol. 37, no. 5, pp. 164-177, 2003.
See <http://cgi.di.uoa.gr/~charnik/files/xen.pdf>
- [8] S. Berger, R. Caceres, D. Pendarakis, et al., *TVDc: Managing Security in the Trusted Virtual Datacenter*, ACM SIGOPS Operating Systems Review, vol. 42, no. 1, pp. 40-47, January, 2008.
- [9] D. Grawrock, *The Intel Safer Computing Initiative*, Hillsboro, OR: Intel Press, 2006.
- [10] D. Nurmi, R. Wolski, C. Grzegorzcyk, et al., *Eucalyptus: A Technical Report on Elastic Utility Computing Architecture Linking Your Programs to Useful Systems*, Technical Report 2008-10, University of California Santa Barbara Computer Science, Santa Barbara, CA, 2008.
- [11] *Amazon Elastic Compute Cloud* (Amazon EC2), see <http://aws.amazon.com/>
- [12] S. Pearson, *Trusted Computing Platforms: TCPA Technology in Context*, Upper Saddle River, NJ: Prentice Hall, 2003.
- [13] *TCG Specification Architecture Overview*, vol. 1.4, Trusted Computing Group, 2007;
See http://www.trustedcomputinggroup.org/resources/tcg_architecture_overview_version_14.

- [14] D. Grawrock, *Dynamics of a Trusted Platform*, Hillsboro, OR: Intel Press, 2009.
- [15] *Trusted Computing Group*, TPM Specification Version 1.2 Revision 103, Trusted Computing Group, 2007;
http://www.trustedcomputinggroup.org/resources/tpm_main_specification.
- [16] S. Berger, R. Cáceres, K. A. Goldman, et al., vTPM: *Virtualizing the Trusted Platform Module*, in 15th USENIX Security Symposium, Vancouver, B.C., 2006.
- [17] *Exploring the Integration of Memory Management and Trusted Computing*, Nihal A.D' Cunha, Dartmouth Computer Science Technical Report TR2007-594, 2007
- [18] P. M. Chen and B. D. Noble. *When virtual is better than real*. In Proc. 2001 Workshop on Hot Topics in Operating Sys. (HotOS-VIII), Schloss Elmau, Germany, May 2001.
- [19] Tal Garfinkel, Ben Pfaff Jim, Chow Mendel, Rosenblum Dan Boneh, *Terra: A Virtual Machine-Based Platform for Trusted Computing*
- [20] Ben Smyth, Mark Ryan, and Liqun Chen, January 9, 2008, Direct Anonymous Attestation (DAA): *Ensuring privacy with corrupt administrators*
- [21] Ronald Toegl and Michael Hutter, 2010, *An approach to introducing locality in remote Attestation using near field communications*
- [22] Frank John Krautheim, PhD work: *Building trust into utility cloud computing*, 2010
- [23] *Credentials and Beliefs in Remote Trusted Platforms Attestation*, Andrea Bottoni, Evangelos Kranakis, Proceedings of the 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks
- [24] *Beyond Secure Channels*, Yacine Gasmi, Ahmad-Reza Sadeghi, Patrick Stewin
- [25] *Remotely Keyed Cryptographic Secure Remote Display Access Using (Mostly) Untrusted hardware*, Debra L. Cook Ricardo, Baratto Angelos D. Keromytis, 2009
- [26] The web hacking Incidents database 2008, Breach Annual report , February 2009
See http://www.breach.com/resources/whitepapers/downloads/WP_WebHackingIncidents_2008.pdf
- [27] *Anatomy of an ARP Poisoning Attack*. By Corey Nachreiner, WatchGuard Network Security Analyst;
See: http://www.softpanorama.org/Net/Internet_layer/arp.shtml#ARP_vulnerabilities
- [28] *Towards Secure E-Commerce Based on Virtualization and Attestation Techniques*, Frederic Stumpf, Claudia Eckert, Shane Balfe, 2010
- [29] *Security evaluation of scenarios based on the TCG's TPM Specification*, Sigrid Gurgens,

Carsten Rudolph, Dirk cheuermann, Marion Atts, and Rainer Plaga, 2008

- [30] *Introducing windows Azure*, David Chappell and Associates, march 2009
- [31] *Trust Visor: Efficient TCB Reduction and Attestation*, Jonathan M.McCune, Ning Nu, Yanlin Li, 2009
- [32] *Efficient remote attestation framework for dynamics systems*, wenjuan Xu, Gail-Joon Ahn, HongXin Hu, Xinwen Zhang, 2008
- [33] *Information Security in the Clouds*, Cloud computing series, Donald Faatz, Lawrence Pizette, page 3
- [34] *Cloud computing with the Azure platform*, Jouni Mäenpää, Helsinki University of Technology
- [35] *Trusted distribution and Utilization of Virtual machines*, Daniel Kouril, Ludek Matyska, Michal Prochazka, 2006
- [36] Halдар, V., Chandra, D., Franz, M., *Semantic remote attestation: A virtual machine directed Approach to trusted computing*. In: USENIX Virtual Machine Research and Technology Symposium. 2004
- [37] D. Grawrock, *The Intel Safer Computing Initiative*, Hillsboro, OR: Intel Press, 2006.
- [38] Chen, L., Landfermann, R., Loehr, H., Rohe, M., Sadeghi, A.R., Stübke, C. *A protocol for Property-based attestation*. In: STC '06: Proceedings of the First ACM Workshop on Scalable Trusted Computing, ACM 2006
- [39] Torry Harris, *Cloud Computing Services – A comparison*, 2007
- [40] Kuhn, U., Selhorst, M., Stübke, C. *Realizing property-based attestation and sealing with Commonly available hard- and software*. In: STC '07: Proceedings of the 2nd ACM Workshop on Scalable Trusted Computing, ACM Press 2007
- [41] *Extending IPSec for Efficient Remote Attestation*, Ahmad-Reza Sadeghi, Steffen Schulz, 2009
- [42] *Amazon web service*, Amazon Elastic Compute Cloud, User Guide, API Version 2010/08/31
- [43] Locating the datacenters in Amazon,
See :<http://www.readwriteweb.com/cloud/2010/08/a-map-to-your-nearest-data-cen.php>
- [45] Amazon and Datacenters: weblink :
<http://www.cloudtweaks.com/2010/04/amazon-aws-adds-singapore-data-center-cloud-news/>
- [46] *Trusted computing group*, Main specification: trusted computing platform alliance

(TCPA), 2003

- [47] Trusted computing (TC) information society technologies, *DO4.7 vTPM Architecture*, 2005
- [48] IBM T J Watson research center, *Virtualization and the trusted platform module*, Ronald Perez, 2006
- [49] *Virtualized Trusted computing platform for adaptive security Enforcement of web services Interactions*, Ivan Djordjevic, Srijith K.Nair, Theo Dimiittrakos, 2007
- [50] Network security and cryptography, A review of VTPM: *Virtualizing the trusted platform Module*, Rich Gotyette, 2007
- [51] *Improving the Live Migration process of large Enterprise Applications*, Stuart hacking , benoit Hudzia, 2009
- [52] *Amazon web service*, see <http://aws.amazon.com/importexport/>
- [53] *Survey of virtual machine migration techniques*, sharath venkatesha, shatrugna sadhu, Sridhar kintakli, 2009
- [54] *Empirical exploitation of live virtual machine migration*, Jon Oberheide, Evan Cooke, Farnam jahanian, 2008
- [55] DNS cache poisoning: *Definition and Prevention*, Tom Olzak, 2006
- [55] VTPM: Virtualising the trusted platform module, Stephan Berger, Reiner Caceres, Kenneth A. Goldman, 2006
- [56] *Towards trustworthy virtualisation environments: Xen library OS security service Infrastructure*, Melvin J.Anderson, Micha Moffie, Chris I Dalton, 2007
- [57] *D04.7 VTPM Architecture*, Open Trusted Computing group, 2009
- [58] *Property based attestation and trusted Computing: Analysis and challenges*, Aarthi Nagarajan, Vijay varadharajan and Michael hitches (2009) third international conference On Network and system security
- [59] *Flexible Integrity protection and Verification Architecture for virtual machine monitors*, Bernhard Jansen, Hari Govind V. Ramasamy, and Matthias Schunter, IBM Zurich research Laboratory
- [60] I.S Berger, R. Ca ceres, D. Pendarakis, R.sailer, E. Valdez, R. Perez, W, shhilhauer, and D. Srinivasan, *TVDc: Managing security in the trusted virtual data centre*, operating Systems rev. 42, No. 1, 40-47 2008

- [61] Serder cabuk, Chris I Dalton, Konrad Eriksson, Dirk Kuhlmann, HariGovind V, Ramasamy c, Gianluca Ramunnod, *Towards automated security policy enforcement in Multi-tenant virtual data centres*, 2010
- [62] Finger, S. and J. Dixon “A review of research in mechanical engineering design. Part I: Descriptive, prescriptive, and computer-based models of design processes.” Research In engineering design 1: 51-67, 1989
- [63] IA Newsletter, The Newsletter for Information Assurance technology Professionals, *cloud Computing: Silver lining or Storm Ahead*
- [64] Cloud security Alliance: *Top Threats to Cloud Computing v1.0*, The cloud security alliance, 2010
- [65] Information technology - *Open Systems Interconnection - Upper layers security model*, International standard ISO/IEC 10745
- [66] Dlamini M T, Eloff M and Eloff J H P, ‘Internet of People, Things and Services – *The Convergence of Security, Trust and Privacy*’, 2009.
- [67] *The OpenNebula Project*. OpenNebula. see <http://www.opennebula.org>.
- [68] *Eucalyptus Systems*, Inc. Eucalyptus. See <http://open.eucalyptus.com/>.
- [69] *The Citrix Open Cloud Bridge*, White Paper, see www.citrix.com
- [70] *A Best Practice Model for Cloud Middleware Systems*, Ajith Ranabahu¹ and E. Michael Maximilien
- [71] Brodtkin, J. (2008, July 02). *Gartner: Seven cloud computing security risks.*, see InfoWorld: <http://www.infoworld.com/d/security-central/gartner-seven-cloud-computing-security-risks-853>
- [72] Tout, Sverdlik, and Lawver, *Cloud Computing and its Security in Higher Education*, 2009
- [73] Exact online, see <http://www.exact.nl/>
- [74] Cloud computing: *The Emerging Computing technology*, Feng-Tse Lin and Teng-San Shih, July 2010
- [75] *Design Methods: seeds of human futures*, John Chris Jones 1970, John Wiley and Sons, New York and Chichester
- [76] *A survey of design philosophies, models, methods and systems*, N. Evbuomwan, S Sivaloganathan, 1996
- [77] Right scale, *Top reasons Amazon EC2 instances disappear*,
See also <http://blog.rightscale.com/2008/02/02/top-reasons-amazon-ec2-instances-disappear/>

- [78] *Live Migration of Virtual Machines*, Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, Andrew Wareld, 2000
- [79] NIST Glossary of Key Information security terms,
See <http://csrc.nist.gov/publications/nistir/ir7298-rev1/nistir-7298-revision1.pdf>
- [80] *Not all clouds are created Equal*, An Expedient Communications White Paper in Cloud Computing Services, January 2010
- [81] National Institute of standards and Technology, *Guidelines on Security and Privacy in Public Cloud Computing*, Wayne Jansen and Timothy Grance, 2011
- [82] *Threat Modeling Web-Applications using STRIDEs Average Model*, John Peter Jesan, 2007
- [83] *The Azure cloud*, see <http://www.microsoft.com/windowsazure/whitepapers/default.aspx>
- [84] *Start building and Running Web Apps on Google's Infrastructure*, O'reilly Google Press, Charles severance
- [85] Google Application Engine documentation,
See: <http://code.google.com/appengine/docs/java/gettingstarted/>
- [86] *Scalable SOA in Your Own Open Source Cloud*, Sanjiva Weerawarana, 2009
- [87] *Hyper-V: Requirements for an Integrity-protected Hypervisor on the x86 Hardware Virtualized Architecture*, Amit Vasudevan, Jonathan M. McCune, Ning Qu, Leendert van Doorn and Adrian Perrig,
See <http://www.springerlink.com/content/v5n360617428j685/fulltext.pdf>
- [88] *Active Man in the Middle Attacks: A security advisory*, A whitepaper from IBM Rational Application Security Group, Roi Saltzman, Adi Sharabani, February 27, 2009
- [89] *Security Guidance for Critical Areas of Focus in Cloud Computing V2.1*, Cloud security Alliance, 2009
- [90] Linthicum D. S. , *Cloud Computing and SOA Convergence in Your Enterprise: A Step-by-Step Guide*, Addison-Wesley Professional, US, 2009
- [91] *A Transactional Application Profiling Environment (TAPE)*, Hassan Chafi, Chi Cao Minh, Austen McDonald, Brian D. Carlstrom, JaeWoong Chung, Lance Hammond, Christos Kozyrakis, and Kunle Olukotun, 2008
- [92] *Data de-duplication for dummies*, Mark R. Coppock Steve Whitner, 2009
- [93] *Database Encryption software*, NetLib: <http://www.netlib.com/>

- [94] Amazon web services, *migrating your existing application to the AWS cloud: A phase driven approach to Cloud migration*
- [95] *Amazon Simple Queue Service: Getting Started Guide API Version*, 2009-02-01, Amazon web services
- [96] *Developing with GoogleApp Engine*, Eugene ciurana, 2009
- [97] *HTTP Digest Integrity: Another look, in light of recent attacks*, Version 1.0, Timothy D. Morgan, January 5, 2010
- [98] *Bootstrapping Open Stack Clouds*, platform and Infrastructure for Hyper scale Environments: A dell Technical paper, by Rob Hirschfeld and Greg Althaus
- [99] *Hybrid Clouds: Comparing Cloud Toolkits*, Seminar Paper, Prof. Dr. Martin Schader, April 2010
- [100] *Introducing the Azure services platform*, David Chappell and Associates, 2008
- [101] Torry Harris, *Cloud Computing Services – A comparison*, 2007
- [102] *Amazon Virtual Private Cloud*, Getting Started Guide API Version 2011-02-28, Amazon web services, 2011
- [103] Microsoft Azure: Enterprise Application Development: Chapter No.6, *Azure Blob Storage*, Richard J. Dudley, Nathan A. Duchene, 2009
- [104] Post-Copy Live Migration of Virtual Machines, Michael R. Hines, Umesh Deshpande, and Kartik Gopalan Computer Science, Binghamton University (SUNY), 2009
- [105] *The Xen Users Manual*, Xen v3.3, 2002
- [106] *Semi-invasive attacks: A new approach to hardware security analysis*, Sergei P. Skorobogatov, April 2005
- [107] Accenture Technology Labs,
See: http://blogs.accenture.com/technology_labs_blog/archive/2010/06/16/amazon-cloud-security-best-practices.aspx
- [108] *Cloud Architecture Reference Models: A survey*, NIST CCRATWG 004 v2, January 25, 2011
- [109] *An Overview of Cryptography*, Gary C. Kessler, 21 March 2011
- [110] *Amazon Elastic Compute Cloud, API Tools Reference*,

<http://docs.amazonwebservices.com/AWSEC2/latest/CommandLineReference/>

- [111] *Securing the Cloud: A Review of Cloud Computing, Security Implications and Best Practices*, VMware Whitepaper
- [112] Cloud Security, *Crypto Services and Data Security in Windows Azure*, see MSDN link <http://msdn.microsoft.com/en-us/magazine/ee291586.aspx>
- [113] Michael Howard. *Security Development Lifecycle: Development Practices*.
See <http://download.microsoft.com/download/8/4/7/8471a3cb-e4bf-442a-bba4-c0c907d598c5/Michael%20Howard%20SDL%20Development%20Practices.ppsx>
- [114] Electronic Frontier Foundation. *Cracking DES*.
see http://w2.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/HTML/19980716_eff_des_faq.html
- [115] Crypto Manifesto: Security Research Group Fortify Software, Yekaterina sipenyuk O'Neil
, March 12, 2009
- [116] *Analysis of Integrity Vulnerabilities and a Non-repudiation Protocol for Cloud Data Storage Platforms*, Jun Feng, Yu Chen, Wei-Shinn Ku, Pu Liu

Acronyms

AIK-Attestation Identity Key
AWS- Amazon web service
AMI-Amazon machine Image
API-Application programming Interface
DMZ- Demilitarized Zone
DNS-Domain Name System
DNSSEC- Domain Name System Security Extensions
DAAS-direct anonymous attestation scheme
Dom 0- Domain 0 (Privileged mode)
Dom U-Domain U (Unprivileged mode)
EBS-Elastic Block Store
EC2 -Elastic compute cloud
EIP-Elastic IP
EOL-Exact Online
EK-Endorsement Key
ECU- Elastic compute unit
HTTP-Hypertext transfer protocol
HTTPS- Hypertext Transfer Protocol Secure
IP-Internet protocol
IPSec-Internet Protocol security
S_{vm}-session key
ML-measurement list
MLS-Multi-Level Security
NVRAM -Non-Volatile Random Access Memory
PCR-Platform configuration register
PDP-policy decision point
Private virtual infrastructure factory (PVIF)
RTM-Root of Trust for Measurement
RTR- Root of Trust for Reporting
RTS-Root of Trust for Storage
RDS-Relational Database Service
S3 – Simple storage service
SSH-Secure Shell
SRK-Storage Root Key
SQS-Simple queuing service
SHA-1 -Secure Hash Algorithm hash
TPM-trusted platform module
TPM-Trusted Platform Module
TCG-The Trusted Computing Group's
TCP-Trusted Computing Platform
TVMM-trusted virtual machine monitor

TK-trusted key (public or private)
TTP-A trusted third party
VPC-virtual Private Cloud
VPN- virtual Private Network
VM-Virtual machine
VTPM-Virtual TPM
VMM-Virtual machine monitor
VS-Verification server
Virtual server domain (vsDom)
vPCR -virtual PCR
vEK- Virtual Endorsement Key
vAIKs - Virtual Endorsement Key