Securing Access to Cloud Computing for Critical Infrastructure

Younis A. Younis

A thesis submitted in partial fulfilment of the
requirements of Liverpool John Moores University
for the degree of Doctor of Philosophy

September/2015

Abstract

Cloud computing offers cost effective services on-demand which encourage critical infrastructure providers to consider migrating to the cloud. Critical infrastructures are considered as a backbone of modern societies such as power plants and water. Information in cloud computing is likely to be shared among different entities, which could have various degrees of sensitivity. This requires robust isolation and access control mechanisms. Although various access control models and policies have been developed, they cannot fulfil requirements for a cloud based access control system. The reason is that cloud computing has a diverse sets of security requirements and unique security challenges such as multi-tenant and heterogeneity of security policies, rules and domains.

This thesis provides a detailed study of cloud computing security challenges and threats, which were used to identify security requirements for various critical infrastructure providers. We found that an access control system is a crucial security requirement for the surveyed critical infrastructure providers. Furthermore, the requirement analysis was used to propose a new criteria to evaluate access control systems for cloud computing. Moreover, this work presents a new cloud based access control model to meet the identified cloud access control requirements. The model does not only ensure the secure sharing of resources among potential untrusted tenants, but also has the capacity to support different access permissions for the same cloud user.

Our focused in the proposed model is the lack of data isolation in lower levels (CPU caches), which could lead to bypass access control models to gain some sensitive information by using cache side-channel attacks. Therefore, the thesis investigates various real attack scenarios and the gaps in existing mitigation approaches. It presents a new Prime and Probe cache side-channel attack, which can give detailed information about addresses accessed by a virtual machine with no need for any information about cache sets accessed by the virtual machine. The design, implementation and evaluation of a proposed solution preventing cache side-channel attacks are also presented in the thesis. It is a new lightweight solution, which introduces very low overhead (less than 15,000 CPU cycles). It can be applied in any operating system and prevents cache side-channel attacks in cloud computing. The thesis also presents a new detecting cache side-channel attacks solution. It focuses on the infrastructure used to host cloud computing tenants by counting cache misses caused by a virtual machine. The detection solutions has 0% false negative and 15% false positive.

Acknowledgements

It is a pleasure to thank my Mother and Father, who made this thesis possible and gave me everything they have got in order to support me to reach this stage.

I offer my regards and blessings to my wife who supported me in any respect during the completion of the project. She has been always behind me.

I am heartily thankful to my director of study Dr. Kashif Kifayat, whose encouragement, supervision and support from the preliminary to the concluding level enabled me to develop an understanding of the subject. His help and guidance have kept me focused and enabled me to successfully complete this project.

I would like to give sincere thanks to Professor Madjid Merabti, who offered me many words of encouragement, support and advice throughout my project. I would like also to thank Professor Qi Shi and Dr. Bob Askwith for their support and help.

Lastly, I would like to thank all my friends, who supported and advise of me through the project.

Contents

List of Diagrams

List of Tables

List of Publications

Y. A. Younis, K. Kifayat, and M. Merabti, "An Access Control Model for Cloud Computing," J. Inf. Secur. Appl., vol. 19, no. 1, pp. 45–60, Feb. 2015

A. Younis, Y., Kifayat, K., Shi, Q., & Askwith, B. A New Prime and Probe Cache Side-Channel Attack for Cloud Computing. In the 13th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC-2015) (p. 7), Liverpool UK. 2015.

A. Younis, Y., Kifayat, K., & Merabti, M. A Novel Evaluation Criteria to Cloud Based Access Control Models. In *The 11th IEEE International Conference on Innovations in Information Technology (IIT'15)* (p. 6), Dubai, UAE. 2015.

A Younis, Y., Kifayat, K. & Merabti, M., 2014. Cache Side-Channel Attacks in Cloud Computing. The Second International Conference on Cloud Security Management ICCSM-2014. (p. 10), Reading, UK. 2014.

A. Younis, Y., Merabti, M. & Kifayat, K. Cloud Computing Security & Privacy Challenges. In The 15th annual post graduate symposium on the convergence of telecommunications, networking and broadcasting. p. 6, Liverpool, UK. 2014.

A. Younis, Y., Merabti, M. & Kifayat, K. Secure Cloud Computing for Critical Infrastructure: A Survey. In The 14th annual post graduate symposium on the convergence of telecommunications, networking and broadcasting. Liverpool: Liverpool John Moores University, p. 6, Liverpool, UK. 2013.

CHAPTER 1 Introduction

This chapter gives a brief introduction to cloud computing in terms of services and security challenges. It also provides basic information about critical infrastructure, its importance to our daily life and its security requirements. In addition, it covers the importance of access control systems to data security and why conventional access control models cannot be deployed in their current state in the cloud. Some information about access control systems' security attacks in general and cache side-channel attacks in particular are also presented here. This chapter also shows our aims and objectives, the motivation behind this project, the novel contributions and the structure of the thesis.

1.1 Cloud Computing

Cloud computing is an open standard model, which is Internet-centric and provides various services either software or hardware. It offers new cost effective services on-demand such as Software as a service (SaaS), Infrastructure as a service (IaaS) and Platform as a service (PaaS). A significant interest in both industry and academia has been generated to explore and enhance cloud computing. It has five essential characteristics: on-demand self-service, measured service, rapid elasticity, broad network access and resource pooling. It is aiming at giving capabilities to use powerful computing systems with reduced cost, increased efficiency and performance [1]. It consolidates the economic utility model with the evolutionary enhancement of many utilised computing approaches and technologies, which include computing infrastructure consisting of networks of computing and storage resources, applications and distributed services. Moreover, there is an ongoing debate in Information Technology (IT) communities about how the cloud computing paradigm differs from existing models and how these differences affect its adoption. One view considers it as a modern or a fashionable way to deliver services over the Internet, while others see it as a novel technical revolution [2]. However, they have agreed that cloud computing gives a new hope for meeting various requirements of service providers and consumers as well. Hence, governments, enterprises and even critical infrastructure providers are considering migrate to cloud computing.

1.2 Critical Infrastructure

Critical infrastructures are a crucial assets for the functioning of vital societal services such as power distribution networks and financial systems [3]. The importance of critical infrastructures is hard to overestimate, and even relatively minor failures can impact on a large number of people e.g. on 31st March 2015 Turkey has suffered a huge power cut affecting almost the whole country [4] due to a cyber-attack. This unexpected power cut caused complete shutdown to public transport and big disruption to air traffic controllers. This incident highlights the importance of keeping such systems running. Failures can result in serious consequences for the functioning of society. It is an alarming situation as an attacker can use a laptop or computer to disrupt the national infrastructures.

Critical Infrastructure Providers (CIP) have started to use the cloud computing due to many benefits. However, there are still many challenges such as multi-tenant billing, virtualization with very strong isolation, Service Level Agreements (SLA), definitions and automatic enforcement mechanisms, end-to-end performance and security mechanisms. One of the objectives of this project is to investigate cloud computing security issues, which hinder migration of CIP to the cloud and also perform requirement analysis for different CIP users to utilise the cloud.

1.3 Cloud Computing Security Challenges

With all of these promising facilities and benefits that cloud computing can offer, there are still a number of technical barriers that may prevent cloud computing from becoming a truly ubiquitous service. Security is the main inhibitor to cloud adoption. Cloud computing may inherit some security risks and vulnerabilities from the Internet, such as malicious code (Viruses, Trojan Horses). In addition, cloud computing suffers from data privacy issues and conventional distributed systems attacks i.e. Distributed Denial of Service attacks (DDoS), which could have a huge impact on its services. Moreover, cloud computing has brought new concerns such as moving resources and storing data in the cloud with a probability of residing in another country with different regulations. Computing resources could be inaccessible due to many reasons such as natural disaster or denial of service.

Cloud computing services may be delivered by a large number of service providers. They use various types of technologies, which may cause heterogeneity problems [5]. Extensibility and shared responsibilities is another concern as up to now it is not clear, how security duties should be assigned in cloud computing and who is responsible for what [6]. Furthermore, virtualization is one of many ways used in cloud computing to meet their consumer necessities, but it brings its own threats such as penetrating the logical isolation between virtual machines and side-channels [7].

Cloud computing provides the perfect platform to lunch cyber-attacks. Many of these attacks are commonly encountered in the wider Internet such as Distributed Denial-Of-Service (DOS) attack [8], insecure application programming interface [9], abuse and nefarious use of cloud computing [9], malicious insiders [9], and access control issues.

1.4 Access Control

An access control system is a crucial requirement to secure assets against unauthorised access in a cloud environment. However, there are still many questions which raise issues e.g. service providers in cloud computing are likely to be outside the trusted domain of users and resided in an another country with different law and regulations [10]. Furthermore, access control systems can be more complex and sophisticated due to dynamic resources [11] heterogeneity [12] and diversity of services [13].

There are various traditional access control models and policies for different environment such as Mandatory Access Control (MAC) model, Discretionary Access Control (DAC) model and Role Based Access Control (RBAC). These models cannot fulfil cloud’s access control requirements due to a diverse set of users and platforms with different sets of security requirements. More details can be found in chapter 2 section 2.2.1. Cloud computing also has unique security challenges such as multi-tenant hosting and heterogeneity of security policies, rules and domains. We believe these models may not fully cover access control requirements of cloud computing as each one of them has been proposed for a specific environment to fulfil consumers’ access control requirements.

Deploying a robust access control model would make cloud service providers and their clients confident about confidentiality and integrity of their data, yet there are a number of common techniques and attacks that can be used to gain unauthorized access to data such as session hijacking or TCP hijacking. In addition, password attacks such as dictionary attack are still used to bypass access control systems. All of the mentioned attacks target the application level. However, there are number of attacks at lower levels such as CPU caches which can have severe impacts on data access control. One of these attacks is a side-channel. Some side-channels can be used without interfering with other cloud tenants’ customers to gain some sensitive information, for instance, cache side-channel attacks [14].

1.5 Cache Side-Channel Attacks

Cloud computing supports multi-tenancy to fulfil future increasing demands for accessing and using resources provided over the Internet. Multi-tenancy enables the sharing of computing physical resources among cloud computing tenants and offers cost-effective on-demand scaling. However, multi-tenancy in cloud computing has unique vulnerabilities such as clients' co-residence and virtual machine physical co-residency. Physical co-residency of virtual machines in cloud computing has been exploited to leak sensitive information and launch sophisticated security attacks. It can facilitate attackers with an ability to interfere with another virtual machine running on the same physical machine due to an insufficient logical isolation. In the worst case scenario, attackers can extract sensitive data using hardware side-channels on the same physical machine.

Side-channel attacks exploit the correlation between the higher level functionality of the software and the underlying hardware phenomena. There are various types of side-channel attacks, which are classified according to the hardware medium they target and exploit, for instance, cache side-channel attacks. CPU caches are one of the most common hardware devices targeted by adversaries because they have high-rate interactions and sharing between processes. Furthermore, full encryption keys of well-known algorithms (i.e. RSA and AES) have been broken using spying processes to spy and collect information about cache lines, which have been accessed [2,3]. This information is analysed and linked to the current virtual machine, which occupies the processor.

There are number of proposed solutions to prevent and detect cache side-channel attacks in cloud computing [17,18,19,20,21]. Firstly, the prevention solutions can be either hardware or software based. Although hardware-based techniques seem to be more secure to implement as they are more efficient and thwart the root cause of those attacks, they cannot be practically applied until CPU makers implement them into CPUs. Furthermore, software-based mitigation techniques are attack-specific solutions, which can only tackle attacks that they are proposed for. Consequently, these solutions might not have the ability to mitigate new side-channel attacks.

Secondly, most of the detection tools fail due to the deceived normal behaviour of cache side-channel. On the other hand, detecting solutions mainly rely on software and applications to detect any abnormal behaviour on the CPU cache. These applications and software have to will slow down the CPU and caches' operations and introduce unwanted overload, which will affect the CPU performance.

1.6 Motivations for the Project

The motivation for this project came from the author's quest to understand the cloud computing security challenges and threats; and analyse the level of security access provided to either cloud service providers or their tenants in order to secure access to the data. This included investigation of security requirements for different critical infrastructure service providers. Among many security requirements, a cloud based access control model has been found one of the crucial security requirements. The author conducted a detailed requirement analysis for proposing a cloud based access control model that can fulfil the security requirements of cloud computing services. Data in a cloud computing environment will be shared among different entities. These entities have various degrees of sensitivity. Current access control models suffer from semantic and heterogeneity problems, which affect used policies and methods used to translate the policies. A cloud based access control model has to insure access to data in all computation levels from users' levels to lower levels; and deal with data leakage at lower level such as CPU caches. However, the used access control models either the conventional access control models or the proposed for cloud computing cannot control access to data at the lower level resources such as RAM and CPU caches. Therefore, this motivates the author to investigate the effects of cache side-channel attacks on cloud computing and available proposed solutions to mitigate them.

In addition, the author's knowledge about securing access to cloud computing is intended to be enhanced throughout this multifaceted project. With very little experience in extensive research, and also the techniques employed in solving the issues, it is believed that this project will pose some significant challenges. However, the author believes that the outcome will be worthwhile as it would contribute to global knowledge and also demonstrate how cloud computing can be utilised securely to offer secure access to its services for tenants such as critical infrastructure providers.

1.7 Aims and Objectives of the Project

This project aims to propose an access control model for cloud computing to meet the security requirements of cloud service providers and their customers such as critical infrastructure providers. It will also enhance the level of security of accessing data in lower levels such as

CPU caches by providing detecting and prevention solutions to any leakage of information by cache side-channel attacks.

The key objective is to produce a cloud based access control model, through exploratory research that results in achieving these aims. The purpose of this research is:

1. To identify cloud computing security challenges, concerns and threats for cloud service providers and their customers.

2. To perform critical infrastructure providers’ security requirements analysis for different critical infrastructure providers.

3. To identify gaps in conventional access models (RBAC, MAC, etc.).

4. To perform a requirement analysis for cloud based access control systems and provide new evaluation criteria for cloud based access control systems.

5. To provide a simplified output model where data can be secured from unauthorized access at all levels from application levels to processing levels.

6. To identify threats of side-channel attacks that can be used to bypass isolation levels enforced by access control policies.

7. To provide an efficient detection solution to identify cache side-channel attacks in cloud computing without affecting the performance or asking for any modifications to customers’ operating systems.

8. To facilitate cloud computing service providers with a prevention solution that prevents cache side-channel attacks without affecting the CPU caches’ performance.

9. To implement the proposed solutions in an identical environment to cloud computing with various number of virtual machines.

10. To compare the obtained results from the proposed solutions with various results from other prevention and detection cache side-channel attacks solutions that are implemented in the same environment.

11. To publish the results of this project in well-regarded peer reviewed international journals and conferences.

1.8 Contributions

In this project, we have fulfilled all the stated objects in section 1.7. We started looking at cloud computing security challenges for cloud service providers and their customers such as critical infrastructure providers. Security requirements for different critical infrastructure providers were analysed and the conducted analysis led to the chosen of cloud based access control security challenges. We done a security requirement analysis for cloud based access control systems, identified challenges for deploying conventional access models (RBAC, MAC, etc.) in cloud computing and presented new evaluation criteria for cloud based access control systems. The requirement analysis for cloud based access control systems facilitated proposing a new cloud based access control model that can fulfil the surveyed security requirements of cloud computing customers. As the proposed cloud access control model has a number of components, we focused on identifying threats to isolation levels enforced by the cloud based access control model in lower levels (CPU caches). Specifically, we looked at cache side-channel attacks. We proposed two new solutions to prevent and detect cache side-channel attacks in cloud computing. Both of the proposed solutions were fully implemented on a server identical to cloud computing servers. Due to the size of the RAM (16 GB), we could not run more than 30 virtual machines on the server at the same time. Furthermore, results obtained from our solutions to prevent and detect cache side-channel attacks are by far better than results gained from other prevention and detection proposed solutions implemented in the same server.

The contributions of this thesis can be respectively categorized in the following:

1. Critical infrastructure providers' security requirements

   This project presents a list of security requirements for different critical infrastructure providers to be moved to cloud computing. This list covers more than four sectors of critical infrastructure areas, which can also be applied to the other areas. The details can be found in section 3.1 in the chapter three: related work.

2. Requirements analysis for cloud based access control systems

   In order to propose a cloud based access control model, we have carried out an in-depth investigation to identify every possible security requirements needed for cloud based access control model. It is demonstrated in section 3.2.

3. A novel guidelines criterion for cloud based access control models

   This project illustrates novel guidelines criteria, which is shown in section 3.3. It can be used either to propose or evaluate access control models for cloud computing.

4. The cloud access control model contributions

   Chapter 4 presents a novel access control model for cloud computing that facilitates the role and task principles to restrict access to cloud computing resources. Major contributions in the cloud access control model are as follows:

   - It ensures a secure cloud that has sharing of physical resources among potential untrusted tenants.
   - It has three different security levels, which can be used according to level of trust.
   - It supports various sensitive levels of information in order to restrict who can read and modify information in the cloud.
   - It copes with different access permission to the same cloud user and giving him/her the ability to use multiple services with regard to time of authentication and login.

5. A new Prime and Probe attack

   It is a new way to launch cache side-channel attack without utilising the link between memory pages and CPU cache sets. It uses a virtual machine's virtual addresses and translates them to physical. These addresses will be checked either accessed by another virtual machine or not in order to get the virtual machine's data. The details can be found in section 5.1.

6. A novel lightweight solution to prevent cache side-channel attacks in cloud computing

   Section 5.2 in chapter 5 illustrates a novel solution to deal with leakage of data from all levels of the CPU caches by preventing cache side-channel attacks. The major contributions in the novel lightweight solution are as follows:

- The solution erases all data related to a Virtual Machine (VM) from all CPU cache levels directly after its Virtual CPU (VCPU) releases the control of the CPU resources and does not erase or modify any data that does not belong to the targeted VM.
- The first solution to deal with a VCPU and target all its overlapping situations.
- It is generic and can be used with any operating system (the Xen hypervisor is compatible with many operating systems such as Windows, Linux (Ubuntu, SUSE, etc.), etc.).
- The solution induces neglected performance overhead.
- All the tenants' operating systems are completely free from any modifications or changing in their applications or libraries.
- It uses Clflush to erase virtual addresses from cache by utilising the Linux Ubuntu operating system.
- The solution can deal with all overlapping access and core migration in the following cases:
  - Number of VMs or domains overlapping access to one VCPU.
  - Number of VMs or domains overlapping access to number of VCPUs.
  - Number of VCPUs overlapping access to a CPU core.
  - A VCPU migrates from a CPU core to another.

7. A novel infrastructure detection solution to detect cache side-channel attacks in cloud computing

   One of our main contributions is a novel solution that can detect cache side-channel attacks in cloud computing without needing for any other software to detect cache misses caused by a virtual machine. It is demonstrated in section 5.3. Furthermore, major contributions in the novel infrastructure detection solution are as follows:

- The solution measures cache misses caused and when they happen by a VM in its execution time without using any attached software or application.
- It measures the time taken by CPU Fetch cycle to detect CPU cache misses.
- The solution has a small performance overhead.
- The hypervisor used in the cloud infrastructure does not require any major modifications.
- All the tenants' operating systems are completely free from any modifications or changing in their applications or libraries.

1.9 Outline of the Chapters

The report is divided into 8 chapters, each covering a specific area of the project work. The following outline sections provide an overview for each of the chapters to guide the reader through the report.

Chapter 1 Introduction: This chapter details the overall aims of the project and what it hopes to achieve. It briefly discusses the background of the project and justification as to why the project was conducted. The final part of this introductory section provides a general overview of each of the chapters in the report.

Chapter 2 Background: This chapter looks in-depth at cloud computing in terms of its features, services and security challenges. Here, the author highlights basic information about access control systems. It also illustrates the conventional access control models and reasons for not being deployed as they are in the cloud. The chapter also provides basic information about access control attacks and a concise review of cache side-channel attacks and previous research work in this domain, and how far they went in solving identified issues.

Chapter 3 Related Works: Here, the author sets out the definition of critical infrastructure and how it is crucial to societies. A novel analysis of security requirements of various critical infrastructure providers is demonstrated in this chapter. Furthermore, this chapter highlights the importance of an access control system to cloud service providers and their tenants. A new cloud based access control criteria are presented in this chapter.

Chapter 4 Access Control Model For Cloud Computing: This chapter presents the proposed solution model and its design methodology. The author also describes the implementation strategies, a security evaluation, a case study, analysis and discussion of characteristics of the proposed solution.

Chapter 5 The Prevention and Detection Solutions to Cache Side-Channel Attacks in Cloud Computing: This chapter presents a new Prime and Probe attack. In this type, attackers do not have to have any information about number of cache lines or memory pages. Furthermore, novel solutions to prevent and detect cache side-channel attacks are illustrated in this chapter. They are a novel lightweight solution to prevent cache side-channel attacks in cloud computing and a novel infrastructure solution to detect cache side-channel attacks in cloud computing. This chapter details the intended goals to be achieved for both solutions and the design of them.

Chapter 6 The Implementation of the Proposed Prevention and Detection Solutions to Cache Side-Channel Attacks: This chapter illustrates full details about the testbed, which is used to implement cache side-channel attacks prevention and detection solutions. It justifies the reasons behind using the Xen hypervisor and gives full explanation about how both solutions are implemented in the Xen hypervisor source code.

Chapter 7 Results and Evaluation of the Prevention and Detection Solutions to Cache Side-Channel Attacks: This chapter sets out the evaluation results, which were gained from a number of conducted experiments. The results were compared with other results from a number of proposed solutions implemented in the same conditions used to test our solutions.

Chapter 8 Conclusion and Future Works: This chapter summarises the entire project and reviews the findings. It also outlines future work that can be done to improve the project.

CHAPTER 2 Background

This chapter provides an overview of the necessary background on cloud computing services and security challenges. It also presents detailed information about access control's definition and models; and why they cannot be deployed in their current state in the cloud. Furthermore, it illustrates a number of security attacks that target access control systems. It provides a concise review of side-channel attacks, specifically cache side-channel attacks. Previous research work in cache side-channel attacks and their specific threat to cloud computing are also illustrated here.

2.1 Cloud Computing

In recent years, we have seen a dramatic growth in IT investments, and a new term has come to the surface which is, cloud computing. The National Institute of Standards and Technology defines the cloud computing as "*a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction*" [1]. Cloud computing is an open standard model, which is Internet-centric and provides various services either software or hardware. It needs minimal management effort from service providers. A significant interest in both industry and academia has been generated to explore and enhance cloud computing.

Cloud computing can enable ubiquitous computing and offer on-demand network access to a shared pool of configurable computing resources. It provides all of its resources as services such as storage, computation and communication. Cloud computing is a unique combination of capabilities and innovation technologies. It needs minimal management effort from service providers and delivers scalable and dynamic infrastructure, global/remote access and usage control and pricing. Furthermore, it changes delivery platform, services consumption and the way users and businesses interact with IT resources. It consolidates the economic utility model with the evolutionary enhancement of many utilised computing approaches and technologies, which include computing infrastructure consisting of networks of computing and storage resources and applications.

Cloud computing has a number of unique features, for example, virtualization and multi-tenant. It has five essential characteristics: on-demand self-service, measured service, rapid elasticity, broad network access and resource pooling. It is aiming at giving capabilities to use powerful computing systems while reducing the cost and increasing efficiency and performance [1]. In addition, it offers users scalable services on-demand, which will be used in cloud computing to provide services to a wide variety of consumers [22]. As illustrated in figure 1, these service models are:

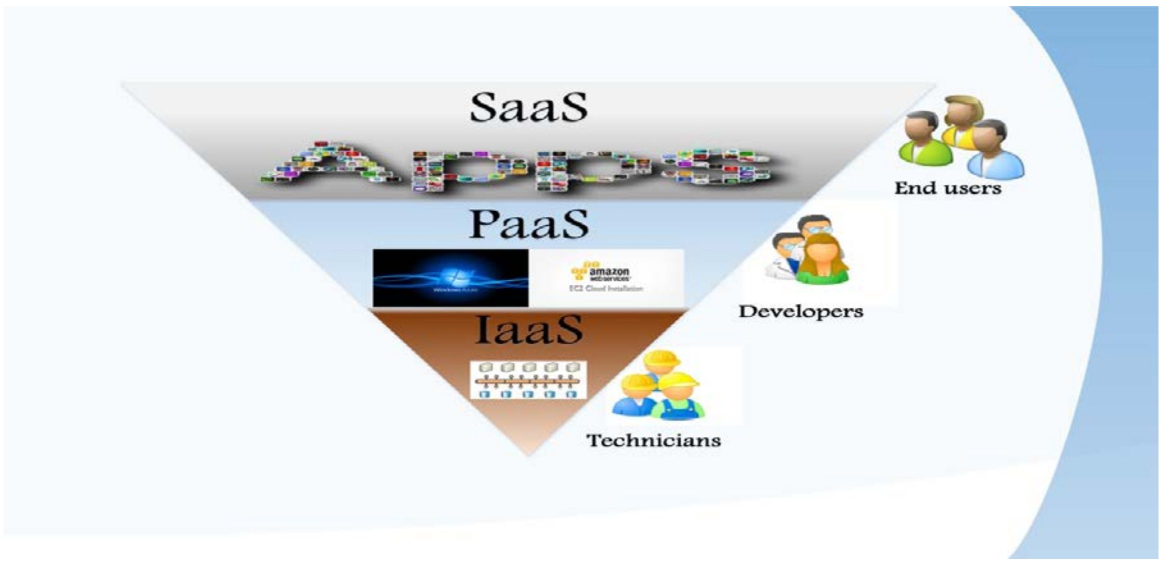


Figure 1: The cloud computing service models

1. Software as a Service (SaaS)

   It is on-demand applications and software service, which hosts applications and their data in the cloud. Software and applications are available to be accessed anywhere at any time by using ordinary web browsers over the Internet.

2. Platform as a Service (PaaS)

   PaaS will facilitate cloud computing customers with the ability to build, enhance and deploy their application without the complexity of configuration, cost of managing and buying the underlying either software or hardware.

3. Infrastructure as a Service (IaaS)

   It offers cloud computing consumers computing infrastructure components such as storage, processing and any other computing resources. It supplies virtualized infrastructures, which are bought or rented as a fully outsourced service.

In cloud computing, there are four architectures have been considered for being ways to address cloud computing customers' demands [23], they are described as follows:

1. Public cloud

   Cloud computing resources are provided by a service provider whom makes them accessible and available to industry groups and general public.

2. Community cloud

   It is another kind of cloud, which is shared by various organizations and offers shared infrastructure for a specific community.

3. Private cloud

   It is supplied and managed by a cloud service provider for a single organization.

4. Mixed cloud

   It is a combination between two or more clouds such as public, private or community clouds. However, these mixed clouds will remain unique entities.

Cloud computing gives a new hope for meeting various requirements of service providers and consumers as well, when they look at what the cloud can offer to them. A report from The Economist Intelligence Unit and IBM finds that among 572 business leaders surveyed, almost three-fourths indicate their companies have piloted, adopted or substantially implemented cloud in their organizations and 90% expect to have done so in three years. Moreover, a number of respondents whose companies have "substantially implemented" cloud is expected to grow from 13% today to 41% in three years [24]. A new report published April 2014 by the Right Scale, finds that among 1000 IT executives surveyed, almost 94% indicate their companies are running Software as a Service (SaaS) or experimenting infrastructure as a Service (IaaS). Furthermore, 87% of surveyed organizations are utilising public cloud services [25].

The unique benefits of cloud computing have provided the basis for many critical infrastructure providers to migrate to the cloud computing paradigm, for example, IBM and Cable & Wireless (C&W) have announced plans to collaborate in the development of a cloud-based smart metering system[26]. This system aims at deploying about 50 million smart meters in the UK by 2020. BT has deployed a new cloud-based supply chain solution to increase the operational efficiency, improve customer service and optimize reverse logistics [27]. In April 2013, the National Grid, the UK's gas and electricity network, has announced plans to replace its own internal datacentres with a CSC-hosted cloud [28]. However, with all of these promising facilities and benefits, there are still a number of technical barriers that may prevent cloud computing from becoming a truly ubiquitous service. Especially where the cloud computing tenants have strict or complex requirements over the security of infrastructures [23].

2.1.1 Cloud Computing Security Concerns

The latest cyber-attacks on high profile firms (Amazon, Google and Sony's PlayStation) and the predictions of more cyber-attacks on cloud infrastructure are threatening to slow the take-off of cloud computing. The numbers of cyber-attacks are now extremely large and their sophistication so great, that many organizations are having trouble determining which new threats and vulnerabilities pose the greatest risk and how resources should be allocated to ensure that the most probable and damaging attacks are dealt with first. These security concerns and attacks could slow the growth of the cloud computing market, which is expected to reach \$3.2 billion in 2013 in Asia alone, while the global market could reach \$55 billion in 2014 [22].

Security is the greatest inhibitor for adoption and the primary concern in cloud computing. Cloud computing inherits some security risks and vulnerability from the Internet, such as malicious code (Viruses, Trojan Horses). In addition, cloud computing suffers from conventional distributed systems security attacks, which could have a huge impact on its services such as back door, Man-in-the Middle attack, and etc. Moreover, cloud computing has brought new concerns such as moving resources and storing data in the cloud with the probability of residing in another country, which has different regulations. A cloud service provider has to ensure its computing resources are fully usable and available at all times [29]. Computing resources could be inaccessible due to many reasons such as natural disaster or denial of service. Protecting data privacy is another aspect in cloud computing security. Cloud computing is a shared environment, which uses sharing infrastructure. Hence, data may face a risk of disclosure or unauthorized access.

Cloud computing services are delivered by a large number of service providers. They use various types of technologies, which cause heterogeneity issues [5]. Extensibility and Shared Responsibilities is another concern as up to now it is not clear, how security duties should be assigned in cloud computing and who is responsible for what [6]. Furthermore, virtualization is one of many ways used in cloud computing to meet their consumer requirements, but it brings its own threats such as data isolation problems and communication between virtual machines [7]. Cloud computing makes cyber-attacks more likely; many of these attacks are among the most potential and commonly encountered in the wider Internet such as Distributed Denial-Of-Service (DOS) attack [8], insecure application programming interface, abuse and nefarious use of cloud computing, malicious insiders [9], etc.

Information in cloud computing is more likely to be shared among different entities, which could have various degrees of sensitivity. Therefore, it would require strong isolation and controlling access mechanisms. In order to draw the whole picture, we have done an in-depth investigation to find and analyse the cloud security challenges and security requirements for different users, e.g. critical infrastructure service providers.

Cloud computing is a shared open environment, which has its own characteristics and features such as on-demand services and mobility. Thus, cloud service providers need a strengthened access control system for controlling access to their resources with the ability to monitor who deals with them. They should have the ability to deal with dynamic and random behaviours of cloud consumers, and heterogeneity and diversity of service.

An access control system is a collection of components and methods that are concerned with giving right activities to legitimate users based upon preconfigured access permissions and privileges outlined in the access security policy [30]. Access control models in cloud computing can be very complex and sophisticated due to dynamic cloud computing resources [11]. Entities in cloud based access control models are likely to reside in various trusted domains and may be located in another country that has different regulations. Thus, they may not trust each other [31]. Furthermore, existing access control models suffer from the lack of flexibility in attribute management. Heterogeneity and diversity of service are another concern in designing an access control model for cloud commuting [12]. Diversity of access control policies and various access control interfaces can cause improper interoperability [32].

2.2 Access Control

Enterprises and organizations need to make sure the intended users can access information with the exact level of access, no less and no more. Dealing with such secure access requirements are not an easy task with the growing demand on IT-Infrastructure. Thus, IT experts, researchers and institutions have searched for access control systems that can fulfil their security requirements. An access control system is a collection of components and methods that determine the correct admission to activities by legitimate users based upon preconfigured access permissions and privileges outlined in the access security policy [30]. The fundamental goal of any access control system is restricting a user to exactly what s/he should be able to do and protect information from unauthorized access. Generally, a vulnerable access control system can lead to revealing of customers' data and give the attackers the ability to infiltrate organizations and their assets with the ability to bring the system down. Moreover, there is a wide variety of methods, models, technologies and administrative capabilities used to propose and design access control systems. Thus, each access control system has its own attributes, methods and functions, which derive from either a policy or a set of policies. According to NIST(National Institute of Standards and Technology) any access control system should have all of following terms or a number of them [33]:

1. Subject

   It is any active entity, which requests access to an object or data within an object such as a user, a process.

2. Object

   It is a passive entity which has or receives data.

3. Action

   It is an active process that is used by a subject to perform an action on an object such as read, write and execute.

4. Capability

   It determines a subject's available actions and is associated with it.

5. Privileges

   They are used to reduce granted space of access to authorized users to a specific number of users.

In the access control system planning, there are three primary abstractions: access control policies, models and mechanisms [33]. Access control policies are high-level requirements, which are used to determine who can access data and under what circumstances. A mechanism is employed to translate these policies. In case any gap is found between a mechanism and a policy, a model has to solve it. Furthermore, there are three crucial components in any access control system, which are Identification, Authentication and Authorization [34]. When a subject supplies information to identify him/herself to an authentication service it is called identification, for example username or account number. Authentication is verifying the identity of a subject to recognize a legitimate user. The authentication mechanism can be based upon what the user has (smart card), what the user is (biometrics) or what the user knows (PIN or password). Authorization is the process of granting or denying an identified user a permission to access and perform certain operations on the system resources.

2.2.1 Why Conventional Access Cannot Be Utilised in Cloud Computing?

Cloud computing is a shared open environment, which has its own characteristics and features such as on-demand services and mobility. Thus, cloud service providers need a strengthened access control system for controlling admission to their resources with the ability to monitor precisely who accesses them. They should have the ability to deal with dynamic and random behaviours of cloud consumers, heterogeneity and diversity of services. In this subsection, a background about conventional access control models and why they cannot be deployed in the cloud are presented. It also illustrates fundamental requirements for cloud based access control models and existing proposed solutions.

Due to differences in requirements for military and commercial security policies, two distinctive kinds of policies had to be developed, these produced two different access control models which are Discretionary Access Control (DAC) and Mandatory Access Control (MAC) [35]. These models have a number of flaws, which led to the proposal of other models such as Role-Based Access Control (RBAC). However, we believe these models cannot work in cloud computing as each one of them was proposed for a specific environment to fulfil consumers' security requirements.

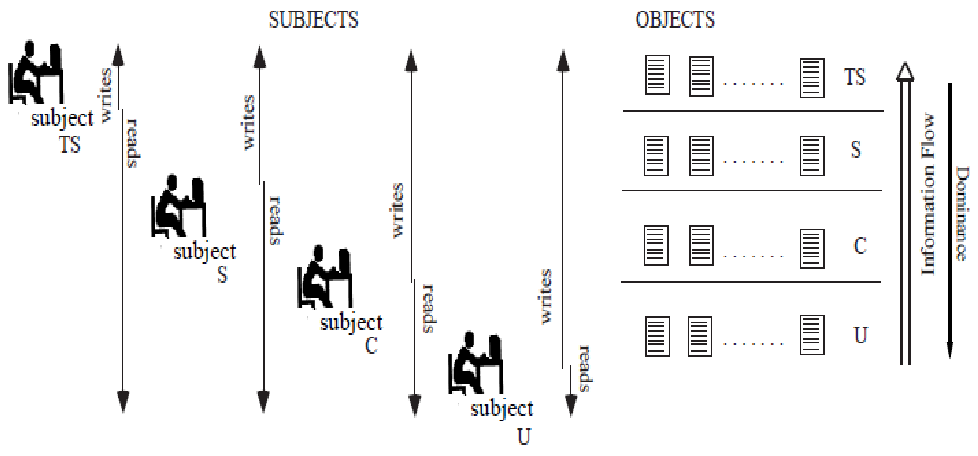


Figure 2: Information flow in the Bell- LaPadula model

2.2.1.1 Mandatory Access Control (MAC) Model

In the Mandatory Access Control (MAC) model, a central authority is in command of giving access decisions to a subject that request access to objects or information in objects. In order to secure access to objects and the information that flows between objects, MAC assigns an access class to each subject and object. An access class is a security level that is used to secure the flow of information between objects and subjects with dominance relationship. Object classifications are security labels that are used to classify objects based upon the sensitivity of information they have. Subject clearances are security levels used to reflect the trustworthiness or rules of subjects. The early formula and most well-known relationships were proposed by [36]. This model is also known as multilevel security and uses only two properties no-read-up and no-write-down as shown in figure 2. The Bell LaPadula model has concentrated on securing and controlling data flow, but protecting the confidentiality in a system is not the only goal in securing information. Hence, [37] used the same principles utilised by Bell LaPadula model to propose a model for protecting the integrity of objects.

Although the mandatory access control model provides protection against information flow and indirect information leakages, it does not guarantee complete secrecy of the information either in the Bell LaPadula model or the Biba model. For example, any unclassified subject can write into top secret objects, and could cause improper modifications to objects and violate their integrity. In fact, this model is very expensive and difficult to deploy and does not support separation of duties, least privilege, and delegation or inheritance principles. Dynamic activation of access rights for certain tasks is not supported. Moreover, it does not support time and location constraints. It needs a precise management system for dealing with system components that reside either inside or outside the model. For instance, processes and libraries are considered as trusted components, but sometimes they need to break MAC principles. Thus, they might need to reside outside of the MAC model. Furthermore, over classifying subjects or objects can happen. The BLP still does not deal with the creation or destruction of subjects or objects [30]. Security's labels are not flexible and are not convenient for task execution. MAC needs a central authority to determine what information should be made accessible and by whom. For example, a manager might want to access information about a staff member, but s/he should not be able to have full access to the member's file, as s/he could access and reveal sensitive information such as bank account details. As cloud computing is going to use current web applications to deliver their services, MAC has to deal with a lack of sophisticated semantics models, which represent and communicate privileges and constraints that are provided via access control policies.

2.2.1.2 Discretionary Access Control (DAC) Model

The Discretionary Access Control (DAC) model, grants the owners of objects the ability to restrict access to their objects, or information in the objects based upon users' identities or a membership in certain groups. The DAC model is generally less secure than the mandatory access control model, so it is used in environments that do not require a high level of protection. However, DAC is the most used model in commercial operating systems such as UNIX and Windows-based platforms [34] because it is more flexible and easier to be utilised than other models. There are two ways to implement a discretionary access control model, this can be achieved via identity based access control or by means of an access control matrix Access Control List (ACL) or capabilities [38].

The DAC depends on allowing owners of objects to control access permissions to objects, yet it has many side-effects when it is utilised in cloud computing. For instance, there is no mechanism or method to facilitate the management of improper rights (e.g. risk awareness), which owners of objects can give to users. Occasionally users are required to use privileges that reveal information about objects to third parties. For instance, a user can only read a file in a company, and then s/he can copy the file contents to another file in order to pass it to another user. The DAC does not have the ability to control information flow or deal with Trojan horses that can inherit access permissions [39]. In addition, a user may pass their rights to another user, and that can violate the integrity and confidentiality of objects. Finally, it is not scalable enough for cloud computing.

2.2.1.3 Role Based Access Control (RBAC) Model

Role-based access control (RBAC) is considered as a natural way to control access to resources in organizations and enterprises [35]. The motivation behind RBAC comes from considering "a subject's responsibility is more important than who the subject is" [40]. In the RBAC model (see figure 3), a subject can have more than one role or be a member of multiple groups. For example, an employee within an organization can be a member in secretaries group and employees group.

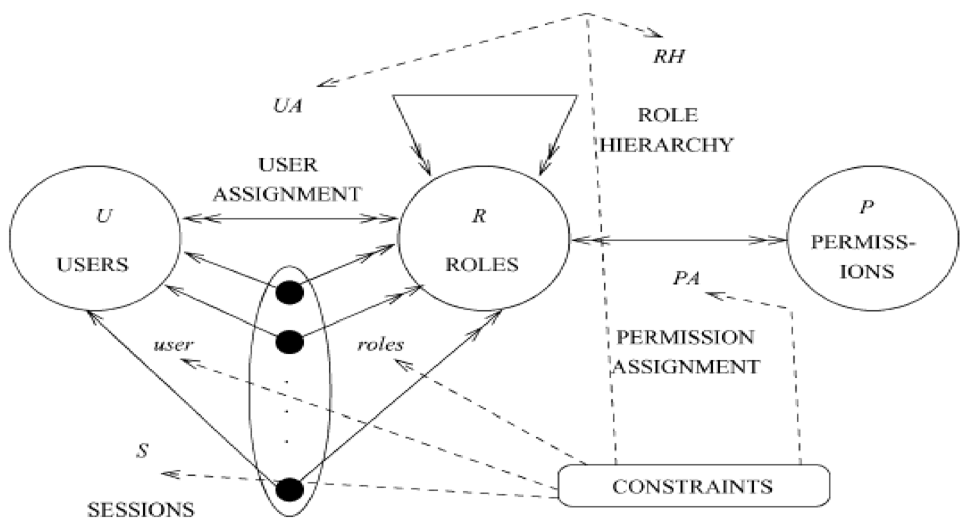


Figure 3: Role-based access control model

Task-Role Based Access Control (T-RBAC) model is another access control approach that has been proposed, which is based upon the RBAC model [41]. However, it assigns permissions to tasks instead of roles. Users in this model are assigned roles, which are assigned tasks that have permissions. It uses the workflow authorization model for synchronizing workflow with authorization flow. The scheme has used tasks to support active access control and roles to support passive access control.

The RBAC model has many advantages compared to, DAC and MAC models, yet it has its own difficulties and problems when it is deployed in the real-world [42]. Firstly, picking the right roles that represent a system is not an easy task and dividing subjects into categories based upon roles might make things worse. Roles in the RBAC model classify subjects in a number of categories; thus each subject has to have a role in order to access the system. Despite that, roles can give a subject more rights than s/he necessarily needs to have, with a possibility of having another role which could lead to the violation of the access security policy. It fails to cope with the following issues:

1. It does not provide any kind of sensitivity to the information. For example some information is more sensitive than others such as the medical history in a patient file.

2. Relationships define according to identities not just roles such as the doctor-patient relationship. For example, a paediatric doctor should not be allowed to access a patient file in the psychology department.

3. It does not support the delegation principle, which is needed in organizations for dealing with absences of their staff. Furthermore, it does not consider the time and location constraints, which are utilised for restricting access to system files and decreasing the probability of information leakage. It also fails to cope with dynamic and random behaviours of users.

4. It does not support active responsibilities of staff as it does not separate tasks form roles. Moreover, dynamic activation of access rights for certain tasks is not supported.

5. The RBAC has to deal with a lack of sophisticated semantic models to represent and communicate privileges. For instance, a doctor in a remote area might not be able to access the system via cloud computing due to lack of syntactic and semantic support. It also has to cope with a semantic gap between the user authentication mechanisms and the authorization mechanisms.

6. In cloud computing, there is a huge demand for testing and verifying access control functions, which are considered as static tests. There are also other dynamic compliance functions that can be used as support functions; for example reporting alerting privileges or conflict of rules and monitoring the system current states [33]. For instance, a doctor who has full access to patients' files in his/her department should not be allowed to move, copy them to another place or even access them from home.

7. Before utilising the RBAC in cloud computing, it has to ensure granting access decisions in a reasonable time and according to system requirements. For example, the response time is crucial in many applications such as a health care system. A consultant away from a hospital needs to access the system in a timely manner, disregarding a number of access requests to the RBAC and distance.

8. Any critical infrastructure service provider who aims to migrate to the cloud, with thousands of users, hundreds of roles, and millions of permissions face a tremendous task that cannot realistically be centralized by a small team of security administrators [43].

9. In a health care system, there is always a sequence of operations which will need to be controlled. For example, in order for a doctor to give a patient the right treatments, s/he needs to examine the patient's physical conditions, look at the patient's medical history

and asks for tests or scans. S/he might ask for help from another doctor or transfer some information to another hospital. Each one of the previous operations needs a different set of permissions. Thus, the RBAC may not be able to ensure access for a sequence of operations in cloud computing.

2.2.1.4 Attribute Based Access Control (ABAC) Model

The Attribute Based Access Control (ABAC) model relies on a set of attributes associated with a requester or a resource to be accessed in order to make access decisions [44]. There are many ways to define or use attributes in this model. An attribute can be a user's work start date, a location of a user, a role of a user or all of them. Attributes may or may not be related to each other [45]. After defining attributes that are used in the system, each attribute is considered as a discrete value, and values of all attributes are compared against set of values by a policy decision point to grant or deny access. These kinds of models are also known as either Policy-Based Access Control (PBAC) or Claims Based Access Control (CBAC). Moreover, a subject does not have to be known in advance to the system, it just needs to authenticate itself to the system then provide its attributes. However, reaching an agreement about what kind of attributes should be used, and how many attributes are taken into account for making access decisions is a complex task in cloud computing [46]. This model has not at present been implemented for well-known operating systems [45]. Finally, proposing a security policy that can work accurately with this kind of access control model is vital, because the security policy is responsible for selecting the important attributes that are utilised to make access decisions.

2.2.1.5 Risk-Based Access Control (RBAC) Model

Risk-Based Access Control was proposed to cope with multinational organizations that face various kinds of policies and regulations [47]. This model tries to use different kinds of risk levels with environmental conditions and utilise the principle of "operational need" in order to make access decisions [42]. In Quantified Risk Adaptive Access Control (QRAAC) [48], risk is calculated as risk =V*P, where V is the information value that reflects the sensitivity level of the resource and P is the probability of unauthorized disclosure, which reflects the trustworthiness of the user. The security policy in this model is dynamic; it is changed according to a variety of risk levels stated on the security policy. However, this model is difficult to be deployed in cloud computing because of the amount of analysis required and number of systems to be merged to compute risk levels. It needs expertise that can deal with the model. Finally, security policies and environmental conditions need to be standardized as they play a crucial role in making access decisions [42].

2.2.1.6 Existing Proposed Solutions

There are a number of access control models were proposed for cloud computing, yet each one of them was targeted a specific vulnerability in cloud computing without addressing the security requirements of cloud computing services. Furthermore, none of them was implemented in a real scenario or environment and tested against any type of access control attacks. They had not considered the components and processes engaged in any attempts to access data or resources in cloud computing. For example, virtual machines that may be used to access data and any processes or programs used to read or transfer data. Moreover, there is always a sequence of operations that need to be controlled in order to access cloud services. Thus, a cloud based access control model needs to be tested in such cases with different scenarios to make sure right permissions are given for every task or action engaged in these operations.

Most of the proposed models were tried to deploy the conventional access control models directly to cloud computing. However, no improvements had been made to these models before they attended to be deployed in cloud computing. Cloud computing service providers and consumer have their concerns and security requirements from cloud based access control models. None of the conventional access control models can be utilised in their current state in the cloud as stated in the second paragraph. The well-known access control models are suffering from semantic problems which affect the interoperability between various cloud service providers. Cloud computing and cloud based access control models have heterogeneity problems, yet there is no sign how these problems can be solved in the proposed models. The following number of proposed access control models for cloud computing utilised the traditional access control models:

1. Wang et al. proposed an adaptive access control algorithm for cloud computing environments based upon the contextual information such as security information and time [13]. In this scheme, authors combined the trust relationship (either between a number of cloud service providers, or a cloud service provider and its consumers), with the role based access control system. The trust level is updated and changed automatically by the trust management system according to evolution done by clouds

after each transaction. In this scheme, authors assume every cloud has a global certificate Authority Authorization Centre (AAC), which is responsible for access control. Combining the access control system with a trust level calculated and modified based upon a user's behaviours, is a good approach. However, scores related to a trust level that is calculated by either users or resources can cause misbehaving problems. Furthermore, the type of mechanism used and how the access can be granted is not evident. The AAC has the potential to be a single-point of attack. In the model, it is not clear where the RBAC model is located and how access to resources is granted (roles, trust scores or both).

2. Tianyi et al. proposed the cloud optimized RBAC model (coRBAC) [32]. It inherits many features from RBAC and distributed RBAC (dRBAC) such as dRBAC's domain. It merges the dRBAC's distributed authentication services together and gives the CA ability to issue certificates. It also allocates domains for enterprises and companies with the capability to manage their roles and users in their own inner network. The coRBAC implements an internal RBAC in each organization, and there is only one manager role called D.manager in each internal RBAC. This approach depends on Certificate Authority (CA) for issuing users' certificates, which might cause efficiency and scalability problems as a new certificate must be issued each time access is required. A third party infrastructure for storing some information such as domain information has been utilised, yet it is not clear how it enhances the level of security, and it might bring other concerns such as trust issues. The network performance can be affected by using the CA as it can be a single point of attack and bottleneck in some situations. Moreover, they have not proposed any mechanism for dealing with heterogeneity caused by bringing various security domains together in the model. In the issued certificate, no information about the security domain is provided.

3. Sun et al. proposed a semantic access control scheme for cloud computing to authenticate users of healthcare systems based upon ontologies [49]. This scheme implements an access control system in semantic web environments and uses ontologies for the RBAC security model. The authors extended the RBAC model by using semantic web technologies. This scheme has not sorted out the dynamic activation problems in the RBAC or diversity of data sensitivity. For a health care system that aims to migrate to the cloud, with thousands of users, hundreds of roles,

and millions of permissions, it is a tremendous task that cannot realistically be centralized by a small team of security administrators. Moreover, in the health care system, there is always a sequence of operations that need to be controlled. Thus, the model needs to investigate how the right permissions can be given for every task or action engaged to complete the preformed operations.

4. Task-Role-Based Access Control scheme is another access control approach which has been proposed for healthcare systems in the cloud computing environment [50]. Permissions in Task-Based Authorization Control (TBAC) are activated or deactivated according to the current task or process state. As there is no separation between roles and tasks, they use different factors such as users, information resources, roles, tasks, workflow, and business rules, to solve the separation problem and determine the access control mechanism. This scheme was implemented in the Amazon Elastic Compute Cloud (Amazon EC2), yet there is no clear sign about how semantic problems can be handled, how information is meaningfully shared among different hospitals, and how the separation problem between roles and tasks is solved. Furthermore, health care systems usually suffer from heterogeneity problems and T-RBAC as well, yet here there is no sign how these problems can be solved. It does not provide any sensitivity levels to information, as it must be considered that some information is more sensitive than others such as medical history.

5. A reference ontology framework using Role-Based Access Control model was proposed by [51]. It aims at providing an appropriate policy with an exact role for every tenant. In addition, different policies are used to grant permissions such as access policy and security policy. Policies might be used as components of a role according to the role's characteristics, such as priority and business values. However, this approach needs good ontology transformation operations algorithms to compare the similarity of different ontology. A new back-end database schema to support O-RBAC is needed for this scheme. Furthermore, it has to be evaluated by testing the role/user ratio according to the position hierarchy. It does not give sensitivity levels to the information as some information is more sensitive than others, and does not support the delegation principle and dynamic activation of access rights for certain tasks. This model has to ensure granting access decisions in a reasonable time and according to system requirements.

6. Mon & Naing proposed a privacy enhancement system on academic-based private cloud system using Eucalyptus open source cloud infrastructure, and they call it Attribute Role-Based Access Control (ARBAC) [52]. They try to guarantee privacy of cloud's users and security of the personal data, by combining two approaches together, which are role-based access control and attribute-based access control model. The authors stated protecting data privacy is the main object in their scheme, but there is no clear explanation or evidence how it is protected. They combine the RBAC and ABAC, but there is no sign or indication how they are combined or what benefits are gained from merging them.

7. The Mandatory Access Control (MAC) model was used by Hu and others to propose a data security access control model for cloud computing[53]. However, this model is very expensive and difficult to deploy and does not support separation of duties, least privilege, and delegation or inheritance principles. Dynamic activation of access rights for certain tasks is not supported. Moreover, it does not support time and location constraints. It needs a precise management system for dealing with system components that reside either inside or outside the model.

8. Zhou and Li proposed a hybrid access control framework for IaaS Clouds [54]. They combined the Role-based Access Control (RBAC) and Type Enforcement (TE) model to enable unified access control and authorization for IaaS clouds. A permission transition model was used to dynamically assign permission to virtual machines. However, combining two traditional access control models may make the generated access control model more complicated and hard to manage. Moreover, the overload induced by proposed model is very big, which is one second for every operation carried by the model. No information about how the model was implemented, how it was tested and why the authors did not performed security evaluation on the proposed model.

The following are a number of proposed cloud based access control models looked specifically at particular problems:

1. A mutual trust based access control (MTBAC) model was proposed by [55]. They aimed to take into consideration both users' behaviour trust and cloud services node's credibility by using mutual trust mechanisms. In the proposed model, user's behaviour is divided into three types in a user's trust model (the control factors, the network level

and data items of trust attributes) and each type of attribute has a certain weight. A user's trust level will be acquired through trust quantization of the user's behavior. A trust model of a cloud service node is based on the ant colony optimization algorithm. However, mutual trust between cloud service consumer and provider needs a unified definition to trust relationship and really requires to be evaluated in real cloud environment to deal with uncertainty of access decisions. Scores related to a trust level that is calculated by either users or resources can cause misbehaving problems unless a trusted third party engaged in the process. Furthermore, the type of mechanism used and how the access can be granted is not clear.

2. Ricardo et al. proposed a dynamic risk-based access control architecture for cloud computing [56]. They used three new components added to an extension of the XACML standard. They are the risk engine, the risk quantification web services and the risk policies. Although, adding the new components might help for calculating the risk on cloud computing, the risk awareness depends on calculating the security risk of access decision in real time, which is not easy. Calculating the risk in a real time requires dealing with considerable challenges, for example, quantifying the trust level, operational need and users’ access history.

Most of the proposed solutions are either using the traditional access control models in their current state without considering the cloud computing characteristics or combining a mechanism(s) with an access control model(s). In our view that is not enough, firstly, the features of cloud computing and its tenants’ security requirements have to be investigated to come up with an in-depth analysis to a cloud based access control model. Secondly, this analysis has to be considered for proposing a cloud based access control model that can confidently fulfil the cloud computing tenants’ security requirements. We have done both of them, a novel security analysis for cloud based access control model and a novel access control model for cloud computing. The novel access control model can provide three different security levels with using security tags and risk engines. Furthermore, it supports access to data and resources according to their sensitivity and importance to cloud computing tenants.

2.2.2 Access Control Attacks

There are various types of access control attacks targeting access control systems where they are placed or used. These attacks can allow unauthorized users accessing privileged resources such as file and database. In the next subsections, we are going to explore different types of access control attacks and techniques used to gain access to data or systems.

2.2.2.1 Port Scanning and Compromise

They are common techniques used to identify possible vulnerabilities on remote devices. Attackers use them in order to either control or gain access to these devices, or stopping the network from working and preventing legitimate users from accessing it. The most well-known attacks in this type are:

1. Back doors

   Trojan horses are the common tools used to create back doors into networks. Back doors are used to gain access to networks by inserting a program or utility that creates an entrance for an attacker. This program can give the attacker ability to log in without providing password, or gain administrative rights.

2. DoS and DDoS

   DoS attack is the most popular attack in the Internet nowadays and preventing legitimate users from accessing their data or resources. It can be used to send thousands of SYN TCP messages which make the target's buffer full, flood the target server with a huge number of UDP packets or ICMP ping and cause choking to the bandwidth.

2.2.2.2 Hijacking

This threat could happen when an attacker hacks into a web site that is hosted in the cloud service provider and then secretly installs his/her software and controls the cloud service provider infrastructure. Methods used in this kind of threat are not new such as Man-in-the-Middle attack, fraud and phishing. However moving the data to the cloud might make the situation more complicated. There are different kinds of attacks in this type such as:

1. Man in the middle attack

   It is also can be called an access attack. It is an attack which involves placing a piece of software between two communicating parties and both parties are unaware of it. For example, (A) wants to communicate with (B), (C) will pretend to be (B) to (A) and vice versa. So, (C) could read and modify any messages from (A) to (B) or from (B) to (A).

2. Session hijacking

   This attack can happen when an attacker hijacks a session between trusted client and network, then the attacker substitutes his/her IP address for the trusted client and the server continues the dialog.

3. TCP hijacking

   It gives an attacker ability to gain access to a client in a network by taking over the connection between the client and the network server. The attacker can insert another machine with the same IP address. This happens quickly, which gives the attacker access to the session and the server continues dialog.

2.2.2.3 Malicious Code

Malicious code can be any kind of software that intends to create security breaches or damage to a system. In our digital world, there is a wide variety of malicious codes such as viruses, worms and Trojan horses. In the rest of this section, we are going to explore two of them affect the access control systems, which are:

1. Trojan horse

   Trojan horses are a well-known malicious code since computers and networks have been used. It is attached to programs and once programs are executed, the malicious code will attack. Malicious code could be used to gain access control, create back door and so on.

2. Malicious mobile code

   It is a lightweight program, which is downloaded from a remote server and executed locally with no or minimal user intervention.

2.2.2.4 Password Attacks

It is a classic attack, which aims at gaining access to a system by trying every possible combination of symbols, to find out the password and log in. The types of this attack are:

1. Dictionary attack

Common words are used in this kind of attack to identify users' passwords. This attack can be automated by employing several tools existing in the Internet.

2. Brute force attack

   This kind of attack tries every possible word that can be used as a password until a successful guess occurs. It might take a long time especially when passwords lengths are more than 6 characters with a combination of different symbols.

3. Hybrid attack

   Combines the dictionary as well as brute force attacks.

4. Replay attack

   It is well known attack, which occurs when an attacker intercepts messages and then tries to send them later in order to impersonate one of the participants. It is used for access or modification attacks.

2.2.2.5 Side-Channel Attacks

They are low level attacks that look at the correlation between underlying hardware phenomena and high-level functionalities of the software. These attacks can be hidden to any access control system. There are various types of side-channel attacks, which are classified according to the hardware medium used to collect the data such as cache side-channel attacks.

2.3 Side-Channel Attacks

Cryptographic systems rely on security mechanisms to authenticate communicating entities and ensure the confidentiality and integrity of data. The security mechanisms have to be implemented according to cryptographic algorithms and to meet the security goals of the security systems. Although the security mechanisms can control and specify what functions can be performed, they cannot specify how their functions are implemented. For example, a security protocol's specification does not usually include whether the encryption algorithms are implemented in custom hardware units or using software running on a general processor. It is also independent of the used memory to store intermediate data during computations, either it is on a separate chip or on the same chip with the computing unit [14]. Moreover, cryptographic algorithms are always implemented in the hardware or the software of physical devices that interact with, and are influenced by their environments. These interactions can be monitored and instigated by attackers. The gap between how security functions are specified and how they are implemented, has led to a new class of attack, which is a side-channel attack.

Side-channel attacks are an implementation level attack on cryptographic systems. It exploits a correlation between high-level functionalities of the software and the underlying hardware phenomena. For instance, it looks at the correlation between the internal state of the computation processing device and the physical measurements taken at different points during the computation. It gathers information about certain operations taking place on computation processing activities i.e. power consumption of a custom hardware unit or electromagnetic radiation.

There are various types of side-channel attacks, which are classified according to the hardware medium they target and exploit, for instance, cache side-channel attacks. Furthermore, attackers are always looking for hardware functions that offer a high-rate of computing interactions, which can facilitate attackers with detailed information about the state of computing operations taking place. For example, CPU cache side-channels are one of the hardware devices most prone to be targeted by adversaries due to their high-rate of interactions and sharing between processes [57]. Moreover, according to Zhou et al. [14], there are three major classifications of side-channel attacks:

1. Classifications depending on the method used in the analysis process.

   This type is classified according to the tools or methods used to analyse the sampled data collected from attacks. It has two different methods to perform analysis:

   - Simple Side-Channel Attack (SSCA)

     This type exploits the relationship between side-channel outputs and executed instructions [14]. A single trace is used in an SSCA analysis to extract the secret key. However, the side-channel information related to the attacked instructions (the signal) has to be larger than the side-channel information of the unrelated instructions (the noise) to deduce the secret key [58].

   - Differential Side-Channel Attack (DSCA)

It exploits the correlation between side-channel outputs and processed data. In this type, many traces are used in the analysis, and then statistical methods are used to deduce the possible secret keys.

2. Classifications depending on the control over the computation process.

   In this type, the side-channel attacks are classified according to the control over a computation process by attackers. It is divided into two major categories:

   - Passive attacks

     Where the attacked system works as there is no attack has occurred and the attacker has not been noticed interfering with the targeted operation.

   - Active attacks

     It has some interfering with the targeted operation, and some influence might be detected.

3. Classifications depending on the way of accessing the module.

   This type relies on the kind of interfaces, which the attackers use to exploit the security system. These interfaces can be a set of logical, physical or electrical interfaces. Anderson et al. [59] has classified these attacks in three different types:

   - Invasive Attacks

     De-packaging is used in an invasive attack to get direct access to the internal components of a cryptographic device or module. For instance, an attacker captures a signal of a microcontroller chip of a cryptoprocessor by placing a micro-probing-needle on a bus line after opening a hole in the passivation layer of the microcontroller chip [59].

   - Semi-invasive Attacks

     In such attacks, an attacker gains access to the device, yet without damaging the passivation layer such as using a laser beam to ionize a device to alter some of its memories in order to change its output [59].

- Non-invasive Attacks

  In this attack, an attacker requires close observation or manipulation of the device's operation and does not need to get direct access to the internal components of cryptographic devices or modules, thus becoming completely undetectable. In this process, the attacker just analyses data that unintentionally leaked, such as timing analysis. Timing analysis correlates an operation performed by a device with the time consumed to execute the operation in order to deduce the value of the secret keys.

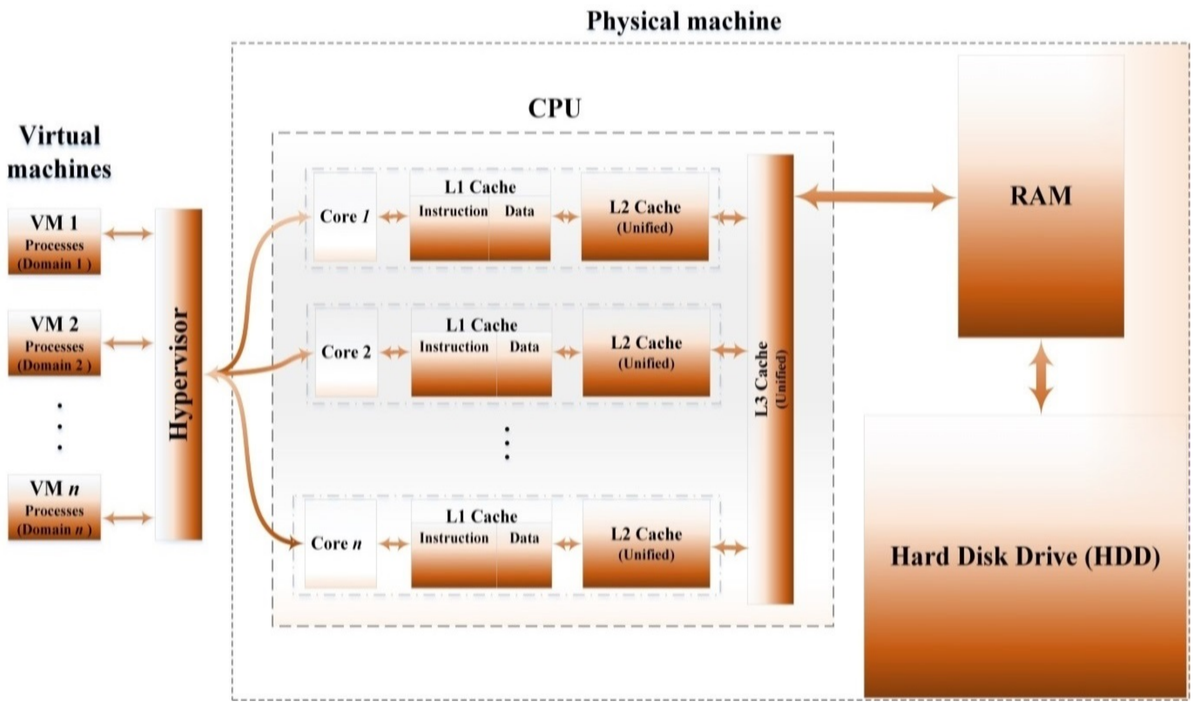


Figure 4: CPU caches and their levels

2.3.1 Cache Side-Channel Attacks

Cache is a small high speed section of memory built in and outside the CPU as illustrated in figure 4. It is usually a Static RAM (SRAM) that any requested data must go through. It contains the most recently accessed data and frequently accessed memory addresses. Furthermore, cache can lead to massive speed increases by keeping frequently accessed data and reducing time taken to evict and fetch data from the main memory. It is utilised to increase the speed of memory access as the time to execute an instruction is by far lower than the time to bring an instruction (or piece of data) into the processor [60]. For example, a 100 MHz processor can execute most instructions in 1 Clock (CLK) or 10 nanosecond (ns); whereas a typical access time for DRAM is 60 ns and the SRAM is 15 ns, which is 3 to 4 times faster than DRAM [61].

As shown in figure 5, a core can have three different levels of cache. Level 1 (L1) is the smallest among them, but it is the fastest. It is usually divided into data and instruction cache. If the requested data is not kept at L1, it will yield a cache miss. Otherwise, it will return a cache hit that means the data are located at the L1. Because L1 is small, therefore, Level 2 (L2) has been introduced. It is much larger than L1, yet slower than it. When the core experiences a cache miss from L1, it will look at the L2 for the wanted data or address of data. If the core gets another cache miss, it will jump to look for the requested data at Level 3 (L3). L3 is the biggest cache in terms of the size, but it is the slowest among them. The L3 is shared between the processor's cores, and others (L1and L2) are shared between processes and threads.

Cache side-channel attacks are a type of Micro architectural Attack (MA), which is a large group of cryptanalysis techniques within side-channel analysis attacks [62]. CPU caches are one of the most targeted hardware devices by adversaries due to the high-rate of interactions between processes [57]. Cache side-channel attacks in cloud computing environments take advantage of running multiple virtual machines simultaneously at the same infrastructure to leak secret information about a running encryption algorithm. Furthermore, full encryption keys of well-known algorithms and schemes such as Data Encryption Standard (DES) [62], Advanced Encryption Standard (AES) [15] and RSA [16], have been broken using spying processes to collect information about cache lines, which have been accessed. The information is analysed and linked to the current virtual machine that occupies the processor.

In the side-channel attacks, attackers are always looking for high-rate hardware functions to explore current running cryptographic operations and the state of the operation in execution. The high-rate hardware functions can communicate information more quickly and deduce the needed data to yield the secret key. Thus, CPU caches are always an interesting target to adversaries due to the following reasons:

1. They are shared among VMs or cores. Therefore, an attacker can easily use clients' co-residence or VM physical co-residency to interfere and exfiltrate sensitive information of victims.

2. They have a higher-rate of computing interactions between processes.

3. They have the most fine-grained and detailed information about the state of computing operations running on a system.

There are three major types of side-channel attacks, which facilitate adversaries with various capabilities to attack CPU caches [63]:

1. Access-driven side-channel attack

   In access-driven attacks, an attacker runs a spy program on the physical machine that hosts it and the victim, in order to get information about cache sets accessed by the victim. There are various types of CPU caches' architectural components that can be targeted. Adversaries can monitor the usage of instruction cache [62], data cache [64], branch-prediction cache [65] or floating-point multiplier [66] to get information about the executing cryptographic operation in order to get the secret key. The most well-known attack in this category is called the Prime and Probe attack. It measures the time needed to read data from memory pages associated with individual cache sets. In this attack, an attacker uses a process to fill cache lines with its own data as presented in figure 5. This step is named as a prime. Then, the process will wait for a prespecified time to let the victim access the cache. Having waited for the predefined time, the process starts the probe stage, which refills the same cache sets with attacker's data and observes the victim's activity on cache sets. If the victim accesses a primed line, data on the line will be evicted and caused a cache miss. This will yield a higher time to read this line than if it is still untouched.

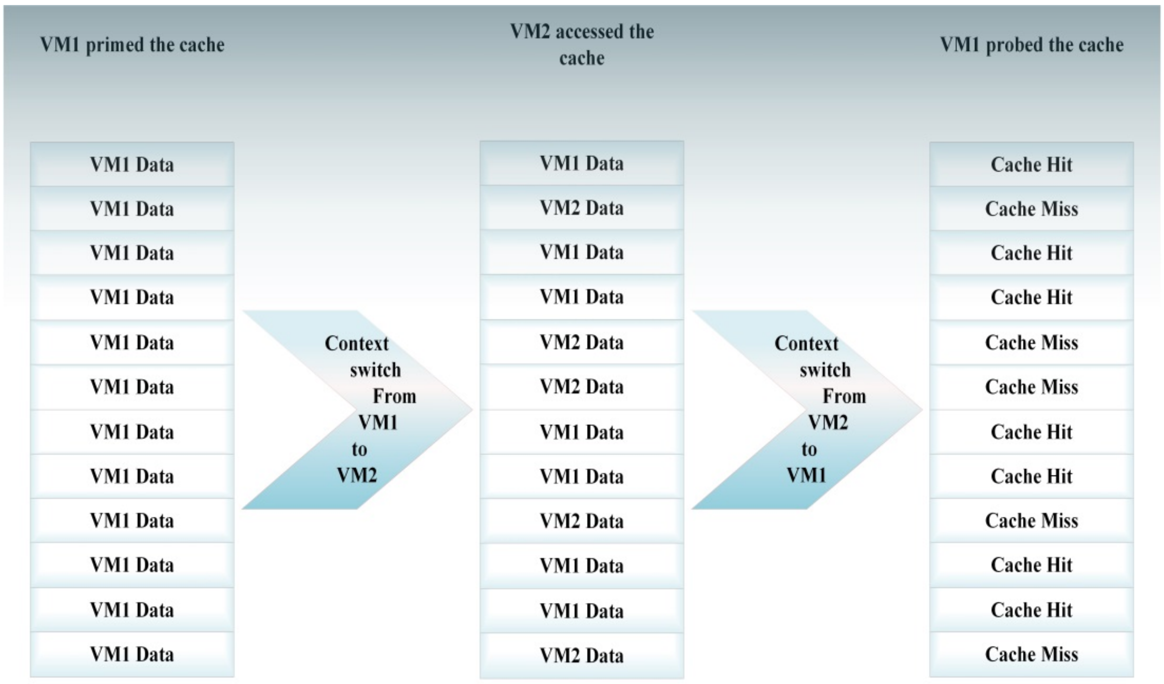


Figure 5: the Prime and Probe attack

There is another type of the Prime and Probe attack, which is Flush+Reload [67]. The spy process shares memory pages with the victim and measures the time to access certain lines. The attack works as follows:

- The spy process flushes the monitored memory lines from the cache hierarchy (L1, L2 and L3 if found) as shown in figure 6.

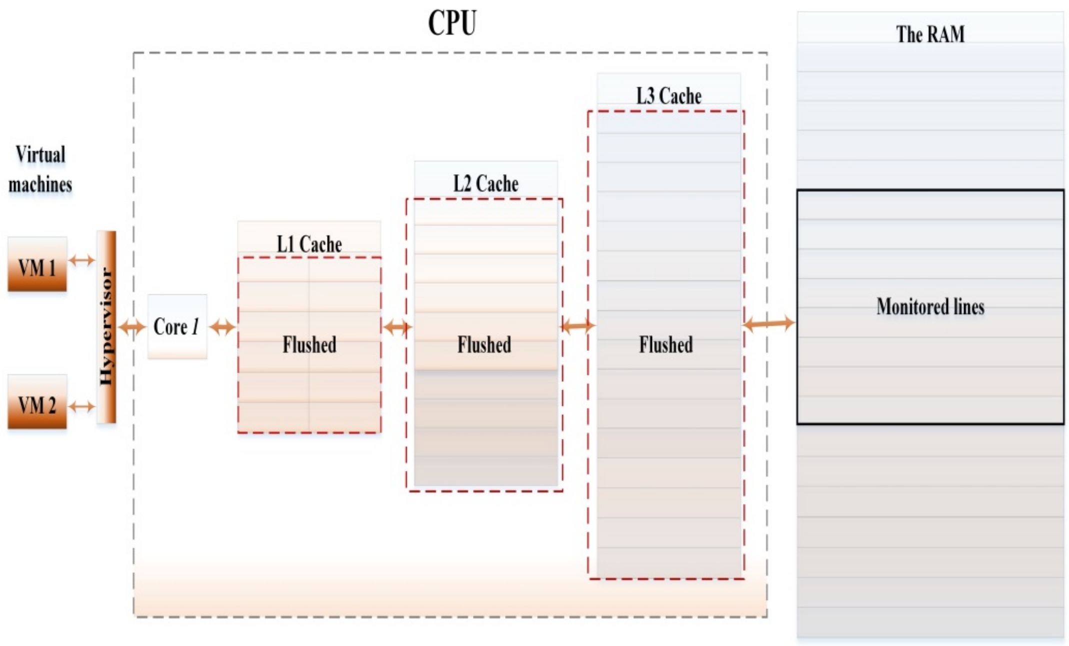


Figure 6: The monitored line in the Flush+Reload attack

- The spy process waits for a predefined time to allow the victim to access the main memory and cache hierarchy.

- The spy process will reload the targeted memory lines and measure the time. If the accessed time is less than a predefined threshold, then it is a hit and the memory lines are accessed by the victim as shown in figure 7. Otherwise, it is a miss.

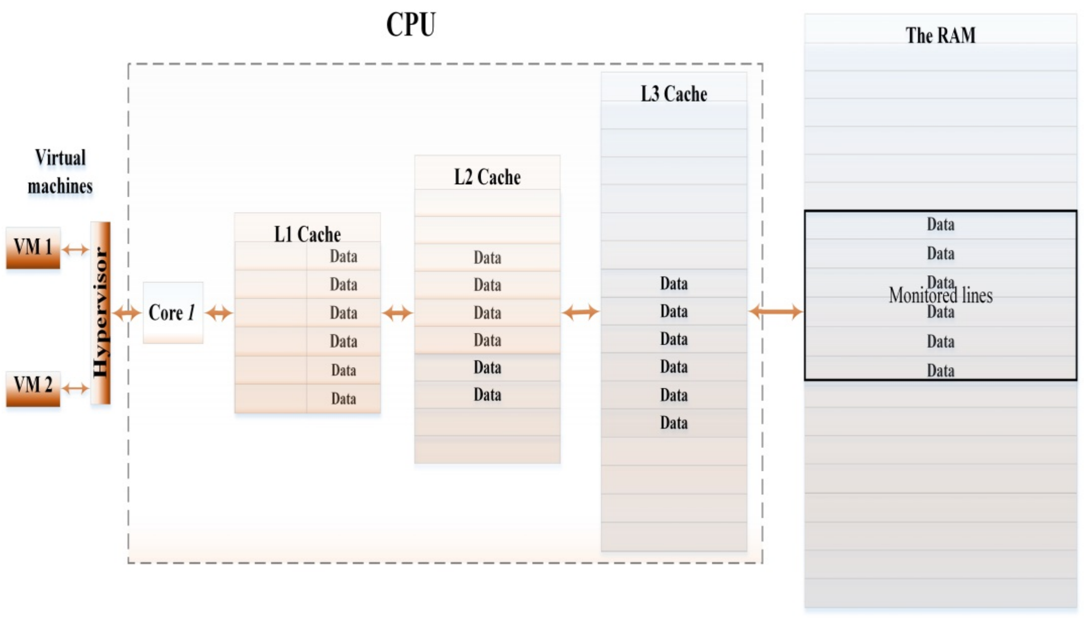


Figure 7: The victim accesses the monitored lines

2. Time-driven side-channel attack

   In this type of attack, an attacker aims to measure the total execution times of cryptographic operations with a fixed key. The whole execution times are influenced by the value of the key. Thus, the attacker will introduce some interference to the victim to learn indirectly whether a certain cache-set is accessed by the victim's process or not. This attack is called Evict and Time [68]. An attacker will execute a round of encryption; evict one selected cache-set by writing its own data on it and measure the time it takes to a round of encryption by the victim. The time to perform the encryption relies on values in the cache when the encryption starts. Hence, if the victim accesses the evicted set, the round of encryption time tends to be higher [69].

3. Trace-driven side-channel attack

The third class is trace-driven, which looks at getting information related to the whole number of cache misses or hits for a targeted process or machine [69]. An attacker can capture the profile of cache activities during a round of encryption in terms of the victim's misses and hits accesses. Moreover, these attacks need to monitor some aspect of CPU caches constantly throughout a round of encryption such as electronic emanations [70]. The ability to monitor CPU caches continuously makes these attacks quite powerful [71].

2.3.2 Gaps in Existing Preventing Researches

Although side-channel attacks in general and cache side-channel attacks in particular have been known for a quite long time, it seems there is a lack of remedies and countermeasures that can be applied in cloud computing. Multi-tenancy and co-residency in cloud computing have gained the researchers' attention to explore and examine the level of damage side-channel attacks can do in cloud computing [72,73,74]. Moreover, it is also proven that side-channel attacks can extract cryptographic private keys from unwary virtual machines [71]. This section will be focusing on a number of proposed solutions to tackle cache side-channel attacks in cloud computing. The proposed mitigation approaches can be classified in two different types of approaches: software-based mitigation techniques or hardware-based solutions.

1. Hardware-based solutions

   A considerable number of hardware solutions have been proposed to tackle and prevent side-channel attacks in general [75,76,77,78,79]. Most of these solutions focus on reducing or eliminating interfering in cache accesses such as cache randomisation [76] and cache partitioning [78]. In the randomisation approach, the cache interferences are randomised by randomising the cache eviction and permutation of the memory-cache mapping [77]. However, the cache partitioning approach is focusing on partitioning the cache into distinctive zones for different processes. Therefore, the cache interfering will be eliminated due to the fact that each process can only access its partition that has reserved cache lines [78]. Although hardware-based defences' techniques seem to be more secure to be implemented as they are more efficient and thwart the root cause of those attacks, they cannot be practically applied until CPU makers implement them into CPUs and that does not seem to have been feasible in the recent times.

2. Software-based mitigation techniques

Software-based mitigation techniques are attack-specific solutions, which can only tackle attacks that they are proposed for. Consequently, these solutions might not have the ability to mitigate new side-channel attacks.

- Assigning predefined cache pages to CPU cores

  This solution relies on assigning one or many prespecified private pages of the CPU cache, particularly the last level of cache (L3) to the CPU cores [17]. So, each core will have a limited amount of memory, which will not be accessed by or shared with other cores. However, it suffers from insufficient uses of CPU cache as operations executed by CPU cores demand different sizes of cache pages. They require various sizes of pages according to operations they are performing. As a consequence, cores will be assigned with more or less than they need of cache pages. Furthermore, when numbers of virtual machines are increased, this approach will suffer from scalability and security issues, as virtual machines can overlap using a CPU core that assigns exclusive pages. Therefore, assigning private pages to a CPU core used by various virtual machines will not prevent cache side-channel attack. Finally, the proposed solution only targets the last level of cache (L3) with extra cost and aims to mitigate active time-driven and trace-driven side-channel attacks. Hence, it cannot prevent other types of side-channel attacks such as access-driven side-channel attacks or deal with other CPU cache levels (L1and L2).

- Flushing the CPU cache

  This solution is targeting the Prime and Probe attack, which is presented in section 3. It flushes the CPU cache to prevent an adversary from gaining any information about timing to read data from memory associated with individual cache sets [18]. In this solution, when two machines overlap and use the same CPU cache, the CPU cache will be flushed immediately after changing from one VM to another. Thus, when a virtual machine primes the CPU cache and waits for another virtual machine to access the CPU cache, the cache will be flushed directly after the second virtual machine takes control of the CPU cache, and that will destroy the probe step. Although this solution can prevent access-driven cache side-channel attacks by preventing interfering between virtual machines, it affects the cache usefulness by flushing the CPU cache when virtual machines overlapping occur. It also introduces overhead particularly when the numbers of virtual machines are increased.

- Inject noise to cache timing

  This approach aims to inject additional noise into the timing that an adversary may observe from the CPU cache [19]. This approach is also targeting the prime and probe attacks. When an attacker periodically primes the CPU cache with its own data, a periodic cache cleansing process will be called to cleanse the CPU cache. So, the attacker cannot observe any timing information about the victim when it launches the probe step. The periodic cache cleansing process primes the CPU cache in random order until all the cache entries have been evicted. However, this approach actually flushes all the CPU cache entries, which will reduce the cache usefulness and introduce unacceptable overhead to the CPU.

2.3.3 Gaps in the Cache Side-Channel Attacks Detecting Solutions

There is a lack of detective solutions that can be applied to cloud computing due to multi-tenancy and co-residency characteristics. HomeAlone solution was proposed by Zhang et al. [20] to allow cloud computing tenants to verify they are physically isolated and exploit the co-resident malicious VM. They gave the cloud's tenants ability to use side-channels analysis as a defensive detection tool. A tenant can utilise the Prime and Probe attack to launch cache side timing channel on CPU cache level 2 (L2). This detection tool only focuses on a special case when the two cloud tenants are physically isolated. It relies on a classifier to distinguish between normal and abnormal CPU cache activities. It enforces each virtual machine (VM) of a tenant to implement a coordinator and re-mapper, which are used to communicate with other VMs and save data in a specific cache sets. Moreover, it might need to silence VMs to avoid noises generated by other VMs. In contrast, our solution does not need any classifier tool, modification to the tenants' VMs operating systems and kernels or to silence any VM. It also does not degrade the performance or introduce overload. It can detect any malicious VM and distinguish it from other VM in all CPU cache levels.

A Two-stage Mode technique for detecting cache side-channel attacks in cloud computing has been proposed by [21]. It uses two stages. Firstly, the host detection step. It uses software (OProfile) to count CPU cache misses and determine whether they are a sequence or not.

Secondly, the gust detection step, which extracts the attack features and computes a number of parameters (standard deviation and utilisation rate index) to decide if it is a cache side-channel attack or not. Although it depends on good software to measure CPU cache misses caused by VM, the OProfile can generate at least 2,000 interrupts per second to the CPU [80]. It also needs to implement an agent in each VM to receive data for the host and the OProfile no order to calculate the other parameters. In addition, it has a quit big false negative rate 40%. In opposition, our proposed solution does not need any attached software or application. It directly measures the CPU cache misses in the Fetch cycle. Furthermore, it has just one stage implemented in the hypervisor kernel. It induces 0% false negative rate and 15% false positive rate.

2.4 Summary

This chapter gave the needed background for cloud computing in terms of definitions, services, types, security challenges and threats. Cloud computing is Internet-centric and provides various services either software or hardware with minimal management effort from service providers. A significant interest in both industry and academia has been generated to explore and enhance cloud computing. However, there are still a number of concerns regarding the security level provided by cloud computing, particularly, securing access to resource and data on the cloud. This chapter also shed light on access control models and reasons behind not being deployed directly in the cloud. The conventional access control models were proposed for specific environments to meet predefined security requirements. Moreover, the conventional access control models and proposed cloud based access control models have not been tested against leakage of data in lower levels caused by cache side-channel attacks. Thus, this chapter covered cache side-channel attacks and gaps in the current detection and prevention solutions.

The conventional access control and proposed cloud based access control models cannot be used in the cloud without addressing its security requirements. This motivates us to perform an in-depth investigation to identify security requirements of cloud computing tenants. These security requirements need to be satisfied in any cloud based access control model. We will see an intensive investigation into four different critical infrastructure providers in order to identify their security requirements in the next chapter. Next chapter will also present new criteria that can be used for proposing a cloud based access control model.

CHAPTER 3 Related Works

Many enterprises and organizations are considering moving to the cloud as one of their future aims, if and only if their security requirements and the needed level of data accesses control can be met. This chapter provides a brief introduction to what the term critical infrastructure means and its importance to our daily life. It details a new intensive investigation into four different critical infrastructure providers in order to identify their security requirements. Furthermore, the importance of an access control system to cloud computing providers is highlighted in this chapter. It describes a novel security requirements analysis for proposing a cloud based access control model. This chapter also presents a novel cloud based access control criteria that can be used for evaluating cloud based access control models.

3.1 Secure Cloud Services for Critical Infrastructure Providers

The Critical Infrastructure (CI) is an essential asset for the maintenance of vital societal services such as power distribution networks and financial systems [3]. The importance of critical infrastructures is hard to overestimate, and even relatively minor failures can impact on large numbers of people. In May 1990 for example, the AT&T PSTN network suffered a fault causing nationwide problems, but which was the result of a human error at just a single switch. At the time the expected failure rate by AT&T for switches was not more than 2 hours in 40 years, a fact that highlights the recognized importance of keeping such systems running. Failures can result in serious consequences for the functioning of society, and this has also been recognized by attackers who have targeted these national infrastructures as a way of disrupting large numbers of people relatively easily. While the AT&T example was the result of accidental failure and human error, it indicates the potential for even a limited attack to have much wider consequences.

As the benefits of cloud computing are hard to ignore, many critical infrastructure providers are aiming to utilise the unique benefits of cloud computing and migrate to the cloud computing paradigm. For example, the National Grid, the UK's gas and electricity network, has announced plans to replace its own internal datacentres with a CSC-hosted cloud [28]. Moreover**,** critical infrastructure providers would require scalable platforms for their large amount of data and computation, multi-tenant billing and virtualization with very strong isolation, Service Level Agreement (SLA) definitions and automatic enforcement mechanisms, end-to-end performance and security mechanisms. For example, NASA and Amazon spent seven months negotiating a cloud service contract because of wrangling over NASA's rights to hardware inspection [81]. Thus, Amazon introduced a new cloud service with physically isolated and tenant-specific hardware to meet NASA's security requirements [20].

However, these requirements might not be met by the cloud computing service providers as they suffer from some challenges and threats. Cloud computing security challenges and the security requirements of critical infrastructure providers may hinder migration to the cloud. Furthermore, moving to the cloud without addressing all of the previously mentioned cloud security challenges is not going to happen soon. In this subsection we are to investigate security requirements for different critical infrastructure providers such as smart grids, telecommunication, transportation and finance.

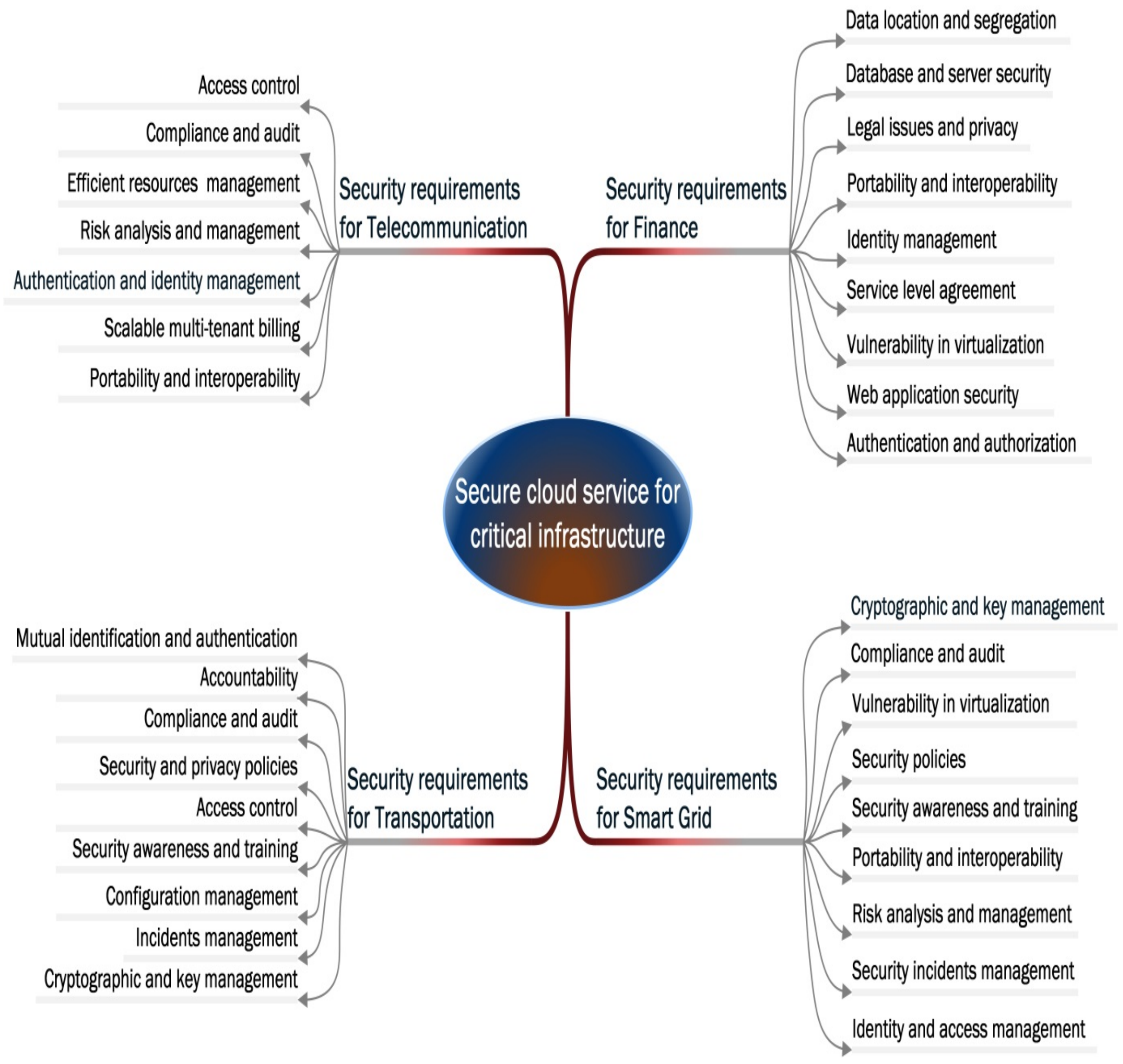

Secure cloud service for critical infrastructure
Security requirements for Telecommunication
Access control
Compliance and audit
Efficient resources management
Risk analysis and management
Authentication and identity management
Scalable multi-tenant billing
Portability and interoperability
Security requirements for Finance
Data location and segregation
Database and server security
Legal issues and privacy
Portability and interoperability
Identity management
Service level agreement
Vulnerability in virtualization
Web application security
Authentication and authorization
Security requirements for Transportation
Mutual identification and authentication
Accountability
Compliance and audit
Security and privacy policies
Access control
Security awareness and training
Configuration management
Incidents management
Cryptographic and key management
Security requirements for Smart Grid
Cryptographic and key management
Compliance and audit
Vulnerability in virtualization
Security policies
Security awareness and training
Portability and interoperability
Risk analysis and management
Security incidents management
Identity and access management

Figure 8: Security requirements for a number of CI providers

3.1.1 Requirement Analysis

Critical infrastructure providers operate using various kinds of infrastructures and may have different security requirements in their unique environments. A successful migration of various critical infrastructure providers to the cloud would need to meet all of their requirements. We have investigated and analysed the security requirements of various critical infrastructure services (see figure 8) to find the common security requirements such as data security, compliance and audit, cryptography and access control. An access control system has been found as one of the core requirements.

In cloud computing, information comes from multiple sources, which need to be secured and controlled accurately. Data should be available only to authorized users, secured from attempting to alter it and available at any time. Privacy of consumers should be ensured at any stage either when data is collected and processed or when it is transferred. So, assurance of 100% availability, integrity and confidentiality is crucial for clouds [82]. Furthermore, the situation in cloud computing might be different from other IT fields as their data can be revealed for reasons such as court orders. So, cloud service providers have to state that in their terms and policies. Privacy issues have to be considered here as well, as data may face different kind of regulations, and any security or privacy policy should illustrate that.

Moreover, sensitive information about consumers could be revealed unintentionally by aggregating data from multiple sources and moving the aggregated data from one place to another can lead to violation of the privacy of the data [83]. Clouds' consumers have to know in advance where their data will reside and how will be segregated in order to avoid data leakage problems. In addition, lack of visibility about the way data is stored and secured, lead to a number of concerns which have to be considered when moving to the cloud. Data centres have to deal with a huge amount of data that is collected from everywhere in the cloud. Data centres are not stand alone; they have to be connected to other data centres. So, security and latency should be managed in a proper way [82].

Compliance, security-breach audit and forensics are used to ensure no one violates or attacks the security within the system [82]. In addition, cloud computing service providers have to apply the right operating models and services to meet compliance and security regulations.

Virtualization brings well known benefits to the cloud, yet it has a number of security concerns such as performance concerns and Hypervisor security [5]. Supporting robust isolation and scalable multi-tenant billing are crucial requirements of any attempt to migrate a system to the cloud. However, in cloud computing there might be multiple networks running in the same infrastructure. So, strong isolation is another requirement to guarantee there is no security or performance interference between cloud tenants. Metering and charging for virtual resources consumption are needed in cloud computing [84].

There are a number of issues which should be considered such as customisation of applications and services, dealing with latency, eliminating any technical barriers and sorting out the complexity of integrating cloud services with existing legacy environments. Highly configurable, secure, virtual machines that provide granular control and allow easy customization are required as well [5]. Moreover, as cloud computing is an environment that has shared platforms, shared storage and shared network, it has to ensure its components work together to achieve the intended mission regardless of providers, storage, OS, etc. [82].

Web applications which are used in the Internet have their own vulnerabilities that have not been solved yet, and these applications are being used again in the cloud to deliver services without a clear idea how their weakness will be sorted out and their impact on cloud users. Additionally, other challenges might be obstacles to moving quickly to the cloud such as meeting security requirements of enterprises, performance, scaling operations, cost-effectiveness, dynamic and size of communication environment, increased size and complexity of operations, changing technology and complexity of services and heterogeneity [84].

Risk analysis and management consist of business risk analysis, a technical risk analysis and infrastructure risk analysis [85]. It is used to deal with dynamic and random behaviours of consumers and mitigates risks involved when consumers are utilising the cloud. A security incident is one of the major questions for any organizations that want to move to the cloud to establish what has to be done if the cloud faces any security incidents and steps to be followed to mitigate that incident. Security's incidents management has to be stated in any agreement between consumers and the cloud [86].

Security and privacy issues, latency, audit and monitoring, reliability, network connectivity and third parties have to be negotiated and addressed in SLA. Cloud computing consumers require SLA definitions and automatic enforcement mechanisms that guarantee sustained and verifiable end-to-end performance. The SLA must state how isolation, bandwidth on-demand and quality of service will be ensured as well [87].

Encryption is often used to secure data in untrusted storage environments such as cloud computing. However, it can be time and cost consuming if it is not handled in a proper way, and it could cause additional storage and bandwidth usage. Key management is another complicated problem, which needs more attention [82].

Consumers are not adequately informed about what can be gained by moving to the cloud computing, and the risk associated with that moving. Consumers should be engaged in the moving process, and in any further action as they have always been considered "the weakest link" [82].

Moving any organization to the cloud needs thinking critically about using multiple sources of identity with different attributes and the ability to identify all the entities involved in a transaction [82]. An access control system is a key element in the four surveyed critical infrastructure providers. Access control mechanisms have to be sufficient and may allow consumers to define access policies to their data and utilities. Furthermore, consumers should be allowed to specify and update access polices on data they own. User credentials should be known in advance whether are stored in either organizations' servers or the cloud, in order to avoid disclosure problems. Last but not least strong mutual identification and authentication between users and network are still an open research area either for cloud computing or for any system wanting to migrate to the cloud [5]. Moreover, there is a huge demand for having proper polices which can organize relations between consumers, utilities and third parties, but using security and privacy policies should not introduce unacceptable latencies.

3.2 Access Control in Cloud Computing

In cloud computing, different entities are likely to share the same resources and information, which could have various degrees of sensitivity. Therefore, it would require robust isolation and controlling access mechanisms. In order to draw the whole picture, we have done an in-depth investigation into cloud security and identified different security requirements for different cloud users (e.g. critical infrastructure service providers and small businesses). We have found access control is one of the common and fundamental requirements for all types of cloud users. However, conventional access control models cannot be applied in the cloud environment due to the following reasons:

1. Cloud computing can be very complex and sophisticated due to the dynamic nature of the cloud's resources [11].

2. Entities which are cloud based are likely to reside in varied trusted domains and may be located in different countries that have various regulations. Thus, they may not trust each other [31].

3. Conventional access control models in cloud computing would suffer from the lack of flexibility in attribute management as defining attributes that should be taking into account for granting access and their weights is a complex tasks [46].

4. Heterogeneity and variety of services [12].

5. Diversity of access control policies and various access control interfaces can cause improper interoperability [32].

6. Dealing with a large number of users, different classification, high dynamic performance, mobility features and changes in high frequency [13].

7. Different access permissions to a same cloud user, and giving him/her ability to use multiple services with regard to authentication and login time [16,88].

8. Sharing of resources among potential untrusted tenants, multi-tenancy and virtualization, mechanisms to support transfer of customers' credentials across layers to access services and resources are crucial aspects in any access control model going to be deployed in cloud computing [88].

There may be possibilities to extend existing access control models and use them in the cloud environment. However, this could be a potential risk and may not solve the problem as conventional access models may focus on a specific problem in a specific platform or environment and miss the remaining interconnected issues. This could happen due to non-existence of a complete list of access control requirements for cloud computing. In other words, the success on any access control solution for cloud computing will depend on analysing and accurately identifying a complete list of requirements.

We have performed a detail investigation and identified access control requirements for cloud computing. To the best of the author's knowledge, fundamental requirements of cloud based access control models have not yet been adequately investigated. We believe the proposed model can fulfil access control requirements for diverse cloud based users who are sharing resources among potential untrusted tenants. In addition, the proposed model has three different levels of security, which can be used according to the level of trust. It supports various sensitive levels of information in order to restrict who can read and modify information in the cloud. The proposed model has the flexibility to cope with different access permissions to the same cloud user and give him/her the ability to use multiple services with regard to time of authentication and login.

3.2.1 A New Evaluation Criteria to Cloud Based Access Control models

We have seen a variety of suggested properties and factors affecting proposing cloud based access control model. They reflect different purposes and backgrounds. Moreover, to the best of our knowledge, there is no complete work, which has been done to outline factors or properties that might have effect on proposing or evaluating cloud based access control models. However, in our view, any proposed access control model for cloud computing needs to address the dynamic and flexible constructions of clouds, big numbers of dynamic users and large amounts of resources.

In order to develop an appropriate access control model for cloud computing environments, we need to investigate the fundamental security requirements for access control models in the cloud. These requirements can help to meet the cloud computing access control necessities and evaluate any proposed system. We provide cloud based access control developers with a novel guidelines list of factors which should be taken into account for proposing a cloud based access control model. We have classified these factors into three classes, which are principles that should be supported, access management issues and how an access control can be measured as illustrated in figure 9.

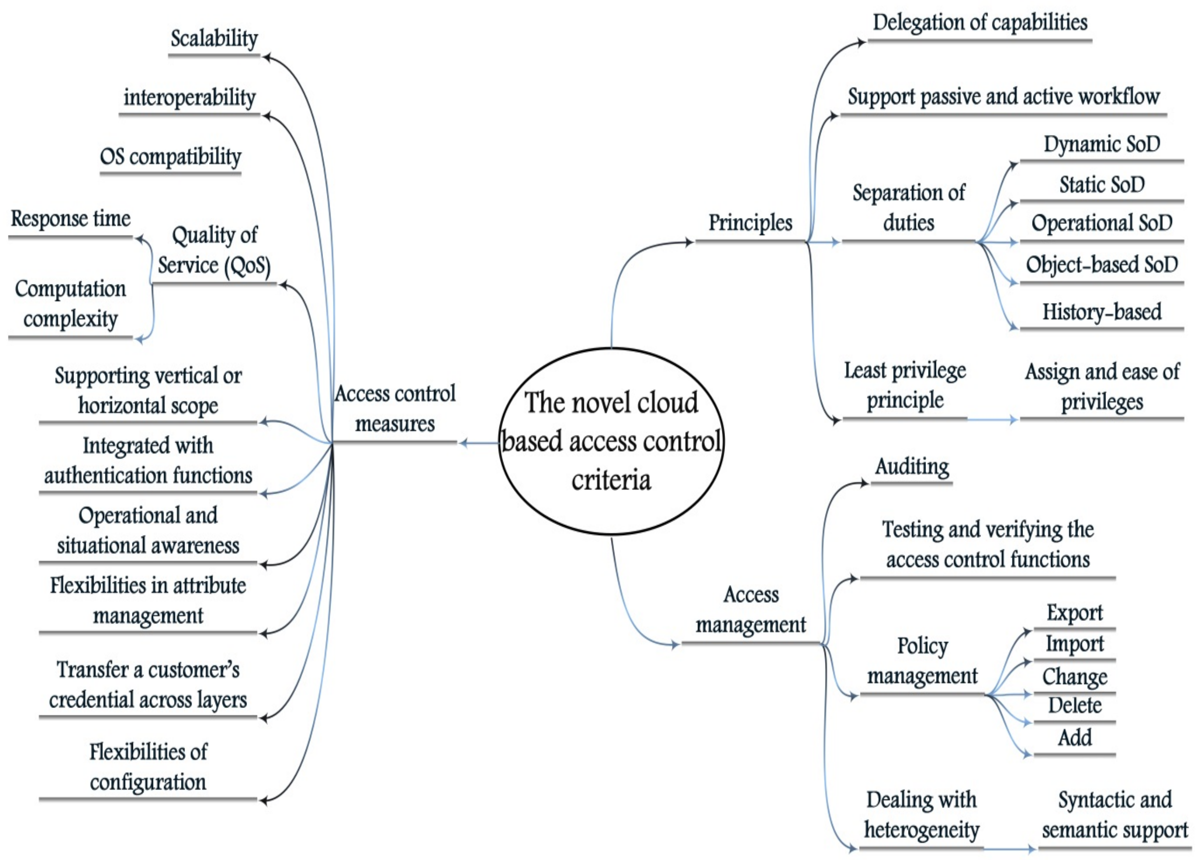

The novel cloud based access control criteria
Access control measures
Scalability
interoperability
OS compatibility
Quality of Service (QoS)
Response time
Computation complexity
Supporting vertical or horizontal scope
Integrated with authentication functions
Operational and situational awareness
Flexibilities in attribute management
Transfer a customer's credential across layers
Flexibilities of configuration
Principles
Delegation of capabilities
Support passive and active workflow
Separation of duties
Dynamic SoD
Static SoD
Operational SoD
Object-based SoD
History-based
Least privilege principle
Assign and ease of privileges
Access management
Auditing
Testing and verifying the access control functions
Policy management
Export
Import
Change
Delete
Add
Dealing with heterogeneity
Syntactic and semantic support

Figure 9: Factors affecting design of an access control system for cloud computing

3.2.1.1 Principles

1. Least privilege principle

   It is the philosophy of granting a subject only the permissions needed to accomplish his/her tasks, even when the subject has more permissions than that necessary for doing tasks [33]. In cloud computing, any proposed access control model might need to support this principle, as it can help prevent misuse of permissions, coping with malicious security breach and limiting damage that can result from system error or malicious events. For example, a user can have more than one role with multi permissions, but s/he should be allowed only to use the necessary privileges to perform his/her job. Moreover, an access control system in cloud computing has to specify how the least privilege principle is ensured either via constraints or access control rules and other specifications [33]. However, enforcing the least privilege principle should not affect the flexibility or add any complexity to the administration of the system.

2. Assignment and ease of privilege

   Assignment and ease of privileges are a critical aspect in any access control system, particularly one that is targeting cloud computing. Whenever fewer steps are required to assign or ease privileges, fewer mistakes can be made due to either human or system errors. The steps required for adding, removing, and changing privileges or capabilities to a subject in an access control system are crucial to the usability of that system. [89].

3. Delegation of capabilities

   In order to make an access control system flexible and have dynamic resource management in cloud computing environment where users collaborate to fulfill their general tasks, it is important to support delegation of permissions and roles [90].

4. Separation of duties

   Separation of Duties (SoD) is a principle that is supported by the least privilege principle, as it aims at partitioning tasks and permissions associated to roles in order to prevent granting too much authority to one user. It also refers to preventing the conflict of roles and interests. For example, the person preparing a paycheck in a bank should not be the one who authorizes it, as assigning the two roles to the same person can raise the chance of stealing from the bank. The SoD feature can be measured by counting the number of different types of SoD a system can support [33]. There are various types of separation of duties:

- Static SoD

  This kind is the simplest version of SoD, which is used in RBAC policies. It prevents any user from being a member of any exclusive roles at the same time, e.g., bank auditor and teller [89]. Thus, the bank auditor is not allowed at any time to be the teller and vice versa. It is also called strong exclusion, which gives two roles strong exclusivity, if no one is allowed to perform both of them at the same time [91]. It also aims at preventing assigning incompatible roles to users [92]. Although it is a simple way for enforcing SoD, it is not practical and does not support the actual functioning of human organizations as users might have legitimate reasons to use or access two roles at the same time [91].

- Dynamic SoD

  It is also named as weak exclusion. Here, a user might be a member of any exclusive roles at the same time such as a user can be authorized for the bank auditor and the teller roles, yet s/he cannot activate both of them at the same time such as used in Chinese wall policy. It uses more policies than the static SoD to control the activation and use of roles [93].

- Other types

  In the Object-based SoD, a subject can be a member of any two exclusive roles and might activate them at the same time as well. However, s/he must not be able to apply the two exclusive roles upon the same object [92]. While in Operational SoD, a user might be assigned some exclusive roles, but he/she is not given permissions to execute every step of a workflow. Hence, a workflow in this type may be split out into a separate sub-workflow within the main process [92]. The last type is History-based SoD. A user in this kind can have a number of exclusive roles with the complete set of permissions to cover working on an entire workflow, yet the user must not be capable of performing all the workflow steps affecting the same object(s) [91].

SoD is a very important principle to access control systems in cloud computing, as it helps prevent inside misuse by separating available roles for a user. It also prevents unintended access to objects and information in objects, and helps to avoid conflict of interest [33].

5. Binding of Duties (BoD), aims at binding tasks associated to roles or participant. There are two types of BoD, either role-level or participant-level. In the role-level, the assigned tasks have to be executed by the same role. However, in the participant-role, the same participant has to perform the specified tasks [94].

6. Support passive and active workflows

   Passive and active workflows can be a fundamental unit of business work or business activity, which are one of the basic components in cloud computing [41]. Roles are a passive workflow while tasks are active workflow.

3.2.1.2 Access Management Issues

1. Auditing

   Audit is an important aspect for securing cloud computing and access controls systems used in it. In access control systems, audit has to monitor a system's current state, record any fail to take a decision either granting or denying and report any attempt to violate the access policy or alter privileges. Moreover, it has to track and keep records about granted capabilities to subjects and any change applied to objects such as renaming, copying and erasing [33].

2. Policy management (add, delete, change, import, export)

   Polices for access control systems in cloud computing have to be flexible and dynamic to deal with unexpected and changeable behaviours. For example, they should have the capability to combine various sub-policies with different rules and prevent or cope with conflicts between policies. The ability of resolving policies' conflicts is necessary in access control policy management. Conflicts can be between different policies used in a system or rules used in a policy. In some cases, enterprises might have branches in different regions around the world, and each branch may apply a privacy policy and regulations of its region. Thus, any lack of policy management in an access control system might violate the data privacy.

3. Dealing with heterogeneity

   Services in cloud computing are delivered by a huge number of mixing technologies and mechanisms, which can cause heterogeneity threats [95]. Hence, heterogeneity in cloud computing can come as a result of differences at various levels either software or hardware levels. Heterogeneity can also happen in access control systems used in cloud computing as they deal with different types of mechanisms, domains and policies.

4. Syntactic and semantic support

   Current access control languages are designed with specific applications or architecture. So, they are not universally applicable for all access control models or mechanisms. Moreover, logic operators and Boolean logic give access control policy authors adequate ways to cope with complex semantics for policy rules. Thus, not all access control languages support logical expression of rules and are capable of programming logic for rule specification [33].

5. Testing and verifying the access control functions

   They are crucial features in cloud based access control models, as having capabilities to test and verify system functions or new updates can promote the level of security needed in such environments. Thus, the testing and verifying feature in access control systems can provide the following benefits:

   - It gives the access control system ability to handle any future changes in the access control policy such as updates, or new events happening.
   - It can help predict the consequence of activation policies and analyse the impact when the policy is modified or combined with other policies.
   - It gives the ability to verify policy deployment and activation compliance.

- It verifies the access control rules against the intended access control properties from the policy.
- It can check impacts of combining access control policies or rules to ensure there are no leaked privileges because of the syntactic or semantic errors.

Furthermore, all the pervious advantages of using the testing and verifying features are static. There are other dynamic compliance functions that can be used as support functions such as reporting alteration of privileges or monitoring the system current states [33].

3.2.1.3 Access Control Measures

1. Flexibilities of configuration

   The flexibilities of configuration into an access control system can help dealing with dynamic environments such as cloud computing. It is aiming at providing efficient administration and flexible configuration for access control systems [89].

2. Operational and situational awareness

   Operational and situational awareness take into account a number of factors that can affect an access control system such as processors, memories, OS or endpoint system components. All the previous factors can have a huge impact on the performance of any access control system as they might affect access decisions.

3. Quality of service

   - Computation complexity

     Computational complexity for enforceability validation of access control rules is still a hard task for any access control system that is capable of implementing any access control policy [96]. The computational complexity can affect the efficiency and quality of service as it might delay the decision making process and cause unacceptable retardation to other parts within the system.

4. Response time

Access control systems in cloud computing are assumed to have a significant number of consumers for authenticating them and dealing with their requests. Thus, they have to grant access decisions in a reasonable time and according to enterprises' requirements. Furthermore, using the time response can partly help assess the complexity of the decision-making algorithm and vice versa [33]. A number of subjects and computational complexities can be also used to calculate and evaluate the response time.

5. Integrating authorization with authentication functions

   In the current access control systems, there is a gap between the user authentication mechanisms implemented by the web application, and the authorization mechanisms implemented by the lower layers, such as an SQL database [97]. Hence, cloud access control systems have to be integrated with or support identification and authentication mechanisms. In addition, they have to perform internal checks to prevent unauthorized operation and control data flow in lower layers. Attributes of subjects and objects might be associated with the identification of users and objects [33].

6. OS compatibility

   Cloud computing relies on virtualization to deliver services to their consumers. Hence, it is important that cloud computing employs an access control system that is compatible with the well-known operating systems and capable of working with different operating systems [33].

7. Interoperability

   In cloud computing, depending upon consumers' requirements, different service providers often collaborate by contributing their resources and consumers. However, diversity of access control policies and interfaces can cause improper interoperability, which hinders any integration or movement from one service provider to another [98].

8. Supporting vertical or horizontal scope

   Supporting vertical or horizontal scope is another criterion considered in evaluating access control systems according to NIST [33]. The vertical scope means controlling applications, databases and operating systems while the horizontal scope aims at controlling single host, distributed network or virtual communities. The vertical and horizontal scopes in access control systems are very important to cloud computing as they consist of platforms, applications, distributed network and virtual communities.

9. Scalability

   Proposed access control models for the cloud have to be scalable in terms of numbers of users, policy evaluation and enforcement points. In addition, scalability should also consider operational, maintenance and management costs. they should not increase when the number of access system components (users, applications) increases [99].

10. Flexibility in attribute management

    Reaching an agreement about what kind of attributes should be used, and how many attributes taken into account for making access decision is a complex task [46].

11. Transfer a customer's credentials across layers

    A cloud user may utilise services from multiple clouds, which employ various types of access control policies. These policies have to support mechanisms to transfer customers' credentials across layers to access several clouds' services and resources. This requirement includes a single-sign-on mechanism[88].

In order to come up with a clear idea about which one of the most well-known access control methodologies is suitable for being used in cloud computing environments, we have examined all the access control methodologies mentioned earlier in this section against almost all factors present in this subsection. The results of our comparison are illustrated in table 1.

No.	**Comparison criterion**	**DAC**	**MAC**	**RBAC**	**ABAC**	**R-BAC**
1.	Least privilege principle	N	N	Y	Y	Y
2.	Separation of duties	N	N	Y	Y	N/A
3.	Binding of Duties	N	N	Y	Y	N/A
4.	Auditing	Y	Y	Y	Y	Y
5.	Syntactic and semantic support	N	N	N	N	N
6.	Policy management	N	N	N	N	Y
7.	Flexibilities of configuration	N	N	Y	N	N

8.	Operational and situational awareness	N	N	N	N	Y
9.	Response time	N/A	N/A	N/A	N/A	N/A
10.	Integrated with authentication functions	N	N	N	N	N
11.	OS compatibility	Y	N	Y	N	N
12.	Testing and verifying the AC functions	N/A	N/A	N/A	N/A	N/A
13.	Supporting passive and active workflows	N	N	N	N	N
14.	Supporting vertical and horizontal scope	N/A	N/A	N/A	N/A	N/A
15.	Delegation of capabilities	Y	N	N	N	N
16.	Dealing with heterogeneity	N	N	N	N	Y
17.	Transfer a customer's credentials across layers	N	N	N	N	N
18.	Scalability	N	N	Y	N/A	N/A
19.	Flexibility in attribute management	N/A	N/A	N/A	Y	N/A
20.	Computation complexity	N/A	N/A	N/A	N/A	N/A
Y= Yes, N= No and N/A= Not applicable						

Table 3.1: Results of applying the criteria on various access control models

3.3 Summary

This chapter presented our new detailed investigation into security requirements of four different critical infrastructure providers. The importance of critical infrastructures is hard to overestimate; the critical infrastructure is an essential asset for the maintenance of vital societal services such as power distribution networks and financial systems. As the benefits of cloud computing are hard to ignore, many critical infrastructure providers are aiming to utilise the unique benefits of cloud computing and migrate to the cloud computing paradigm. However, the critical infrastructure providers' security requirements might not be met by the cloud computing service providers. In addition, there were number of key elements of critical infrastructure providers' security requirements, yet an access control model was the top of the list.

Cloud computing would require robust isolation and controlling access mechanisms. Sharing of resources among potential untrusted tenants, multi-tenancy and virtualization, mechanisms to support transfer of customers' credentials across layers to access services and resources are crucial aspects in any access control model going to be deployed in cloud computing. Moreover, there are possibilities to extend existing access control models and use them in the cloud environment. However, this could be a potential risk and cannot solve the problem as conventional access models were proposed for a specific problem in a specific platform or environment and miss the remaining interconnected issues. This could happen due to non-existence of a complete list of access control requirements for cloud computing. In other words, the success on any access control solution for cloud computing will depend on analysing and accurately identifying a complete list of requirements. Thus, this chapter showed a novel cloud based access control criteria, which was used to propose an access control model for cloud computing. The following chapter will introduce the proposed Access Control Model for Cloud Computing (AC3).

CHAPTER 4 Access Control Model for Cloud Computing (AC3)

A cloud based access control model has been found one of the crucial security requirements for either cloud computing tenants or service providers. This motivates us to propose a cloud based access control model that can secure access to the data and fulfil the security requirements of cloud computing services providers and customers. This chapter sets out the Access Control Model for Cloud Computing (AC3). The proposed model uses the role and task principles to restrict access to cloud computing resources. It ensures a secure cloud that has sharing of physical resources among potential untrusted tenants. Moreover, the proposed model has three different security levels, which can be used according to level of trust. It copes with different access permission to the same cloud user and giving him/her the ability to use multiple services with regard to time of authentication and login. Various sensitivity levels of information to restrict who can read and modify information in the cloud are supported in the AC3. This chapter also explains how the proposed model ensures the secure sharing of resources among potentially untrusted tenants and its capacity to support different access permissions to the same cloud user. Implementation strategies, a security evaluation and case study are also presented in this chapter.

4.1 Access Control for Cloud Computing (AC3)

The proposed model facilitates the role and task principles as shown in figure 10. In the model, users are classified according to their actual jobs. Thus, users will be located on a security domain that relates to their role. Every role within the model will be assigned a set of the most relevant and needed tasks for practicing this role. Every task will have a security classification for accessing the data or assets, and the exact permissions needed for accomplishing this task. A risk engine is utilised to deal with dynamic and random behaviours of users; it credits consumers according to their access behaviours.

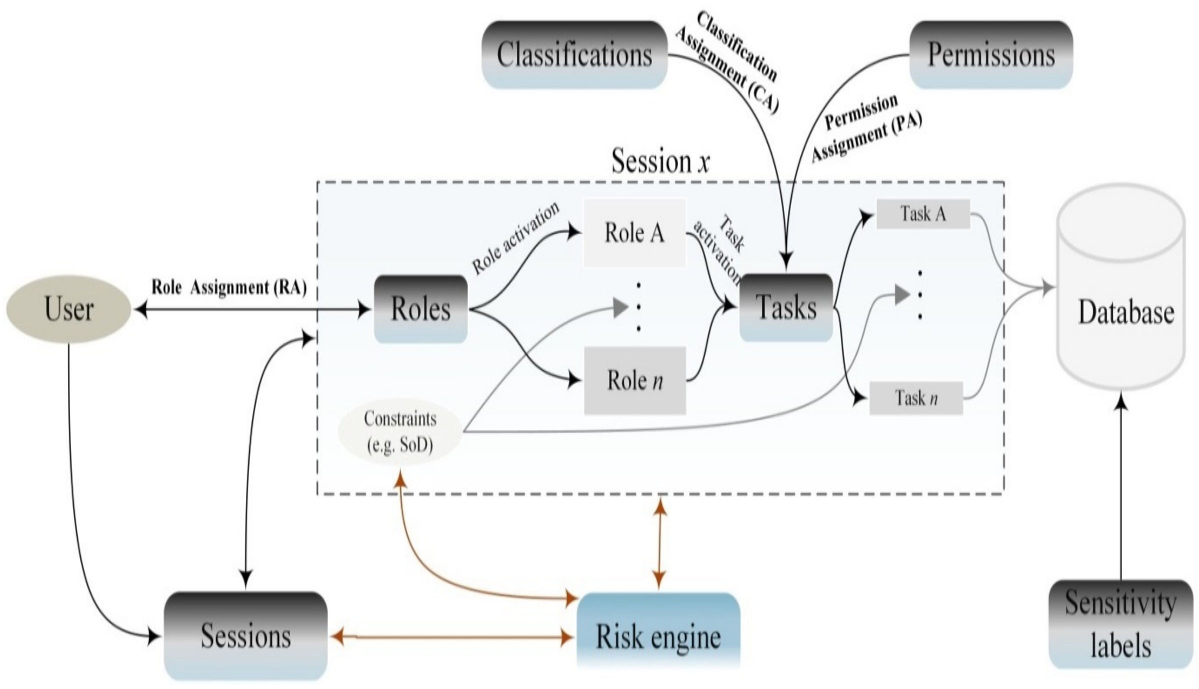
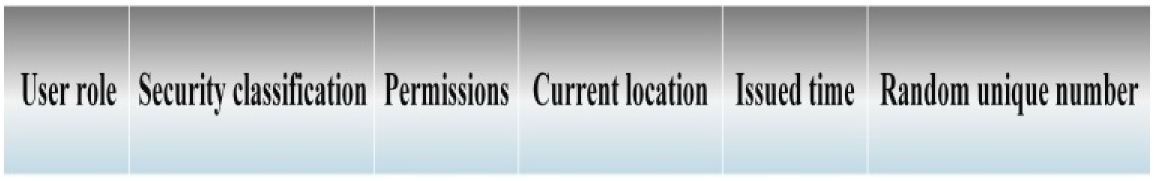


Figure 10: The level 1 in AC3 model

A security tags engine is also utilised for issuing security tags in semi or untrusted environments and processes. The model can secure access to data or assets by marking the data and assets with security labels. Any attempt to access the data has to ensure task classifications dominate the data or assets security labels. In our model, we utilise security tags in some circumstances according to the level of trust and security used within the environment. The security tag proposed will consist of a user role, classification, permissions, the current location, issued time and a random unique number as shown in figure11.



User role	Security classification	Permissions	Current location	Issued time	Random unique number

Figure 11: A security tag

The AC3 utilises a number of essential components and concepts in order to secure access to the cloud.

1. The AC3 has the following essential components:

   - Users (*U*) is a set of users.
   - Roles (*R*) is a set of roles.
   - Tasks (*T*) is a set of tasks.
   - Sessions(*S*) is a set of sessions.
   - Permissions (*P*) is a set of permissions.
   - Data (*D*) is a set of data.
   - User Assignment (*UA*) is a subset of intersection between *U* and *R*.
   - Role Assignment (*RA*) is a subset of intersection between *R* and *T*.
   - Permission Assignment (*PA*) is a subset of intersection between *P* and *T*.
   - Constraints (*Con*) are a set of constraints used in the system such as separation of duties and delegation.
   - Classifications (*Cla*) are a set of security classifications utilised to classify tasks in the model.
   - Sensitivity labels (*SL*) is a set of sensitivity labels used to restrict access to data according to its sensitivity.
   - Security tags (*ST*) is a set of security tags.

2. Every user u in the model can have an outlined number of roles $\{r_1, \dots, r_n\}$ where n is the total number r assigned to u. Every r can have a predefined number of tasks $\{t_1, \dots, t_m\}$ where m is the total number t. Every task t is assigned the exact needed permissions $\{p_1, \dots, p_k\}$ to accomplish its job where k is the permissions total, and a classification cla to access the targeted data or asset.

3. Most of the relationships in the model are many-to-many except user-to-session (a user can have one session) and task-to-classification (every task has a classification) relationships which are one-to-one.

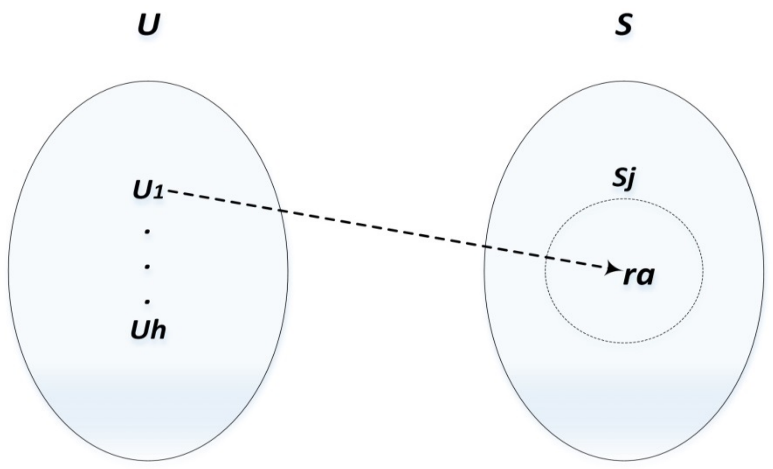


Figure 12: Assigning one session for every user in the AC3

- $\forall\, u_i \in U \rightarrow s_j \in S$.

  u_i cannot activate r_a outside s_j, but can use multiple roles as shown in figure 12. Where $1 < i < h$ (h is the total number of u), $1 < a < n$ and $1 < j < z$ (z is the total number of s).

- $\forall\, t_i \in T \rightarrow cla_j \in Cla$.

  Where $1 < j < m$ and $1 < j < f$ (f is the total number of cla),

- $\forall\, r_i \in R$ has a maximum number of authorised users u and activation at one time. Where$1 < i < n$.

- $UA \subseteq U \times R$. A many-to-many mapping of user-to-role assignments as shown in figure 13, where ((X and Y = U, R or T), (Z = UA, RA or PA)), q is the total of x and w is y's total.

- $RA \subseteq R \times T$. A many-to-many mapping of role-to-task assignments as shown in figure 13.
- $PA \subseteq P \times T$. A many-to-many mapping of permission-to-task assignments as shown in figure 13.

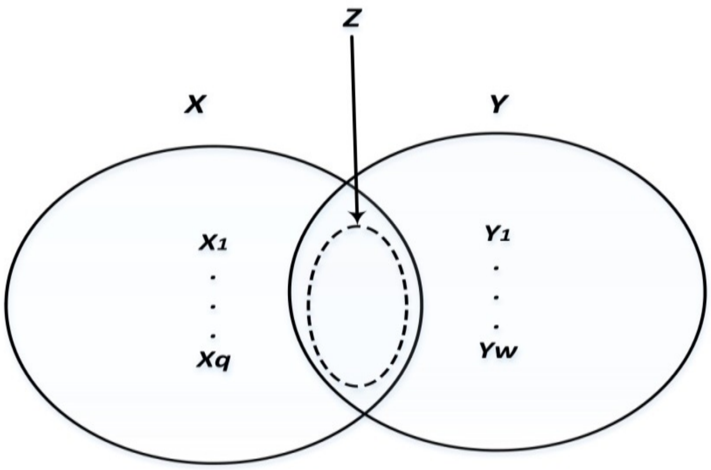


Figure 13: The AC3 relationships

4. This model supports a number of constraints as follows

- Least privilege principle

  It is the philosophy of granting a user u the only needed permissions p to accomplish its task t, even when u has more permissions than necessarily required for accomplishing tasks.

- Delegation of capabilities

  In order to make the model flexible and have dynamic resource management in the cloud computing environment, where users collaborate to fulfil their general tasks, the delegation of tasks is supported. However, delegation of tasks can happen only under the risk engine control, and between two subjects which have equivalent roles and work within the same area.

  $u_i, u_j \in U, t_a \in T, r_b \in R$
  u_i is assigned t_a but cannot finish it.
  Thus, an admin can delegate t_a to $u_j \leftrightarrow u_i, u_j \in r_b$ and in the same location.
  Where

$1 < i < h$ and $1 \ < j < h$, but $i \neq j$,
$1 < a < m$,
$1 < b < n$.

- Separation of duties

  Separation of Duties (SoD) is a principle that is supported by the least privilege principle as it aims at partitioning tasks and permissions associated to roles in order to prevent granting too much authority to one user. It also prevents the conflict of roles and interests. Dynamic SoD is supported in this model for either tasks or roles.

  $u_i \in U;\ r_a, r_j \in R;\ t_b, t_c \in T$
  $u_i\ activates\ r_a\ and\ r_j\ \leftrightarrow r_a \cap r_j = \emptyset$
  $r_a\ can\ activates\ \ t_b\ and\ t_c\ at\ the\ smae\ time\ \leftrightarrow t_b \cap t_c = \emptyset$
  Where
  $1 < i < h$,
  $1 < a < n$ and $1 < j < n$ 1, but $a \neq j$,
  $1 < b < m$ and $1 < c < m$, but $b \neq c$

5. Security labels are attached to data or a system’s assets to restrict access according to the degree of sensitivity. In AC3, a hierarchically ordered set of security labels is utilised, which are *Top Secret (TS)* > *Secret (S)* > *Confidential (C)* > *Unclassified (U).*

6. Classifications are given to tasks when they attempt to access data or a system’s assets. They have to dominate an object’s security label before accessing it. This model employs the same hierarchically ordered security labels set to classify tasks.

   $\forall\, t_i \in T \rightarrow cla_j\ \in Cla$ (assigned $\rightarrow$)
   $\forall\, d_a \in D \rightarrow sl_b \in SL$
   $t_i\ access\ d_a\ \leftrightarrow cla_j\ dominates\ sl_b$ (A task can access data if and only if its classification dominates the security level of data).
   Where
   $D = \{d_1, \dots, d_m\},\ 1 < a < y$ (y is the total number of d), $1 < i < m, 1 < j <$
   $f\ and\ 1 < b < e$ (e is the total number of sl)

7. Security tags are utilised in the AC3 for some situations especially in untrusted environment or employed processes.

   $\forall\ st_i \in ST \rightarrow \{r_j \in UA, cls, u_{current\ location}, t_a, RUN, p\}$

   Where
   $1 < i < w$ (w is the total number of st),
   $1 < j < n$,
   $1 < a < m$
   RUN (Random Unique Number) is used to ensure every security tag is unique,
   $p \in P$={Read (R), Write (W), Execute (E) and Delete (D)},
   The AC3 uses Supervision Role Hierarchy (S-RH) with strict inheritance.

8. Usually every task is a unique action that either can work alone or requires one or more actions to be triggered to fulfil its function. However, in some cases sequences of tasks are used to accomplish a job or one task relies on another task to finish its work. In such cases, the task that is accessing the data or assets directly has to issue a security tag, in order to prevent any leakage of privileges or disallowed access. Moreover, the security tag or its information can be passed to applications or processes utilised by the task according to system security level. Processes in the model can utilise security tags just in level 3. In the other levels, they can only get task classification and permissions from tasks that invoke them, in order to prevent the chance of any security tag being misused. Delegation of tasks can be insured by the risk engine, but the delegation has to follow the security and access policy. There is no delegation of security tags in the model, where a security tag is issued for a task then it delegates to another task or process. The following four scenarios illustrate and explain how a task or a sequence of tasks will employ a security tag and access the data.

   - For a task A that attempts to access data, a security tag st_A is generated according to the task classification as shown in figure 14.

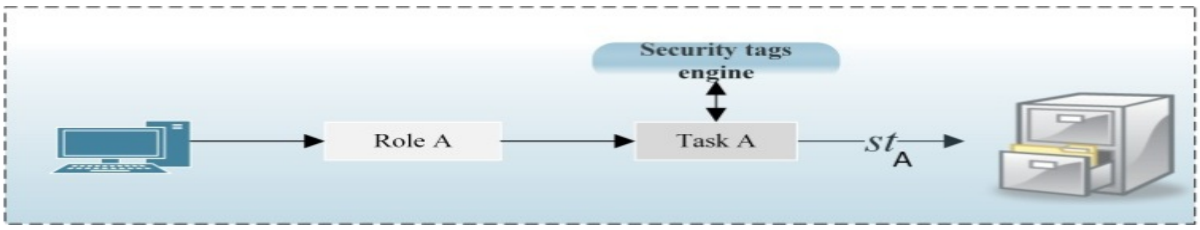


Figure 14: A task accessing data

- For a task A that attempts to access data indirectly and via another task B, a security tag is generated by the actual task accessing as shown in figure 15. In the figure, the *st*$_B$ is issued to because it is the current task accessing the data and a security tag for A might have more rights than B.

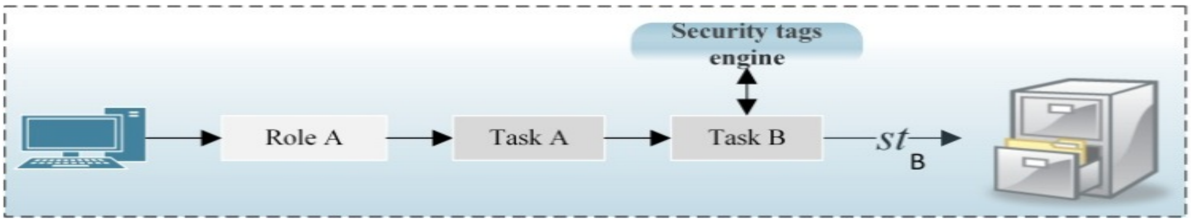


Figure 15: A task accessing data via another task

- Any process that requires access to data has to get a security tag, which is generated by the task employing the process as shown in figure 16.

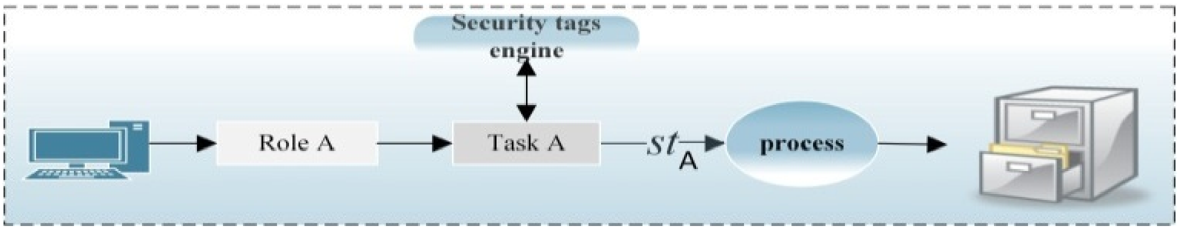


Figure 16: A task passing a security tag to an utilised process

- In order to prevent leakage of privileges, any process which needs to access data, has to get a security tag from the actual task attempting to access the data and employ it. In figure 17, a security tag is issued for task B not A and passed to the process, due to A has no direct connection to the process and might have more privileges than B.

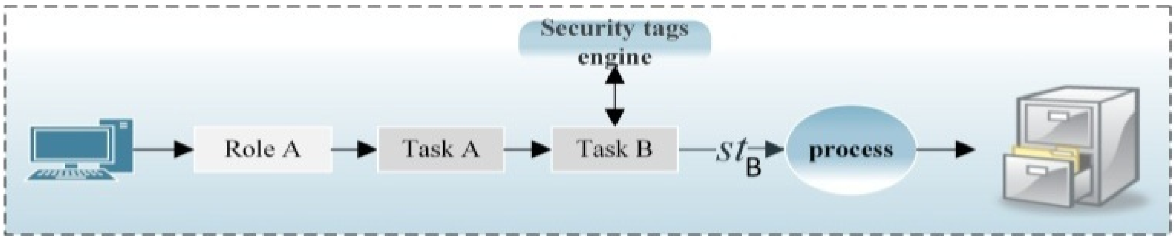


Figure 17: A process got a security tag from a task utilised by another task

9. The AC3 can work with all of the accessing data cases presented in table 2.

Where

$\{u_1, \ldots, u_h\}$ is a set of users and *h*is the total number of *u*,

$\{t_1, \ldots, t_m\}$ is a set of tasks and *m* is the *t's* total,

$\{d_1, \ldots, d_y\}$ is a set of data and *y* is the total number of *d*.

No.	**Case description**	**Case's figure**
1	A user uses a task to access (→) a piece of data	$$\begin{tikzcd} u_1 \arrow[r] & t_1 \arrow[r] & d_1 \end{tikzcd}$$
2	A user uses *m* of tasks to access a piece of data	$$\begin{tikzcd} & t_1 \arrow[dr] & \\ u_1 \arrow[ur, bend left] \arrow[r] \arrow[ddr, bend right] & t_2 \arrow[r] & d_1 \\ & \vdots & \\ & t_m \arrow[uur, bend right] & \end{tikzcd}$$
3	A user uses a task to access *k* pieces of data	$$\begin{tikzcd} & & d_1 \\ u_1 \arrow[r] & t_1 \arrow[ur, bend left] \arrow[r] \arrow[ddr, bend right] & d_2 \\ & & \vdots \\ & & d_y \end{tikzcd}$$
4	A user uses *m* of tasks to access *k* pieces of data	$$\begin{tikzcd} & t_1 \arrow[r] \arrow[ddr] & d_1 \\ u_1 \arrow[ur, bend left] \arrow[r] \arrow[ddr, bend right] & t_2 \arrow[r] & d_2 \\ & \vdots & \vdots \\ & t_m \arrow[uur] & d_y \end{tikzcd}$$
5	*h* of users use a task to access a piece of data	$$\begin{tikzcd} u_1 \arrow[dr] & & \\ u_2 \arrow[r] & t_1 \arrow[r] & d_1 \\ \vdots & & \\ u_h \arrow[uur, bend right] & & \end{tikzcd}$$
6	*h* of users use *m* tasks to access a piece of data	$$\begin{tikzcd} u_1 \arrow[r] & t_1 \arrow[dr] & \\ u_2 \arrow[r] \arrow[ddr] & t_2 \arrow[r] & d_1 \\ \vdots & \vdots & \\ u_h \arrow[uuur] & t_m \arrow[uur, bend right] & \end{tikzcd}$$
7	*h* of users use a task to access *k* pieces of data	$$\begin{tikzcd} u_1 \arrow[dr] & & d_1 \\ u_2 \arrow[r] & t_1 \arrow[ur, bend left] \arrow[r] \arrow[ddr, bend right] & d_2 \\ \vdots & & \vdots \\ u_h \arrow[uur, bend right] & & d_y \end{tikzcd}$$

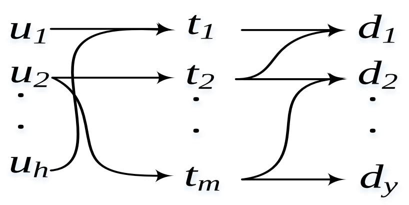
8	*h* of users use *m* tasks to access *k* pieces of data	

Table 4.1: Accessing data cases in AC3

4.1.1 The Model Security Levels

The system can provide three different security levels according to how secure the environment is, and how the processes are trusted:

1. A secure environment and trusted processes

   Here, there is no need to use security tags. Rules and permissions are enforced for every task as shown in figure 10. The tasks' characteristics and privileges can be passed to processes that are employed by tasks for doing their jobs. For example, a task can pass its security classification to any utilised process engaged in its operation.

2. Semi secure environment or trusted processes.

   In this level, security tags are utilised and have validation times as illustrated in figure 18. Their usage is restricted to validation times, and they cannot be passed to processes. When processes require access to data, they will use some information passed by a task utilising them such as classification, time and location.

3. Unsecured environment and untrusted processes.

   It uses the same principle presented in figure 18, yet a security tag is issued for every access to data or assets in order to prevent any chance of reusing the security tag or cheating by either tasks or processes. For instance, a task has to be generated a unique security tag used once for every access or utilised process engaged in its operation.

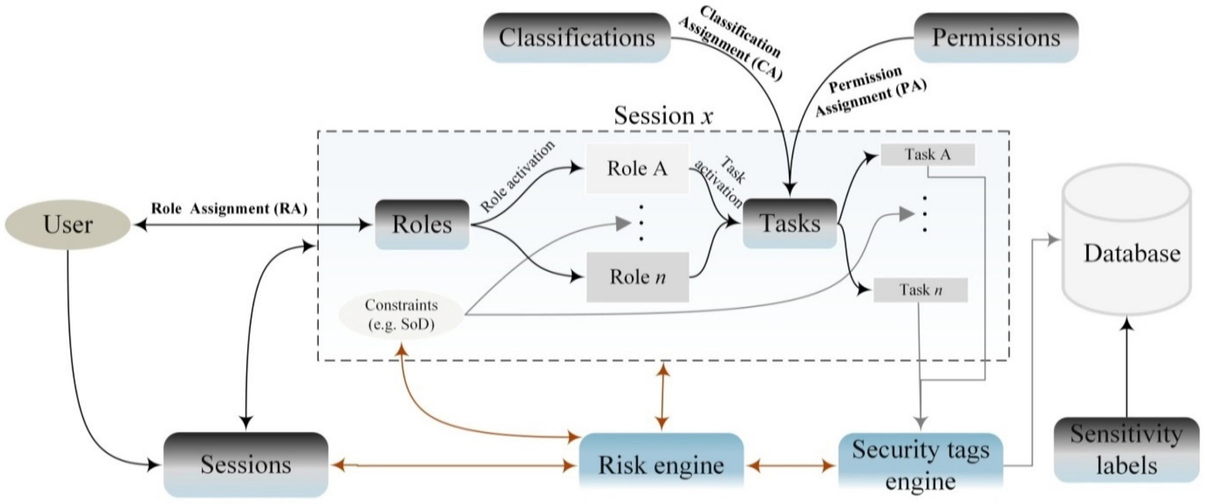
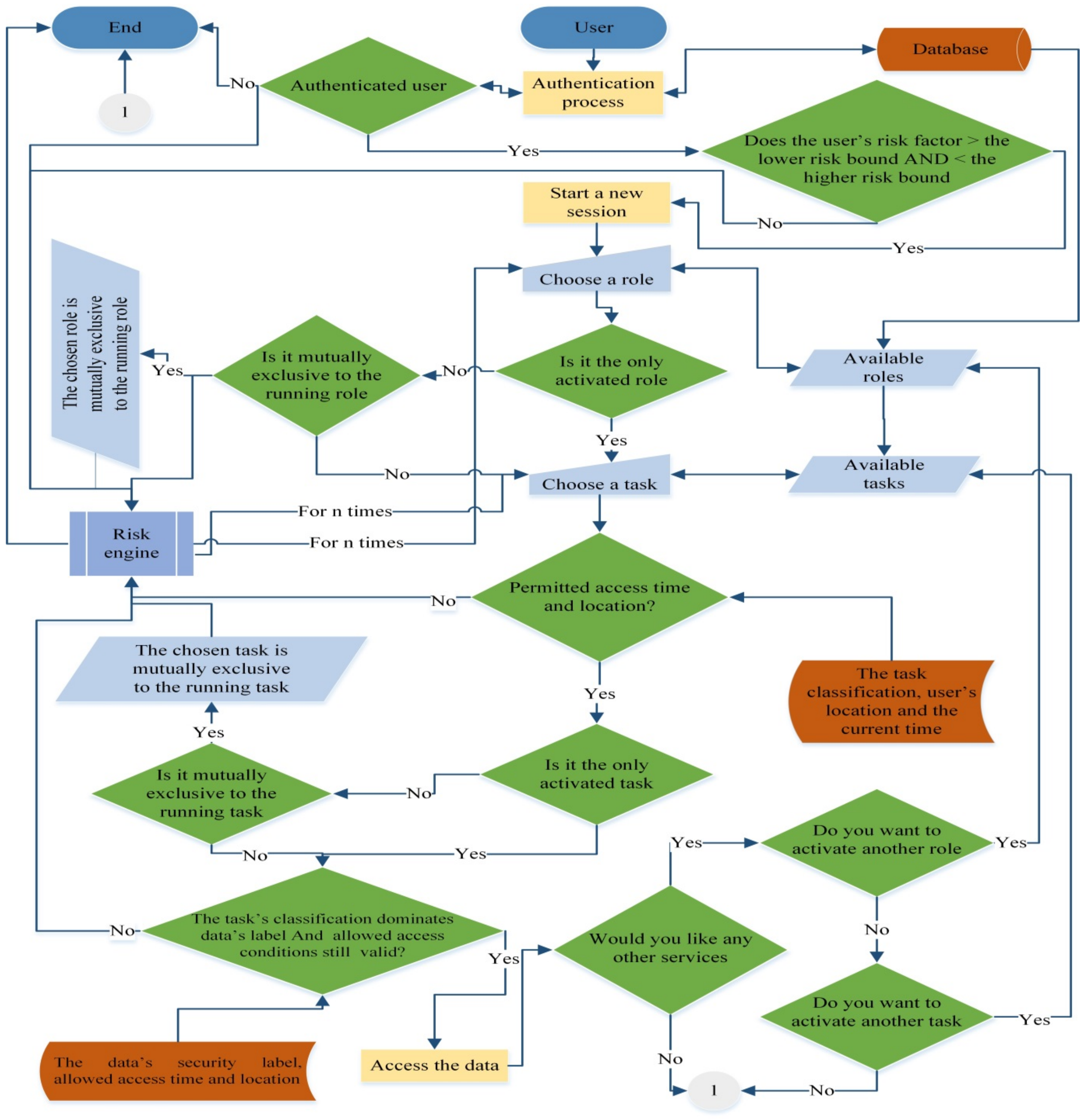


Figure 18: Access Control for Cloud Computing (AC3) (Level 2&3)

4.1.2 The Proposed Model's Flow Charts

The proposed model supports three levels of security. All of them share the same principles in terms of the rules and permissions which are enforced for every task. After a user is authenticated, the user's risk factor will be tested against predefined lower risk and higher risk bounds. Both of these values are under the control of the risk engine and adapted according to the user's behaviour. If the user's risk factor resides within the both risk bounds, a new session will be started and a new role will be chosen by the user. Furthermore, the user has a list of available roles dedicated to him/her by a demonstrator. A list of tasks for the user's role will be available for performing.



End
User
Database
Authenticated user
Authentication process
No
Yes
1
Does the user's risk factor > the lower risk bound AND < the higher risk bound
Start a new session
No
Yes
The chosen role is mutually exclusive to the running role
Choose a role
Is it mutually exclusive to the running role
Is it the only activated role
Available roles
Yes
No
No
Yes
Choose a task
Available tasks
For n times
For n times
Risk engine
Permitted access time and location?
No
Yes
The chosen task is mutually exclusive to the running task
The task classification, user's location and the current time
Yes
Is it mutually exclusive to the running task
Is it the only activated task
No
No
Yes
Do you want to activate another role
Yes
Yes
The task's classification dominates data's label And allowed access conditions still valid?
No
Yes
Would you like any other services
No
Do you want to activate another task
Yes
The data's security label, allowed access time and location
Access the data
No
1
No

Figure 19: The flow chart of solution's level 1

However, in level one no security tags are used as shown in figure 19. In this level, the author assumed the model is deployed in a trusted environment. Thus, the model's level one uses only security labels and classifications to secure access to resources. If the task's classification dominates the data's security label, the task in this case only can access the data or any of a system's resources. The tasks' characteristics and privileges can be passed to processes that are employed by tasks for doing their jobs. For example, a task can pass its security classification to any utilised process engaged in its operation.

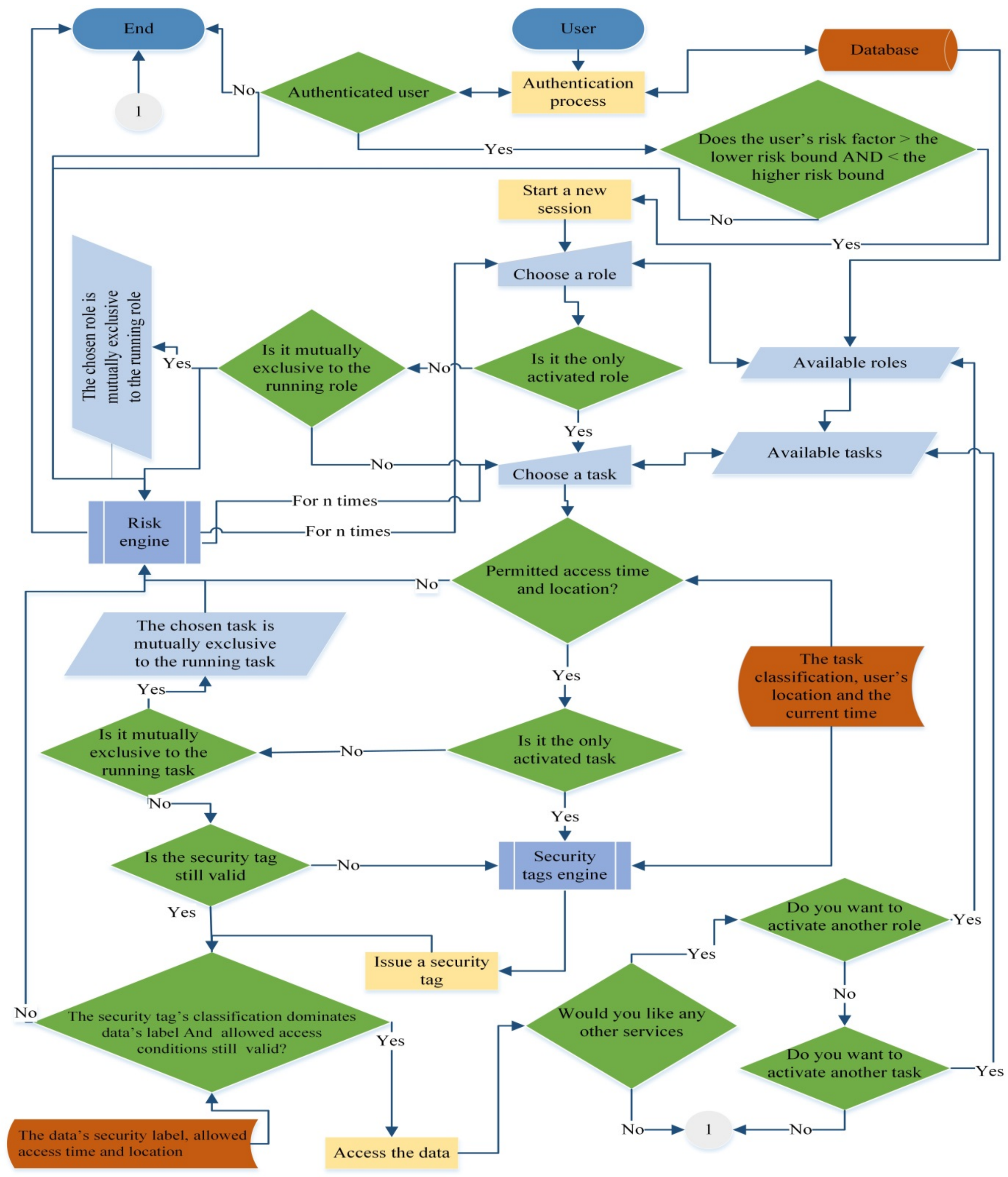
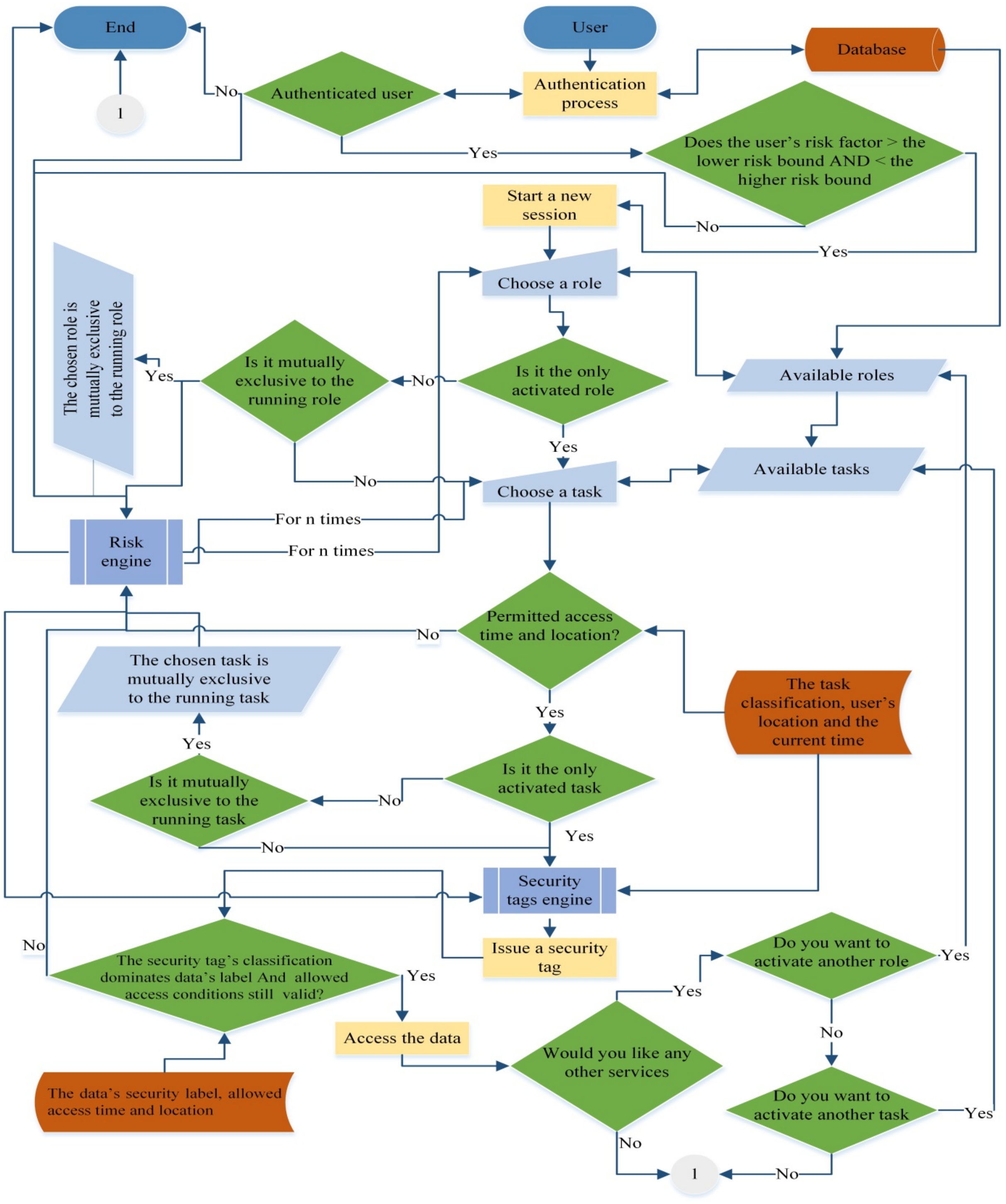

End
User
Database
1
No
Authenticated user
Authentication process
Yes
Does the user's risk factor > the lower risk bound AND < the higher risk bound
Start a new session
No
Yes
The chosen role is mutually exclusive to the running role
Yes
Is it mutually exclusive to the running role
No
Choose a role
Is it the only activated role
Available roles
Yes
No
Available tasks
Choose a task
For n times
For n times
Risk engine
No
Permitted access time and location?
The chosen task is mutually exclusive to the running task
Yes
The task classification, user's location and the current time
Yes
Is it mutually exclusive to the running task
No
Is it the only activated task
No
Yes
Is the security tag still valid
No
Security tags engine
Yes
Do you want to activate another role
Yes
Yes
Issue a security tag
No
No
The security tag's classification dominates data's label And allowed access conditions still valid?
Yes
Would you like any other services
Do you want to activate another task
Yes
The data's security label, allowed access time and location
Access the data
No
1
No

Figure 20: The flow chart of solution's level 2

In the model's second and third levels, security tags are utilised and issued by a security tags engine to strength the model in untrusted environments as was explained in section 4.1.1. On the second level as illustrated in figure 20, a security tag has a validation time. Each security tag is restricted to a validation time, and it cannot be passed to processes. In addition, every time a process attempts to access data, it will use some information passed by a task using them such as classification, time and location.



End
User
Database
1
No
Authenticated user
Authentication process
Yes
Does the user's risk factor > the lower risk bound AND < the higher risk bound
Start a new session
No
Yes
Choose a role
The chosen role is mutually exclusive to the running role
Yes
Is it mutually exclusive to the running role
No
Is it the only activated role
Available roles
Yes
No
Choose a task
Available tasks
For n times
Risk engine
For n times
Permitted access time and location?
No
The chosen task is mutually exclusive to the running task
Yes
The task classification, user's location and the current time
Yes
Is it mutually exclusive to the running task
No
Is it the only activated task
No
Yes
Security tags engine
Issue a security tag
No
The security tag's classification dominates data's label And allowed access conditions still valid?
Yes
Do you want to activate another role
Yes
Yes
Access the data
Would you like any other services
No
The data's security label, allowed access time and location
Do you want to activate another task
Yes
No
1
No

Figure 21: The flow chart of solution's level 3

Furthermore, security tags utilised in the third level in the proposed solution are issued for every access to data or resources in order to prevent any chance of reusing the security tag or cheating by either tasks or processes. As shown in figure 21, a security tag is generated for every task going to access data or resources and it will be passed to any utilised process engaged in its operation.

4.1.3 The Analysis

Despite using the T-RBAC as a backbone to the proposed model because of the T-RBAC simplicity and flexibility in the configuration, it needs to be developed and extended for cloud computing to support the delegation principle, dynamic and random behaviours of users, local and global access. Data mostly has various degrees of sensitivity, which needs to be considered in any access control system. The most well-known access control system uses data sensitivity as a factor for either granting or denying access via MAC. However, it is very expensive and difficult to deploy as illustrated in section 2.2.1 in chapter 2. Moreover, there is a significant gap between web applications utilised in application layers and lower layers as the system components and processes in lower layers are treated as fully trusted. Nevertheless, they should not be fully trusted. The dynamic activation of roles and permissions are difficult to achieve in cloud computing as relationships between users and resources are active. Service providers and users are likely to be in different security domains. In addition, the dynamic and random behaviours of users are a big concern and challenge for access control systems developers, as users have no time or location restrictions.

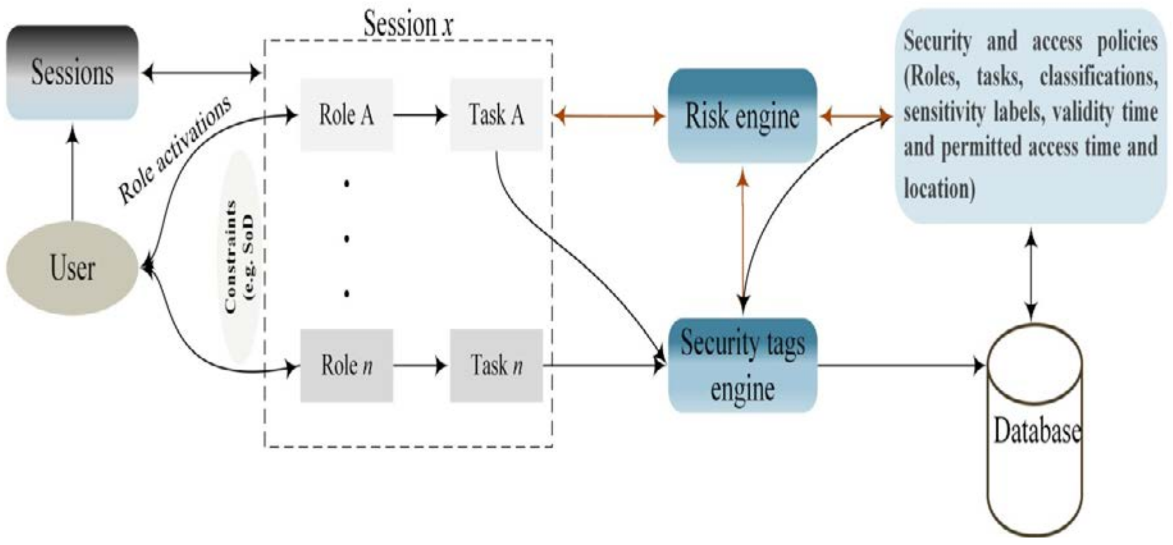

Session x
Sessions
Role activations
Role A
Task A
Constraints (e.g. SoD)
Role n
Task n
User
Risk engine
Security tags engine
Security and access policies (Roles, tasks, classifications, sensitivity labels, validity time and permitted access time and location)
Database

Figure 22: The AC3 block diagram

The AC3 is considerably a big generic model, which cannot be easily implemented due to time constraint and the following reasons: Firstly, it needs to implement the risk engine that control and responsible for many tasks such dealing with heterogeneity, risk management and users' behaviours. It needs good algorithms to calculate risks and deal with any conflict happen between access policies. Secondly, the security tag engine is used for generating security tags that are issued for tasks and process. This engine has to be implemented and tested with varied conditions (i.e. issuing more than security tag for a user and its scalability). The security tags will be used for accessing data and resources; hence, they should not add any computation complexity or affect the response time. Thirdly, the AC3 needs algorithms to give security classifications to data and consider location and time of access. These classifications should not affect the AC3 overall performance. Finally, the AC3 has to have mechanism to control processes that access data and prevent any misuse of security classifications and tags.

The proposed system as shown in figure 22 can deal with all of the previous concerns by utilising the following concepts:

1. Role is considered as the natural way to control access to resources in organizations and enterprises. A subject’s responsibility is more important than who the subject is. Various general job functions are facilitated to create roles such as accounting role, secretary role and manager role. A role is defined by Sandhu et al. as “A role is a job function or job title within the organization with some associated semantics regarding the authority and responsibility conferred on a member of the role” [100]. Using roles give companies an ability to impose their constraints with full flexibility of adding or removing roles to their consumers according to their actual activities and jobs.

2. Task is another concept used in the AC3 to restrict permissions and access assigned to roles. Each user within a system is assigned a role; roles are given tasks that have permissions. Moreover, these permissions are assigned to roles in regards to their tasks and the given permissions dynamically change according to the task in hand. Authorization determines who can do which tasks with what role under which conditions.

3. Permissions in the model are activated and deactivated according to the current task or process state. These permissions are assigned to tasks, and the given permissions are dynamically changed according to users’ behaviours.

4. The AC3 model supports a set of constraints, which are location and time constraints, least privilege principle, separation of duties either static or dynamic and delegation of capabilities. However, the model has no restrictions about constraints that can be supported.

5. Classifications and security labels are used for controlling access to resources and information flow. Sensitivity labels are utilised to mark data internally according to their sensitivity and value, which are top secret, secret, confidential and unclassified. The model gives systems an ability to use their sensitivity labels. Any task or process employed by a task needs a classification to access resources, as there should be no access to any resource without a classification equal or dominant to the resource's sensitivity labels.

6. The AC3 uses a risk engine for dealing with a number of security concerns

    - Controlling and crediting users according to their previous and current behaviours.
    - Offering Policy management such as organizing relations between consumers, utilities and third parties.
    - Risk awareness.
    - Dealing with dynamic and random behaviours.

7. Security tag engines are utilised for generating security tags for tasks, which might be passed to employed processes. The security tags are used in untrusted environments for controlling access to a system's resources by either applications at higher levels or processes at lower levels. Moreover, the AC3 ensured the security of data on processing level when utilised processes access the data and resources by delegating them security classifications. For example, any task is invoked by a user's role in the AC3's level three has to a security tag to any process engaged in accessing the targeted data. However, all access control models cannot enforce their security policies on lower levels resources (RAM and CPU caches), which is solely under the direct control of CPU and ring 0 of operating systems. Thus, other researches needed to control overlapping access (processes and virtual machines) and data leakage (side-channel attacks) in lower levels resources.

4.1.4 A Case Study

A case study approach was used to allow us to investigate how the proposed model can be deployed in the cloud. The AC3 can offer a high level of security to the cloud's layers (Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS)) for both consumers and service providers. It gives administrators the ability to manage access and credentialing for their systems easily and securely as illustrated in figure 23.

We assume a system has been built on a cloud service provider infrastructure such as Amazon or Google. The system utilises the three different services cloud computing can offer (SaaS, PaaS and IaaS).

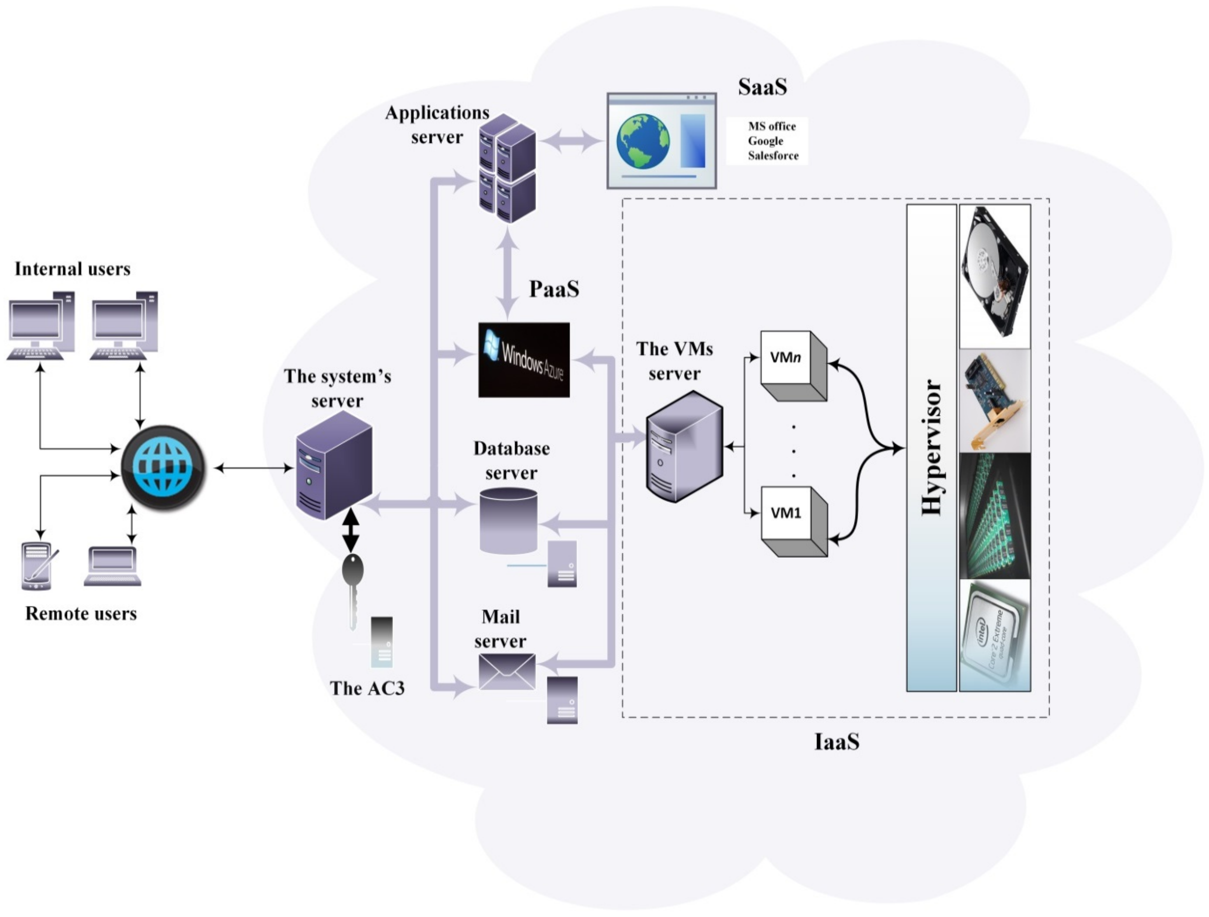


Figure 23: The AC3 in the cloud

1. SaaS

   Our model provides desirable levels of control and needed flexibilities to restrict access to SaaS applications such as Microsoft Office's 360. Access to applications can be up to systems' administrators, either classifying applications according to their value to systems and users, or declassifying them and allowing access according to consumers' demands and roles. The AC3 can restrict access by different ways:

- The Microsoft Office's 360 can have a security level like any other resource within the system. Thus, any task attempting to access this application has to have a classification that equals or dominates the security level of the application.
- The second way is restricting access to a number of outlined roles. Therefore, the application can have unclassified classification and be accessed by users who are members of the defined roles.
- Instead of giving a security level to every application, the targeted application can be added to a group of applications that have the same level of importance within the system. This group can have a security level to restrict access to it, be available to defined roles or be accessed by any consumers.
- The easiest way is declassifying the application and allowing access to any authenticated user.

2. PaaS

   The PaaS offers application developers great features and tools for developing and implementing new applications, which can be utilised within the system. The AC3 can provide administrators with the ability to restrict access to the PaaS, for example, classifies PaaS features and tools according to the importance of data or applications they accessed. Furthermore, in PaaS things can be a little bit complicated as the PaaS will be a medium between applications used by users and lower resources provided by hosted machines. Thus, systems' administrators can assign security labels to the PaaS applications as they are resources (objects) to the user and applications, yet they will delegate security classifications (subjects) to PaaS applications in order to access lower resources (virtual CPU and machines).

3. IaaS

   The model can have a considerable impact on controlling access to IaaS. The AC3 access policies define roles and tasks needed for every role. Every task is assigned the required permissions to accomplish its job and a security classification. The security classification is given to a task according to the role used for, and resources attempting to access. Moreover, it offers at level 2 and 3 an excellent feature to control access by using security tags. These tags are only generated by a security tag engine. They are also used for either one access or a defined time to the same task. Any new user joining the AC3 will assign a role or set of roles according to his/her actual job, and the user can activate his/her roles according to security and access policies. When the user invokes a role, the role will facilitate the user with a set of available tasks. For instance, if the user wants to access his/her emails in the mail server, s/he will utilise a task to access the mail server, and the task has to have classification which equals or dominates the mail server security level. The access ways presented in SaaS can be applied here. Hence, access to the mail server can be according to tasks or to any authenticated user.

No.	Comparison criterion	DAC	MAC	RBAC	ABAC	R-BAC	AC3
1.	Least privilege principle	N	N	Y	Y	Y	Y
2.	Separation of duties	N	N	Y	Y	N/A	Y
3.	Binging of duties	N	N	Y	Y	N/A	Y
4.	Auditing	Y	Y	Y	Y	Y	Y
5.	Syntactic and semantic support	N	N	N	N	N	N
6.	Policy management	N	N	N	N	Y	Y
7.	Flexibilities of configuration	N	N	Y	N	N	Y
8.	Operational and situational awareness	N	N	N	N	Y	Y
9.	Response time	N/A	N/A	N/A	N/A	N/A	N/A
10.	Integrated with authentication functions	N	N	N	N	N	Y
11.	OS compatibility	Y	N	Y	N	N	Y
12.	Testing and verifying the AC functions	N/A	N/A	N/A	N/A	N/A	Y
13.	Supporting passive and active workflows	N	N	N	N	N	Y
14.	Supporting vertical and horizontal scope	N/A	N/A	N/A	N/A	N/A	N/A
15.	Delegation of capabilities	Y	N	N	N	N	Y
16.	Dealing with heterogeneity	N	N	N	N	Y	Y

17.	Transfer a customer's credentials across layers	N	N	N	N	N	N
18.	Scalability	N	N	Y	N/A	N/A	Y
19.	Flexibility in attribute management	N/A	N/A	N/A	Y	N/A	N/A
20.	Computation complexity	N/A	N/A	N/A	N/A	N/A	N/A
	Y= Yes, N= No and N/A= Not applicable						

Table 4.2: Comparing AC3 with conventional access control models

4.1.5 Discussion

When we started to propose a cloud based access control model, we had to choose between two options: either proposing a new access control model from scratch or surveying what is available and can be utilised in the cloud. We chose to survey and analyse the available access control models and policies that can be adapted in cloud computing due to: Firstly, there are access control models that have a number of features that cannot be ignored. Therefore, adapting these features in the cloud can make cloud service providers and consumers confident about the level of control they will get. Secondly, the RBAC model is a well-known trusted model, which is used by enterprises and organizations. Therefore, it is worth using it as a base for any new proposed model. However, it cannot be deployed directly in the cloud (see section 2.2.1.3 in chapter 2). Moreover, some tenants (organizations or enterprises) use it in their internal networks, hence adapting it for cloud computing may encourage tenants to migrate to the cloud.

In order to validate the model, we compare it with conventional access control models. The comparison is present in table 3. It is also compared with a number of the proposed access control systems for cloud computing. The comparison is based upon security features that either the AC3 or other models can offer. Before proposing the model, we have reviewed almost every proposed access control system for cloud computing [16,32,49,50,51,52,53,54,55,56]. Most of them have not been validated or applied in a real cloud computing environment. Moreover, part of the proposed schemes target data outsourced and provisioned over the cloud [31,102]. Others adapt the conventional access control models [50,51].

1. Principles

The model can support a set of well-known principles such as the least privilege principle, delegation of capabilities, separation of duties (static and dynamic), binding of duties (role and participant levels) and temporal constraints (location and time). To the best of our knowledge, only one scheme has addressed all of these principles [50].

2. Support passive and active workflows

   The AC3 supports passive and active workflows, where roles are passive workflow and tasks are active workflow. There is only one approach which offers both workflows and this model targets health care systems [50].

3. Auditing

   The model provides a unique and novel way to control access, which utilises auditing and logs information to control and credit users according to their previous and current behaviours. This is done by the risk engine in the proposed model.

4. Policy management (add, delete, change, import, export)

   There is a huge demand for having proper policies, which can organize relations between consumers, utilities and third parties. The risk engine in the AC3 offers a policy management by dealing with policies conflicts, add deletes. Almost none of the reviewed models [50,51,31,102] have dealt with or mentioned policy management.

5. Dealing with heterogeneity and syntactic & semantic support

   Heterogeneity can happen in access control systems using various types of mechanisms, domains and policies. The model relies on the risk engine for policy management, which has to cope with heterogeneity caused by security policies and support different access control languages to enforce various logical expressions of rules. Hence, it is supported in this model. The ontology principle was used in [49] and[51] for heterogeneity problems, however they need good ontology transformation algorithms to compare the similarity of different ontologies. A new back-end database schema to support O-RBAC is also needed for these schemes. The authors in [102] proposed Uniform Resource Identifier (URI) as a based attribute representation. They combined it with the Resource Description Framework (RDF) for dealing with heterogeneity threats caused by using different attributes within the system. However, deploying the

URI and RDF techniques needs to be tested properly as they have a significant effect in the used attributes, and the Semantic Web Rule Language needs to be compatible with the existing semantic web standards.

6. Quality of service

   As the model has not been implemented, the computation complexity and response time have not been tested. Computational complexity of the AC3 has to be examined as it can affect the efficiency and quality of service of the model. In addition, the response time to grant access decisions still an open question to the AC3.

7. OS compatibility

   Due the AC3 utilises the role and task principles that supported in many platforms and operating systems, we are confident the AC3 can be supported and used in the currently used operating systems such as Windows and Linux.

8. Flexibilities of configuration and attribute management

   The AC3 model is flexible enough to be configured in the cloud computing environment as it utilises the role and task principles, which guarantee easy and adaptable assignment and ease of privilege. Moreover, The AC3 does not use or support attributes in the process of granting access decisions. In the proposed models, one model uses the task and role techniques [50]. Another model utilises the attribute and role principles, which combine the RBAC and ABAC, but by combining them this makes things more complicated, as they combine them without making further improvements [52]. The rest of proposed models employ only the RBAC, yet it suffers from dynamic activation of access rights for certain tasks [16,32,49,51,104].

9. Operational and situational risk awareness

   The risk engine in the AC3 supports risk awareness, which gives the capability to deal with the risk associated with accessing a system's resources. According to our knowledge, none of the reviewed models or approaches provides risk perception.

10. Integrated with authentication functions

In the current access control systems, there is a semantic gap between the user authentication mechanisms implemented by the web application and the authorization mechanisms implemented by the lower layers. Hence, the model links the authentication functions with the authorization mechanisms and controls applications and processes at lower levels, accessing data via security tags issued by the security tags' engine. To the best of our knowledge, all the reviewed models have not offered any method for dealing with this concern.

11. Scalability

    By utilising role and task principles that classify consumers to a number of domains, we are confident the AC3 is scalable and can deal with large numbers of users and administrators as well. For example, customers can be grouped in specific security domains, which can be created for number of reasons, for example, specific locations, types of employment contracts, etc. Moreover, administrators can be divided to different security domains with variant privileges as the cloud computing needs a considerable number of administrators with various assigned privileges to control access to cloud's resources. There is only an approach which uses both principles, but it does not mention how administrator scalability can be ensured [50].

12. Dealing with dynamic and random behaviours

    The AC3 utilises the risk engine in order to deal with any dynamic and random behaviour. It credits consumers' according to how they follow security and access policies, such as accessing the system from different locations or launching mutually exclusive roles or tasks. In addition, the risk engine assigns default risk lower and upper bounds for every new user. When the user starts accessing a system's resources, his risk lower and upper bounds will be changed according to his/her behaviours, either s/he can be assigned more permissions or his/her permissions can be reduced. Thus, a user's random behaviours will be observed by the risk engine, which will test the user's random behaviour against the user's risk lower and upper bounds and access policies, then either the risk engine grants or denies access to resources and record that in the user's record. The trust principle is used in most of the models proposed for cloud computing [16,105]. However, in our view, calculating the trust degree is not the best choice for coping with dynamic and random behaviours of consumers. In these models, the trust degrees are calculated by consumers or service providers, yet it is not clear how they are validated. Moreover, scores related to a trust level are calculated by users or resources, but what mechanism is used, how the trust level can change the access level and how misbehaving problems can be prevented is not explained.

13. Transfer customers’ credentials across layers

    The cloud needs access control policies that support mechanisms to transfer customers’ credentials across layers to access several clouds’ services and resources This requirement includes a single-sign-on mechanism [88]. The AC3 does not support these features as it targets the authorisation levels and these features have to be supported by the authentication mechanisms.

14. Testing and verifying the access control functions

    They are crucial features in cloud based access control models. Having capabilities to test and verify system functions or new updates can promote the level of security needed in such environments. Thus, the AC3 uses the risk engine to support testing and verifying feature enforced by its security and access policies. To the best of our knowledge, there is no access control model support these features.

15. Dealing with access control attacks

    In the background chapter, various types of access control attacks were reported, which target either application levels or lower levels. Defences against the application levels attacks such as password attacks or hijacking, are hugely dependent on authentication methods and mechanisms, which was out of the scope of this research. Moreover, lower levels attacks, for instance, cache side-channel attacks that can bypass authorization mechanisms have to be addressed by the AC3. In the next chapter, prevention and detection solutions that can be attached to the AC3 to defend against cache side-channel attacks are demonstrated.

4.2 Summary

In this chapter, a new Access Control Model for Cloud Computing (AC3) has been introduced. The proposed model meets the security requirements of cloud computing services providers and customers. The AC3 supports various sensitivity levels of information to restrict accessing and modifying information in the cloud. The proposed model uses the role and task principles to restrict access to cloud computing resources and make assigning privileges very dynamic. The AC3 ensures a secure cloud that has sharing of physical resources among potential untrusted tenants by using risk and security tags engines. In addition, the proposed model has three different security levels, which can be used according to level of trust. This chapter also showed why a risk engine is utilised to deal with dynamic and random behaviours of users. In addition, a security evaluation and analysis, case study and implementation strategies were illustrated in this chapter.

The proposed AC3 model has number of components, which secures access to resources in the application and processing levels. Securing access to data in the processing levels require controlling access to information on the RAM and CPU caches. Furthermore, the investigation of how data are processed and accessed in the processing levels such as RAM and CPU caches has shown that most access control models cannot be applied in the RAM and CPU caches as they are under direct control of the CPU and operating systems. Moreover, processes used in any operating system will overlap access to the same physical CPU caches' addresses. Using a spy process can tell exactly which memory addresses have been accessed and by whom.

Multiple regression analysis revealed that securing access to information in lower levels has the same importance or might be more important to tenants of cloud computing than high and application levels. Therefore, our focused in the next chapter is how the proposed AC3 model can deal with the lack of data isolation in lower levels (CPU caches), which could lead to bypass access control models to gain some sensitive information by using cache side-channel attacks. The following chapter will present a new prevention and detection solutions to cache side-channel attacks that can be attached to the AC3.

CHAPTER 5 The Prevention and Detection Solutions to Cache Side-Channel Attacks in Cloud Computing

The proposed AC3 model has number of components, which secures access to resources in the application and processing levels. Securing access to data in the processing levels requires controlling access to information on the RAM and CPU caches. In this chapter we will focus on the processing levels and how the proposed AC3 can prevent and detect any leakage of information in lower levels (CPU caches). Hence, we specifically look at the lack of data isolation in lower levels (CPU caches) and prevent any leakage of data happen by cache side-channels attacks. This chapter presents a new Prime and Probe cache side-channel attack, which can prime physical addresses. These addresses are translated form virtual addresses used by a virtual machine. Then, time is measured to access these addresses and it will be varied according to where the data is located. If it is in the CPU cache, the time will be less than in the main memory. This chapter also details the fundamental principles and design of novel prevention solutions to cache side-channel attacks in the cloud computing. It is a new way to prevent any leakage of information caused by cache side-channel attacks. It erases a virtual machine's data from all CPU cache levels when the virtual machine relinquishes CPU resources in order to prevent any chance of launching a cache side-channel attack. Moreover, a new detection infrastructure solution that detects cache side-channel attacks in cloud computing is also presented here. The solution does not rely on any software or application to detect any abnormal behaviour in the CPU caches. It uses a new way to measure the time taken by every fetch cycle executed by a virtual machine. Both solutions are free from any modifications which have to be carried on the operating systems of cloud tenants.

5.1 A New Prime and Probe attack

As explained in the background section, the Prime and Probe attacks measure the time needed to read data from memory pages associated with individual cache sets. Thus, it primes a number of cache sets with data and waits till the data is accessed by another virtual machine. In order for this attack to be successful, an attacker needs to firstly, build a link between memory pages and CPU cache sets; and then fill cache sets with data. However, both of them require sophisticated techniques and a considerable number of predefined conditions, which we found in some cases unrealistic.

The new Prime and Probe attack is a variant of prime and probe attacks, which relies on revealing sensitive information from the sharing of memory pages and CPU cache lines. In this new attack, an attacker does not have to have any information about the link between memory pages assigned to CPU cache lines or number of cache lines. The attacker needs to know the virtual addresses space for his/her virtual machine or a number of virtual addresses used by processes in the virtual machine. These addresses will be linked to physical addresses which host them to examine whether another virtual machine accesses them or not. As illustrated in figure 24, the virtual addresses will be translated to physical addresses, which will be primed with the attackers' data. Having primed the physical addresses, the attacker will wait for a predefined time to let the victim access the CPU caches and memory. When the victim relinquish the CPU resources, the attacker, firstly, measures the time to access the monitored addresses and then checks data kept in these addresses whether changed or not. If the time to access the monitored addresses is less than the time to access the last level in CPU caches (e. g. L3), the monitored addresses have definitely been accessed by the victim and the data on these addresses is changed.

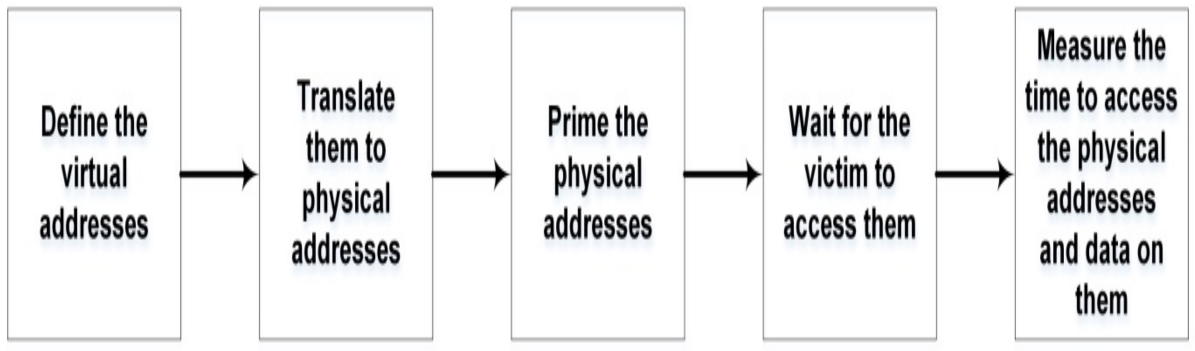
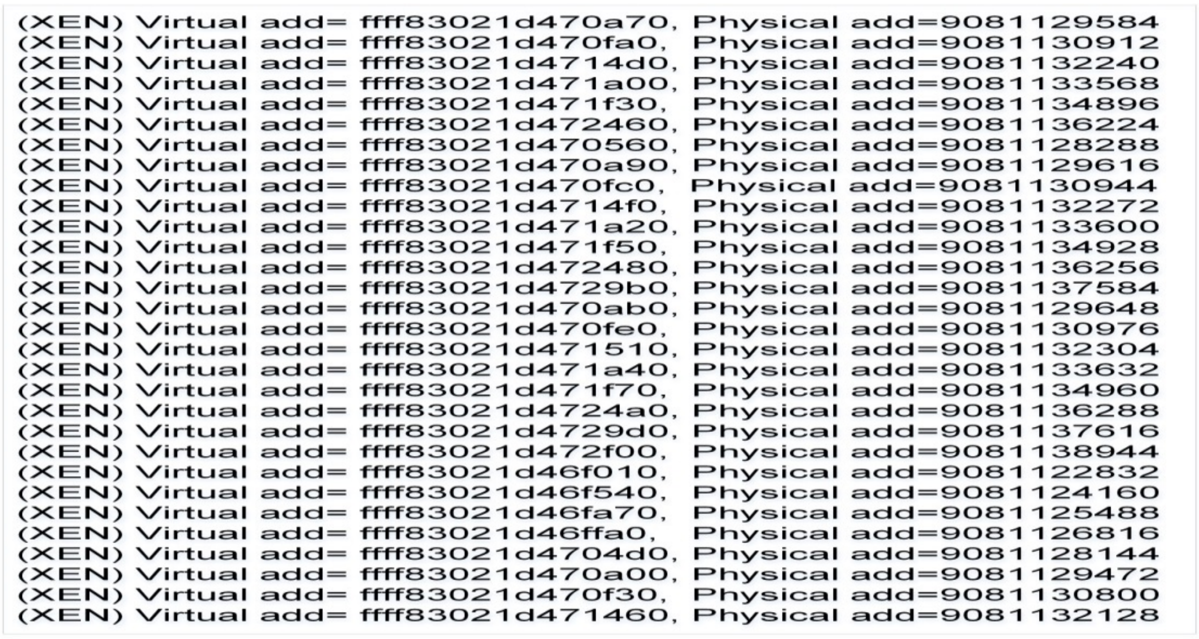

Define the virtual addresses
Translate them to physical addresses
Prime the physical addresses
Wait for the victim to access them
Measure the time to access the physical addresses and data on them

Figure 24: The new Prime and Probe attack

The implementation of the attack follows these steps:

1. Getting virtual addresses used by the attacking virtual machine is an important step that can be done by looking at the virtual addresses space given to the virtual machine or driving them from the running code. In our case, we used a function implemented in the kernel of Xen hypervisor to print all virtual addresses used by running virtual machines, which were two. The function's source code is presented in the first part in appendix c.

2. We used walk page tables provided in the Xen source code to translate virtual addresses. In this function, every virtual address will be translated to linear addresses, which will be used in the page tables (4 levels) to get the physical addresses. In addition, a sample of translated addresses that printed on the Xen command prompt is presented in figure 25. The source code of the function is presented in the second part in appendix c.



```
(XEN) Virtual add= ffff83021d470a70, Physical add=9081129584
(XEN) Virtual add= ffff83021d470fa0,  Physical add=9081130912
(XEN) Virtual add= ffff83021d4714d0, Physical add=9081132240
(XEN) Virtual add= ffff83021d471a00, Physical add=9081133568
(XEN) Virtual add= ffff83021d471f30,  Physical add=9081134896
(XEN) Virtual add= ffff83021d472460, Physical add=9081136224
(XEN) Virtual add= ffff83021d470560, Physical add=9081128288
(XEN) Virtual add= ffff83021d470a90, Physical add=9081129616
(XEN) Virtual add= ffff83021d470fc0,  Physical add=9081130944
(XEN) Virtual add= ffff83021d4714f0,  Physical add=9081132272
(XEN) Virtual add= ffff83021d471a20, Physical add=9081133600
(XEN) Virtual add= ffff83021d471f50,  Physical add=9081134928
(XEN) Virtual add= ffff83021d472480, Physical add=9081136256
(XEN) Virtual add= ffff83021d4729b0, Physical add=9081137584
(XEN) Virtual add= ffff83021d470ab0, Physical add=9081129648
(XEN) Virtual add= ffff83021d470fe0,  Physical add=9081130976
(XEN) Virtual add= ffff83021d471510, Physical add=9081132304
(XEN) Virtual add= ffff83021d471a40, Physical add=9081133632
(XEN) Virtual add= ffff83021d471f70,  Physical add=9081134960
(XEN) Virtual add= ffff83021d4724a0, Physical add=9081136288
(XEN) Virtual add= ffff83021d4729d0, Physical add=9081137616
(XEN) Virtual add= ffff83021d472f00,  Physical add=9081138944
(XEN) Virtual add= ffff83021d46f010,  Physical add=9081122832
(XEN) Virtual add= ffff83021d46f540,  Physical add=9081124160
(XEN) Virtual add= ffff83021d46fa70,  Physical add=9081125488
(XEN) Virtual add= ffff83021d46ffa0,   Physical add=9081126816
(XEN) Virtual add= ffff83021d4704d0, Physical add=9081128144
(XEN) Virtual add= ffff83021d470a00, Physical add=9081129472
(XEN) Virtual add= ffff83021d470f30,  Physical add=9081130800
(XEN) Virtual add= ffff83021d471460, Physical add=9081132128
```

Figure 25: Translating virtual to physical addresses

3. Priming physical addresses can be denied by the operating system or segmentation fault. Hence, there is a command which should be used to assign data to physical addresses. It is malloc (kmalloc) instruction.

The kernel module prime function

```
Static void prime (unsigned long phy_add)
{
   unsigned long *a =(unsigned long*) malloc (sizeof(unsigned long) *
phy_add);
   *a=0x2000000000000000;
}
```

---

4. After priming the addresses, the attacker will wait until the victim accesses the addresses.

5. The attacker will take control of the CPU resources in this step and measure the time to access the same addresses. We used Read Time Stamp Counter (RDTSC) assembly instruction to measure cycles taken to read the addresses. If the time is less than 120 CPU cycles (threshold reflects the longest time to read data from the last level (L3) in the CPU caches levels), at least one of the addresses is accessed by the victim and data in each address accessed by the victim is checked.

The kernel module measure function

```
static inline unsigned long long rdtsc13(void)
{
  unsigned long hi, lo;
  asm volatile ("rdtsc" : "=a"(lo), "=d"(hi) : : "ebx", "ecx");
  return ((unsigned long long)lo) | (((unsigned long long)hi)<<32) ;
}
```

6. Once the monitored addresses have been accessed by the victim, the attacker can then keep monitoring these addresses and collect information about data kept in these addresses either data or addresses point to data. Furthermore, the attack's results are presenting in section 7.1.1.3 in chapter 7.

This attack can be used to extract fine grained information about data processed and addresses accessed by virtual machines. Therefore, solutions to prevent these types of cache side-channel attacks are urgently needed to be deployed in cloud computing.

5.2 A Novel Lightweight Solution to Prevent Cache Side-Channel Attacks in Cloud Computing

The proposed solution is a novel lightweight solution that prevent cache side-channel attacks in cloud computing. It erases data or addresses linked to a virtual machine form all levels of CPU caches without affecting the CPU caches' performance or deleting other virtual machines' data kept in the caches. In addition, the proposed solution has been fully implemented in the Xen hypervisor's kernel of the host server. Thus, there is no need for any modification on tenants' virtual machines codes. Experiments in the lab illustrate that the solution effectively prevents cache side-channel attacks on all levels of time-shared CPU caches.

5.2.1 Our Goals

Most of the anticipated attacks on time-shared CPU caches involve triggering a spy process on the host machine that accommodates it and the victim in order to get information about cache sets accessed by the victim. In section 2.3.1 (background chapter), we have reported two kinds of access-driven side-channel attacks, which are Prime and Probe and Flush+Reload cache side-channel attacks. Both of them rely on calculating the time needed to read data from memory pages associated with an individual cache set. If the time is less than a threshold, then that data is on a cache set. Otherwise, it is not in the cache set. Thus, we aim to erase any driven information from accessing the CPU cache by erasing all data linked to the victim machine from the CPU cache levels without touching other data residing in the cache.

The proposed lightweight solution aims to satisfy the following goals:

1. The solution should erase all data related to a VM directly after its Virtual CPU (VCPU) releases the control of the CPU resources and does not erase or modify any data which does not belong to the targeted VM.

2. The solution should induce neglected or managed performance overhead.

3. The hypervisor should not require any major modifications, which can affect its usability or performance.

4. All the tenants' operating systems should be completely free from any modifications or changing in the operating system's libraries.

5. The solution should deal with all kinds of overlapping access (VCPU or CPU cores) and core migration, which are the direct cause of cache side-channel attacks. The solution should deal with the following cases:

    - Number of VMs or domains overlapping access to one VCPU.

    - Number of VMs or domains overlapping access to number of VCPUs.

    - Number of VCPUs overlapping access to a CPU core.

    - A VCPU migrates from a CPU core to another.

5.2.2 The New Lightweight Solution

In cloud computing, every cloud service provider has a set of Virtual Machines (VM) shared access to Virtual CPU (VCPU) that utilises the physical CPU resources such as Cores (CO) and Caches (CA). All of the service provider resources are shared among Users (U), which cause overlapping access to resources and lead to leakage of data by side-channels (cache side-channel attacks). Thus, the prevention solution will tackle these attacks by erasing all VM's data from all CA levels. Moreover, the prevention solution has two stages: trace stage and erasing stage. The trace stage continuously monitors any changes happens on the current state of a VCPU used by a VM (either a VM migrates to another VCPU or a VCPU migrates to another CO). If any changes detected in the VCPU's state, the erasing stage will start with defining the VM's Memory Pages (MP) and Virtual Addresses (VA), and then deleting them from the CAs.

To be more specific, the above sets of cloud service provider resources can be represented as follow:

1. $VM = \{vm_1, \dots, vm_n\}$

   Where *VM* is a set of virtual machines (*vm*) and *n* is the total number of *vm*

2. $VCPU = \{vcpu_1, \dots, vcpu_m\}$

   Where *VCPU* is a set of virtual cpu (*vcpu*) and *m* is the total number of *vcpu*

3. $U = \{u_1, \dots, u_k\}$

   Where *U* is a set of users (*u*) and *k* is the total number of *u*

4. $CO = \{co_1, \dots, co_l\}$

   Where *CO* is a set of cores (*co*) and *l* is the total number of *co*

5. $CA = \{level_1, level_2, level_3\}$

   Where
   CA is three levels (*level*) of caches,
   $level_1=\{line_1,\dots,line_p\}$ and *p* is the total cache lines in $level_1$,
   $level_2=\{line_1,\dots,line_o\}$ and *o* is the total cache lines in $level_2$,
   $level_3=\{line_1,\dots,line_z\}$ and *z* is the total cache lines in $level_3$,
   line is a line in any cache set

6. $MP = \{mp_1, \dots, mp_q\}$

   Where *MP* is a set of memory pages (*mp)* and q is the total number of *mp* assigned to a vm

7. $VA = \{va_1, \dots, va_x\}$

Where *VA* is a set virtual addresses (*va*) and x is the total number of *VA* a *vm* has

Every *vcpu* has a data structure to save information about *vm* using it, *co* assigned ($->$) to it, and etc. They are represented in form of *vcpu.vm* and *vcpu.co*. The solution can be expressed as follow:

```
$if\ ((vcpu.vm\ changed) or\ (vcpu.co\ changed))$
{
        $erase\ (vcup.vm-> mp_j)\ from\ CA$
        Where $1 < j < q$
        $erase\ (vcup.vm-> va_t)\ \ from\ CA$
        $translate\ (va_t\ to\ mp)$
        $erase\ (mp)\ \ from\ CA$
        Where $1 < t < x$ and mp is memory page has the $va_t$
}
```

Our solution prevents cache side-channels in time-shared caches (e.g. L1 instruction/data and L2 per core, and L3 if it is used), particularly Prime&Probe and Flush+Reload cache side-channel attacks, which are presented in the background chapter (section 2.3.1). Both of them rely on measuring the time to access an address after a victim VM relinquishes the CPU caches. In the Prime-Probe cache side-channel attack, an attacker will define and monitor number of addresses and cache sets. The attacker will access the targeted addressed to make sure they are in the CPU caches. Then the attacker waits for a predefined time to give a victim a chance to access the monitored addresses. Finally, the attacker will re-access the targeted addresses and calculate the time to access them. If the accessed time less than the time needed to accesses data in last cache level (normally level 3), the monitored addresses were accessed by the victim and vice versa. In addition, Flush+Reload cache side-channel attack follows the same step, yet it requires flushing the addresses from all CPU cache levels and does not monitor CPU cache sets.

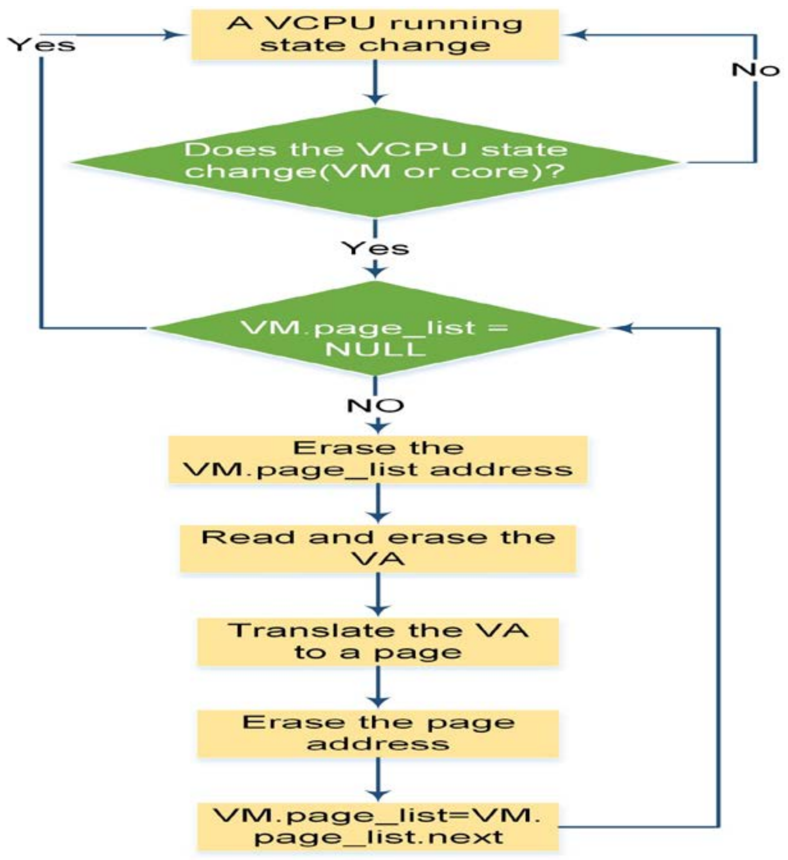
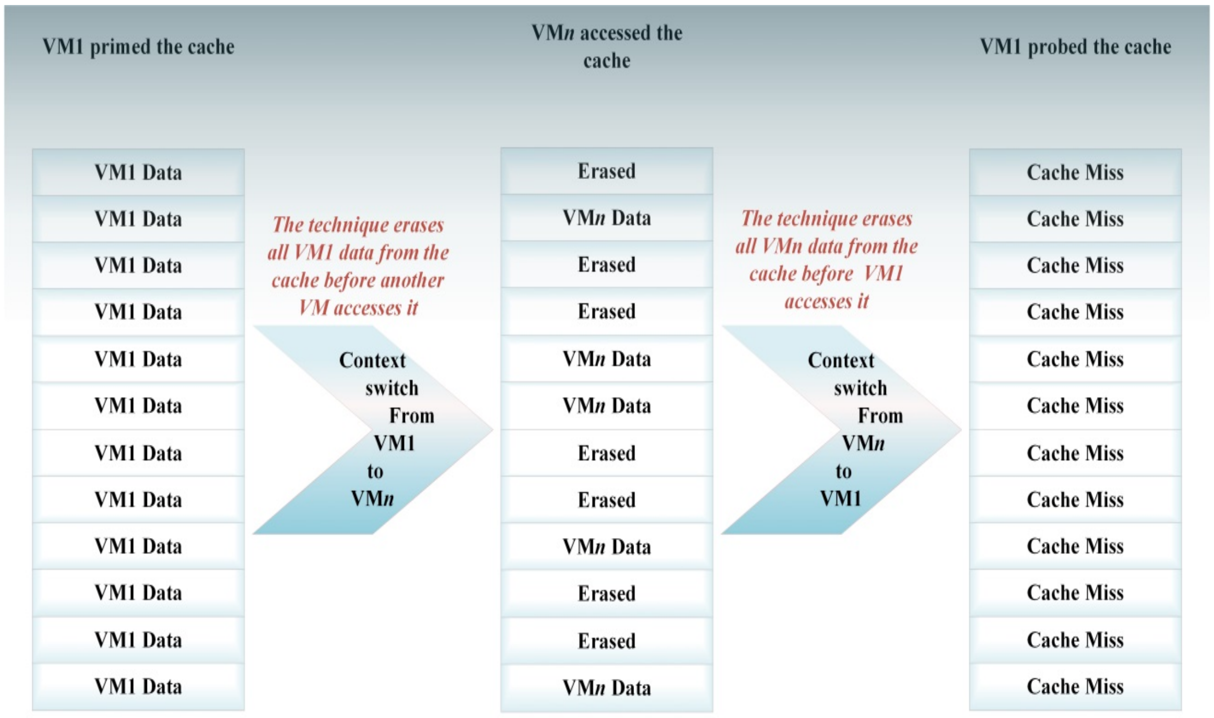

A VCPU running state change
Yes
No
Does the VCPU state change(VM or core)?
Yes
VM.page_list = NULL
NO
Erase the VM.page_list address
Read and erase the VA
Translate the VA to a page
Erase the page address
VM.page_list=VM. page_list.next

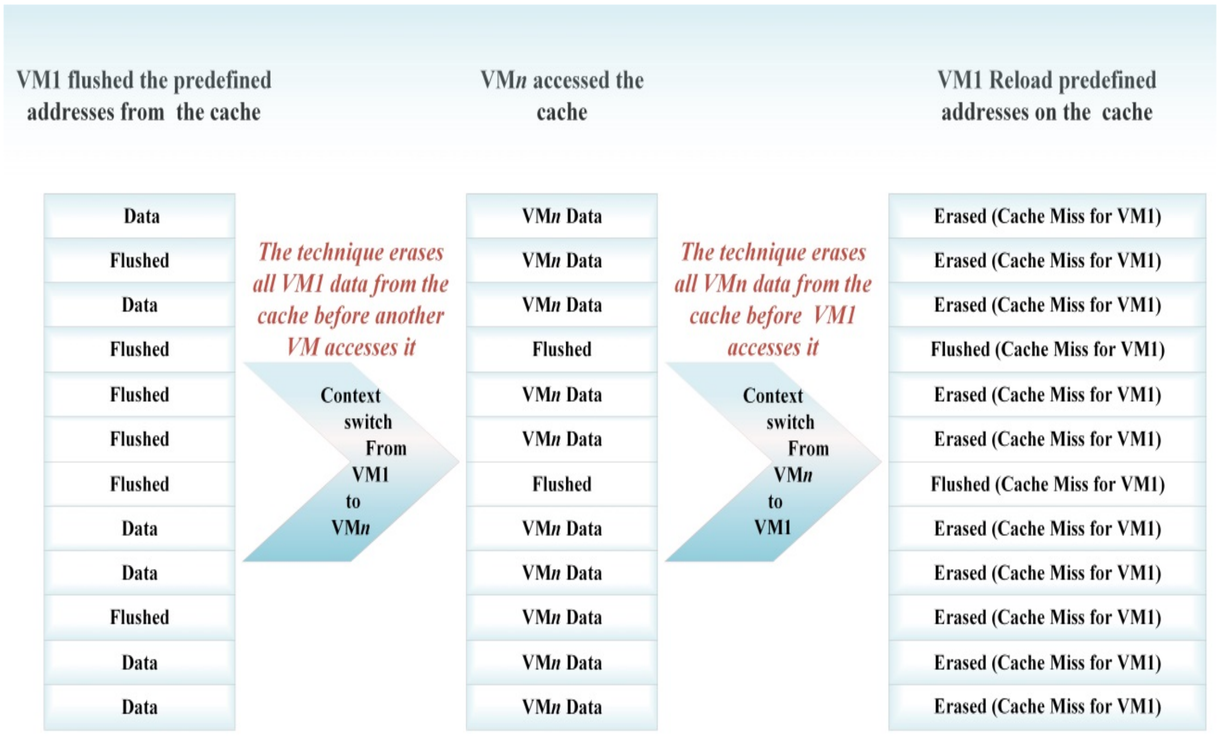
Figure 26: The flow chart of the novel lightweight solution

Thus, erasing any link to what the victim VM has accessed would prevent establishing a cache side-channel attack. The solution starts looking for any changes happening to any VCPU running in the host physical machine. The changes that our solution looks for is any alteration to the core that hosts the VCPU or the VM, which occupies the VCPU as presented in figure 26. When any change is spotted, the solution will erase all virtual addresses used by the VM that occupied the VCPU. It will look practically for the VM's page list and addresses kept on it. Erasing virtual addresses from the CPU cache may raise some concerns about the efficiency and usefulness of the CPU caches. However, as illustrated in details in chapter 7 section 7.1.2, erasing a virtual machine's addresses from the CPU caches by the proposed solution generated at maximum 15,000 CPU cycles (less than 1.5e^{-8} seconds) overload, which is the lowest compared with other solutions proposed for preventing cache side-channel attacks in cloud computing. In the Prime-Probe, the solution destroys any chance an attacker can have to drive any information about previous activities done by a VM as presented in figure 27.



VM1 primed the cache
VMn accessed the cache
VM1 probed the cache
VM1 Data
The technique erases all VM1 data from the cache before another VM accesses it
Context switch From VM1 to VMn
Erased
VMn Data
The technique erases all VMn data from the cache before VM1 accesses it
Context switch From VMn to VM1
Cache Miss

Figure 27: The solution prevents Prime-Probe



VM1 flushed the predefined addresses from the cache
VMn accessed the cache
VM1 Reload predefined addresses on the cache
Data
Flushed
The technique erases all VM1 data from the cache before another VM accesses it
Context switch From VM1 to VMn
VMn Data
The technique erases all VMn data from the cache before VM1 accesses it
Context switch From VMn to VM1
Erased (Cache Miss for VM1)
Flushed (Cache Miss for VM1)

Figure 28: The novel lightweight solution mitigates Flush+Reload attacks

The solution will erase all data from addresses primed by the attacking machine. Thus, any data in a primed address will be removed to return a cache miss to the attacker in the Probe stage. Furthermore, in the Flush+Reload attack, the proposed solution erases a VM's data from all levels of CPU caches instantly after relinquishing the CPU resources (see figure 28). Hence, when the attacker lunches the reload stage, no information can be gained about addresses which were accessed by the victim machine. The proposed solution removes all address accessed by the victim VM without interfering with CPU caches' operation or modifying data. Moreover, it is also injected in the Xen hypervisor and triggered directly after overlapping happens. It will delete all addresses accessed by any VM instantly when it relinquishes the CPU resources. Thus, any attempt to attack by measuring the time of accessing an address would yield cache miss.

5.2.3 Addresses Being Removed From the CPU Caches

Knowing what has to be erased and when is an important aspect for our solution's success. It determines the number of addresses should be removed from the CPU caches. These addresses may need to be translated from any type (logical, linear and physical) to another before being removed from the caches. Moreover, having the exact number of addresses to be erased reduces the total number of CPU cycles needed for deleting addresses from all CPU caches' levels.

Firstly, we will look at what has to be removed. The solution will be applied in Xen hypervisor, which is hosted in an Ubuntu machine. As mentioned in the chapter 6 (section 6.1), there are number of data structures utilised be the Xen hypervisor for each VM in order to keep track of virtual addresses used, pages assigned and accessed. One of these structures is P2M tables, which is a data structure table that has a number of fields to determine number of pages, virtual addresses spaces and etc. Hence, we use this table as a base to what should be removed from the cache. All addresses pointing to pages used by the VM will be removed. Virtual addresses either used by the VM application and processes or utilised to save data structures of the VM, will be sent to the erasing function to be removed from the cache. In addition, in order to reduce the proposed solution's performance overhead, we use a function to translate virtual addresses to a page that hosts them. So, we take every virtual address used by the VM and look for the page that has the virtual address and finally remove its address from the CPU caches.

The kernel module to translate virtual addresses

```
static inline struct page_info *__virt_to_page(const void *v)
{
   unsigned long va = (unsigned long)v;

   ASSERT(va >= XEN_VIRT_START);
```

```
        ASSERT(va < DIRECTMAP_VIRT_END);
        if ( va < XEN_VIRT_END )
            va += DIRECTMAP_VIRT_START - XEN_VIRT_START +
    xen_phys_start;
        else
            ASSERT(va >= DIRECTMAP_VIRT_START);
        return frame_table + ((va - DIRECTMAP_VIRT_START) >>
    PAGE_SHIFT);
    }
```

The second question is when the proposed solution has to be called. Firstly, whenever overlapping access to the CPU caches, the erasing technique should be called. Cloud computing tenants share pools of configurable computing resources such as virtual and physical CPUs; yet sharing of these resources causes overlapping access managed by scheduling techniques and load balancing mechanisms. The overlapping access has the direct cause of cache side-channel attacks. We come across four different overlaps:

1. Number of VM overlapping access to one VCPU

   Here the VMs are overlapping to access one available VCPU. So, all the VMs have time-shared access to the VCPU due to binding all VMs to a VCPU or the host machine has just one core as presented in figure 29. In addition, The Xen hypervisor has a command which can be used to bind VMs to one VCPU.

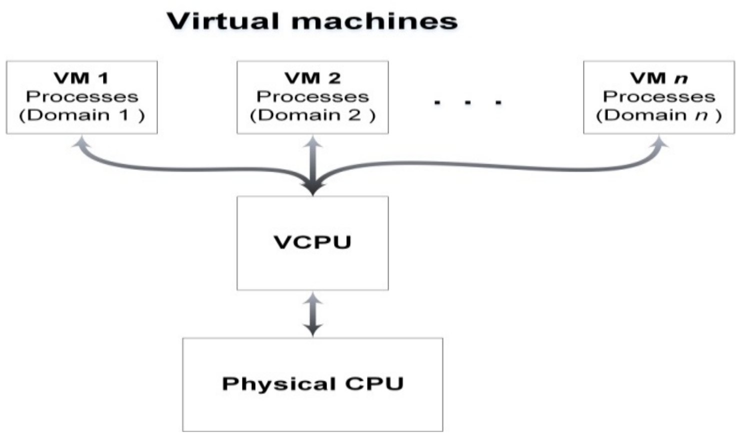

Virtual machines
VM 1 Processes (Domain 1)
VM 2 Processes (Domain 2)
. . .
VM *n* Processes (Domain *n*)
VCPU
Physical CPU

Figure 29: Virtual machines overlapping access to a VCPU

2. Number of VM overlapping access to a number of VCPUs

   In the Xen hypervisor, when there is more than one VCPU, each VM will be bended to a VCPU till the number of VMs is greater than the number of VCPUs. In that case, VMs will have time-shared access to VCPUs as illustrated in figure 30. In the figure, the total numbers of VMs and VCPUs are *n* and *m* respectively. Thus, the overlapping happens only when $m < n$, which should be the common case in the Xen.

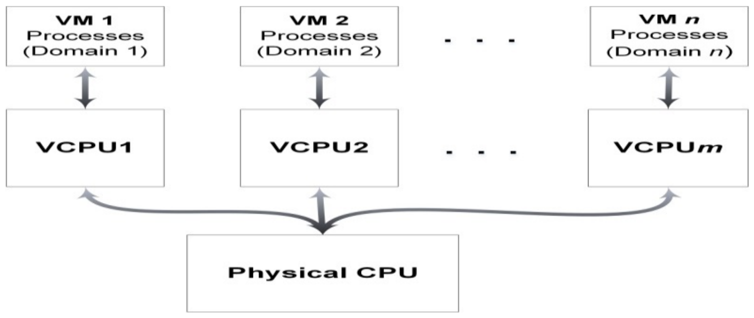


Figure 30: Virtual machines overlapping access to VCPUs

3. Number of VCPUs overlapping access to a CPU core

   When VCPUs have one CPU core, they will utilise it in time-shared manner. Although, in our experiments on Xen we have not come across this case, in other virtual environments that use Symmetric Multi-processing (SMP) it might happen (see figure 30).

4. Number of VCPUs migrates from one CPU core to another

   In the Symmetric Multi-processing (SMP) settings, VMs will be floated from one CPU core to another as shown demonstrated in figure 31. Hence, a VM assigned to a VCPU can gain information about operations and addresses accessed by another VM [71].

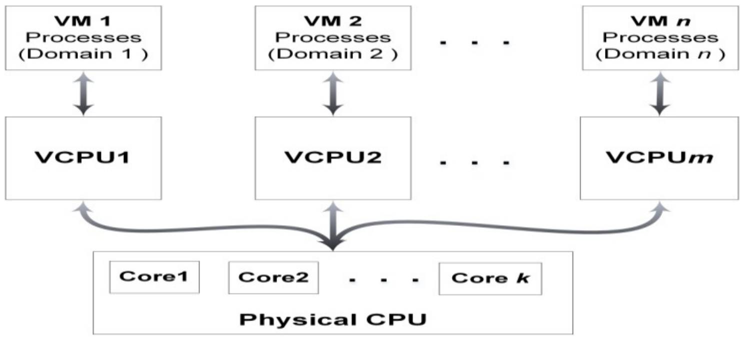


Figure 31: VCPUs migrates from a CPU core to another

5.3 A New Infrastructure Solution to Detect Cache Side-Channel Attacks in Cloud Computing

The reported attacks scenarios (Prime & Probe and Flush+Reload) in cache side-channel attacks category have specific characteristics such as evicting a number of CPU cache lines to launch an attack. In order for a cache line to be evicted, there are two ways to do it either by using a flushing command or reading data from main memory to be written in that line. Either one of the mentioned ways will cause a cache miss to the CPU. Thus, most of the reported attacks will cause a considerable number of cache misses to the CPU in order to evict the data on the cache. These cache misses can be counted and compared with normal cache misses caused in a normal code execution. However, it is a difficult and crucial task.

Although, the number of cache misses out of the total number of execution times granted to the VM is playing a big part to determine cache side-channel attacks, it cannot give a precise decision without examining the cache misses sequences. Launching a cache side-channel attacks such as Prima and Probe will write a big chunk of data (nearly the size of one of CPU caches) to a CPU cache. That will cause a big number of cache misses in a sequence manner. Thus, measuring the number of cache misses and testing whether they are sequences or not, can definitely detect cache side-channel attacks.

5.3.1 Our Goals

In order for the time-shared CPU caches attacks to succeed, they need to evict all the data kept in the cache by reading big pieces of data to the cache and that will induce an unusual number of cache misses to the CPU. Thus, we are measuring the number of cache misses caused in each fetch cycle and the time it is taking to decide whether it is a cache side-channel attack or not.

The proposed detection solution aims to satisfy the following goals:

1. The solution should measure a number of cache misses caused and when they happen by a VM in its execution time without using any attached software or application.
2. The solution should be accurate in terms of distinguishing between normal cache misses and others caused by cache side-channel attacks. It also should report any cache side-channel attacks instantly.
3. The solution should induce neglected performance overhead.
4. The hypervisor should not require any major modifications.

5. All the tenants' operating systems should be completely free from any modifications or changing in their applications or libraries.

5.3.2 The Proposed Detection Infrastructure Solution

The proposed solution is a detecting strategy that mitigates and prevents cache side-channel attacks in cloud computing. It focuses on the infrastructure used to host cloud computing tenants by counting cache misses caused by a virtual machine. The number of cache misses will be counted and added to the sequence of cache misses in order to detect cache side-channel attack. The design, implementation (see section 6.2 in chapter 6) and evaluation (look at section 7.1 in chapter 7) of the solution illustrated that our solution effectively detects attacks on time-shared caches.

The solution attempts to detect cache side-channels in time-shared caches (e.g. L1 instruction/data and L2 per core, and L3 if it is used), particularly Prime-Probe and Flush+Reload attacks. Both of them rely on measuring the time to access an address after a victim VM relinquishes the CPU caches. Thus, they erase data kept in the cache by replacing it with their own data and that will generate a considerable number of cache misses. Our solution implemented in the host's kernel to measure time taken in every fetch cycle. If the time is greater than a predefined threshold, it is definitely a CPU cache miss and it will be added to a data structure used to keep track of every VM's behaviour. Moreover, the time when a cache miss happens is kept in another field in the data structure in order to compute the variation times between cache misses caused by each VM. A standard deviation will be calculated to the variation times. If the standard deviation is lower than a threshold, the CPU cache misses recorded are caused by a cache side-channel attack lunched by a VM.

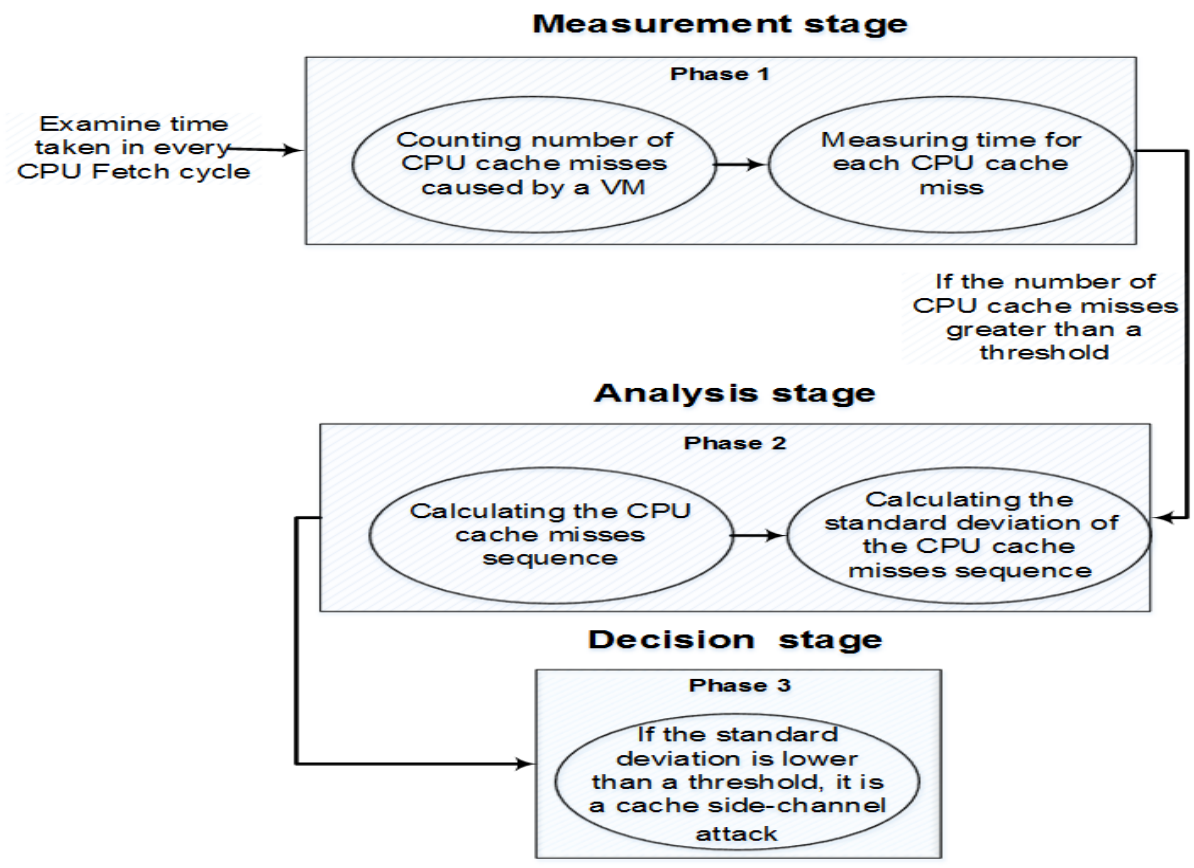

Measurement stage
Phase 1
Examine time taken in every CPU Fetch cycle
Counting number of CPU cache misses caused by a VM
Measuring time for each CPU cache miss
If the number of CPU cache misses greater than a threshold
Analysis stage
Phase 2
Calculating the CPU cache misses sequence
Calculating the standard deviation of the CPU cache misses sequence
Decision stage
Phase 3
If the standard deviation is lower than a threshold, it is a cache side-channel attack

Figure 32: The novel infrastructure solution

As shown in figure 32, our solution has three stages:

1. The first stage is the measurement stage, which has two steps: measuring the number of CPU cache misses caused by a VM and measuring time for each CPU cache miss recorded. Firstly we need to define when a CPU cache access is a miss or hit. We use a threshold reflects the maximum time taken for accessing data on caches (from level 1 up to level 3). Thus, any CPU access time (fetch cycle) greater than the threshold, is a CPU cache miss. When a cache miss is recorded, it will be accounted to the (VCPU) that occupied by the running VM. Moreover, the start and end time of the Fetch cycle that causes the cache miss, will be saved to be used in the analysis stage if the number of cache misses are greater than the threshold. This stage will continue while the VCPU and the VM using it, is still occupying the CPU resources. Before the overlapping between VCPUs happens, the total number of CPU cache misses caused by the VCPU will be compared with the total cache misses threshold. If they are greater than threshold, they are most likely to be a cache side-channel attack launched by the VM and the analysis stage will start. Otherwise, they are normal CPU cache misses and there is no need for the analysis stage.

2. The second stage is the analysis stage. It has two functions:

- Calculating the CPU cache misses sequence

  This step is a very important step in the analysis stage. It calculates the time between the end of a fetch cycle that causes a CPU cache miss, and the start of the following fetch cycle, which induces a new CPU cache miss as illustrated in figure 33. Where *CMT* means a cache miss time, *start* is the beginning time of a cache misses, *end* is the ending time of a CPU cache miss, *S* is the difference between an end of a CPU cache miss and a start of another CPU cache miss followed it, *n* is the total number for *CMT* recorded for a VM during its execution time and *m* is the total number of *S*.

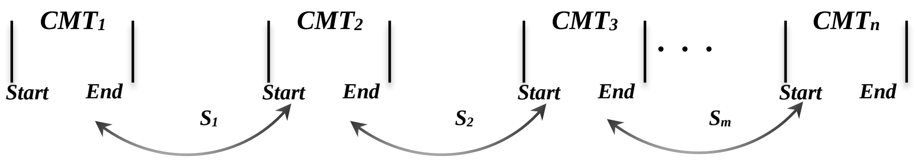


Figure 33: A CPU cache misses sequence

  A CPU Cache Misses Sequence *(CCMS)* is a sequence comprised of several *CMTs* ordered by time. So, *CCMS = {CMT1, CMT2, …, CMTn}*. *S* is defined in (1).

$$s_k = start_{CMTi+1} - end_{CMTi} \qquad (1)$$

  $\forall\ CMT_i and\ CMT_{i+1}$ Where $0 < i < n\ and\ 1 < k \leq m$

- Calculating the standard deviation of the CPU cache misses sequence.

  In order to determine and measure the variation between occur time of CPU cache misses, we calculate the standard deviation of the CPU cache misses sequence. It can help testing the regularity of CPU cache misses.

  For a set *S*= {s_1, …, s_m}, the standard deviation (σ) is calculated as defined in (2).

$$\sigma = \sqrt{\frac{\sum_{1 \le k \le m}(s_k - \bar{s})^2}{m}} \qquad (2)$$

Where $\bar{s}$ is the mean of set S and m is the total number of S entities

3. The final stage is the decision stage

   It takes the output of analysis decision, which is the standard deviation. It is compared with a threshold (pre-calculated standard deviation) that reflects regularity of CPU cache misses in a real attack. If standard deviation is lower than the threshold, the VM that has CPU cache misses is launching a cache side-channel attack. Otherwise, no attack has been launched even when there are a considerable number of cache misses. We will fully explain these cases in the evaluation chapter (chapter 7).

5.4 Summary

This chapter gave a detailed description of a new Prime and Probe attack and the two new solutions that are proposed for preventing and detecting cache side-channel attacks in cloud computing. The new Prime and Probe cache side-channel attack primes physical addresses translated form virtual addresses used by a virtual machine. Then, time is measured to access these addresses and it will be varied according to where the data is located. If it is in the CPU cache, the time will be less than in the main memory. The prevention solution is a new lightweight solution that can prevent cache side-channel attacks with neglected overload. It erases a virtual machine's data from all CPU cache levels when the virtual machine relinquishes CPU resources in order to prevent any chance of launching a cache side-channel attack. The erasing algorithm will not delete or remove any information does not belong to the targeted virtual machine. Furthermore, the proposed new infrastructure detection solution is free from any modifications that have to be done in either cloud tenants' machines or cloud service provider's operating systems. The solution does not rely on any software or application to detect any abnormal behaviour in the CPU caches. It uses a new way to measure the time taken by every fetch cycle executed by a virtual machine. The next chapter will show in details how the proposed prevention and detection solutions will be implemented in the Xen hypervisor and justify the reasons of using the Xen.

CHAPTER 6 The Implementation of the Proposed Prevention and Detection Solutions to Cache Side-Channel Attacks

In order to implement the proposed prevention and detection solutions to cache side channel attacks in cloud computing, a cloud based testbed has been developed which include 30 virtual machines and 8 VCPUs. In this chapter we provide a detailed description of how both solutions are implemented in the kernel of Xen hypervisor. This chapter (section 6.1.1) clarifies the reasons behind using the Xen hypervisor for implementing the solutions. It also details the testbed used to conduct the evaluation experiments. The algorithms developed for implementing the novel lightweight solution and steps followed to write and append the solutions' codes in the Xen's kernel are illustrated in section 6.2. Moreover, full details of the steps used to implement the new detection solution are presented in section 6.3.

6.1 The Testbed

6.1.1 Xen Hypervisor

The Xen hypervisor resides between the hardware and the operating system in order to grant virtual averment [105]. It is an open-source bare-metal or type-1 hypervisor implementation. It is evolved by a world-wide community of researchers, employees of companies and individuals. It is being used on contemporary hardware architectures of public clouds including Amazon's EC2 [106] and Rackspace [107]. Moreover, the Xen hypervisor is also used to separate operating systems into several isolated components with various privilege levels such as Qubes [108]. Xen makes it possible to run multiple guest operating systems in a single host domain. The Xen hypervisor hosts virtual machines and runs directly on the physical hardware of the host machine. The virtual machines can run different operating systems (Linux, NetWare and MS Windows).

The Xen hypervisor runs on the hardware and is responsible for handling CPU, memory, and interrupts. A number of virtual machines are running on top of the hypervisor. The running virtual machines are called domains (*DomUs*) or guests. The host domain is named domain 0 (*Dom0)* and contains the drivers (e. g. network backend driver) for all the devices in the system. Domain 0 controls virtual machine creation, destruction, and configuration. In addition, software isolation mechanisms are used in the Xen to separate domains and prevent leakage of information from one domain to another. However, side-channel attacks have been used to bypass and break the software isolation mechanisms [71].

1. Type of VMS

   Two different kinds of modes are supported in the Xen hypervisor: Full or Hardware assisted Virtualization (HVM) mode and Para-virtualization (PV) mode. They can be used and run on a single hypervisor at the same time.

   - Para-virtualization (PV)

     It is a light-weight virtualization technique. Although, PV guests do not require virtualization extensions from the host CPU, they require a PV-enabled kernel and PV drivers in order to make them aware of the hypervisor and can work without emulation or virtual emulated hardware. Moreover, PV can work with most Linux distributions such as Ubuntu. It also works with other kernels (e.g. NetBSD, FreeBSD and OpenSolaris)[107].

   - Hardware-assisted virtualization (HVM) or Full Virtualization

     It employs virtualization extensions from the host CPU to virtualize guests. Thus, HVM requires AMD-V or Intel VT hardware extensions. Qemu is used in the Xen hypervisor to emulate PC hardware, including BIOS, VGA graphic adapter, IDE disk controller, USB controller etc. In addition, there is no need for any kernel support to (HVM) guests. However, the required emulation in HVM guests makes them slower than PV guests [107].

2. Memory management in Xen

   The para-virtualization of the memory management unit (MMU) was one of the original innovations of the Xen hypervisor. It enables the Xen hypervisor to model the platform on which it runs (e.g. x86) [105]. The using of para-virtualization facilitates guest operating systems with efficient and fast virtualization. Moreover, memory management in Xen depends on the platform used to host the domains such as X86 or ARM and the type of domains either Para-virtualization (PV) or Hardware-assisted virtualization (HVM). Para-virtualization (PV) utilises direct paging and HVM uses shadow paging.

In our experiments, we have used PV domains. Thus, we will only talk about direct paging. At the beginning there are three different types of addresses used in the Xen hypervisor [105]:

- Virtual memory addresses which are employed for user-space applications.
- Pseudo-physical memory addresses which appear as virtual memory to Xen and as physical memory to parts of the kernel that are not virtualization-aware.
- Machine addresses which point to physical memory locations in the host system.

In the Xen hypervisor, there are two different tables used alongside the ordinary page table utilised for translating virtual address to machine address. These tables are Physical to Machine (P2M) mapping table and Machine to Physical (M2P) mapping table. They are a simple array of frame numbers, indexed by physical or machine frames and looking up the other. In addition, the P2M mapping table is managed by the guest operating system. The table size has to be the same size as the guest's pseudo-physical address space size. On the other hand, the M2P mapping table is sized according to the total amount of RAM in the host. Thus, it is controlled by the host operating system.

3. Xen credit scheduler

The credit scheduler is a relatively fair share CPU scheduler, which balances and schedules accessing to the CPU resources [109]. There are two files in the Xen package which can be used to control sharing the CPU resources (credit scheduler and credit scheduler2), yet the credit scheduler is the default scheduler in the Xen 4.04. There are number of important parameters employed to control execution time assigned to each Virtual CPU (VCPU):

- Weight

It is used to determine the CPU time can one VM has compared with another VM. For instance, a VM with weight 512 will get twice as much processing time as a VM which has weight 256.

- Cap

  It expresses the percentage a VM can have from the total processing capacity of a CPU. For example, when a cap of a VM equals 100, the VM will have a full physical CPU.

- Timeslice

  The Timeslice illustrates the default amount of time allocated to a VCPU for execution on a CPU core. Its value is measured in milliseconds. When the Timeslice has higher values it will give each VM a big time slice in the credit scheduler and vice versa. Moreover, assigning lower values to it increases context switching rates between domains [109].

- Ratelimit

  It demonstrates a minimum amount of time that a VCPU must run before it might be interrupted by another VCPU that has higher priority. It was added in Xen 4.2. The default value is 1000μs [109].

The credit scheduler parameters' values can be viewed and modified as shown in the followed table (table 4) by using this command

The command	Its description
# xl sched-credit	Displaying the credit scheduler's parameters current values
# xl sched-credit -d domain -w weight	Modifying the weight parameter for a domain
# xl sched-credit -d domain -c cap	Modifying the cap parameter for a domain
# xl sched-credit –r timeslice	Modifying the timeslice parameter for a domain
# xl sched-credit –t ratelimit	Modifying the ratelimit parameter for a domain

Table 6.1: The credit scheduler commands

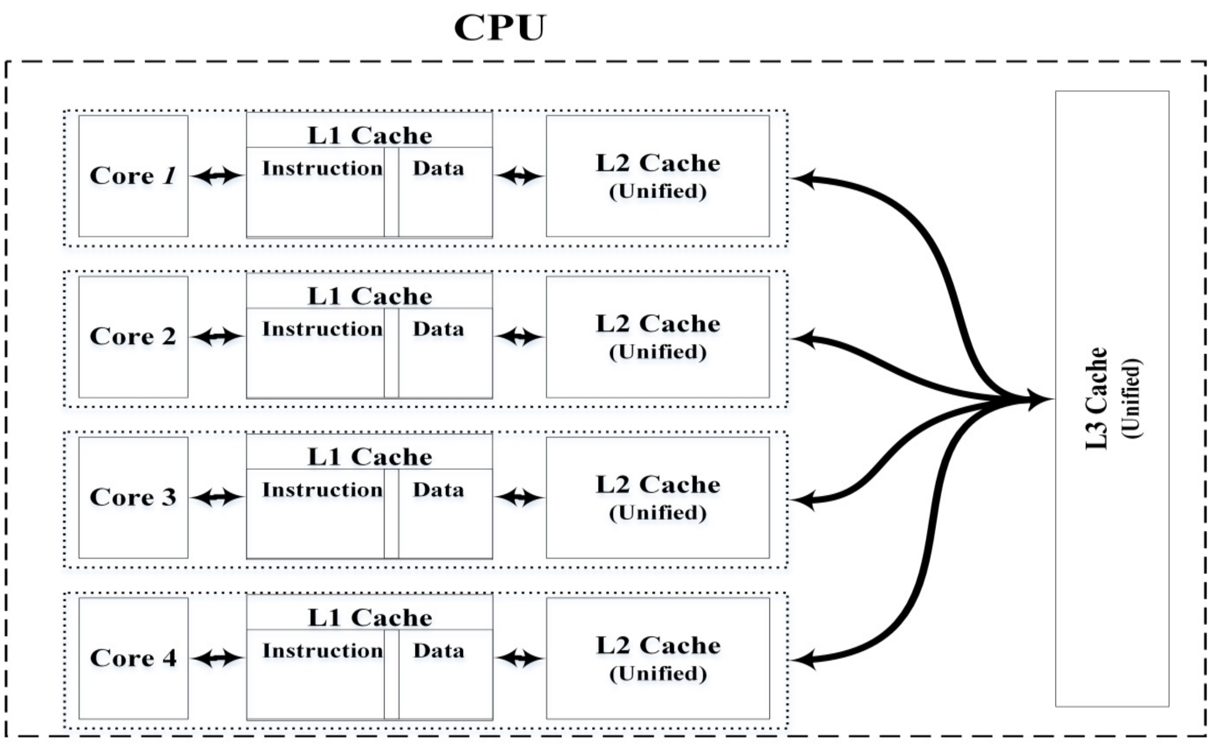
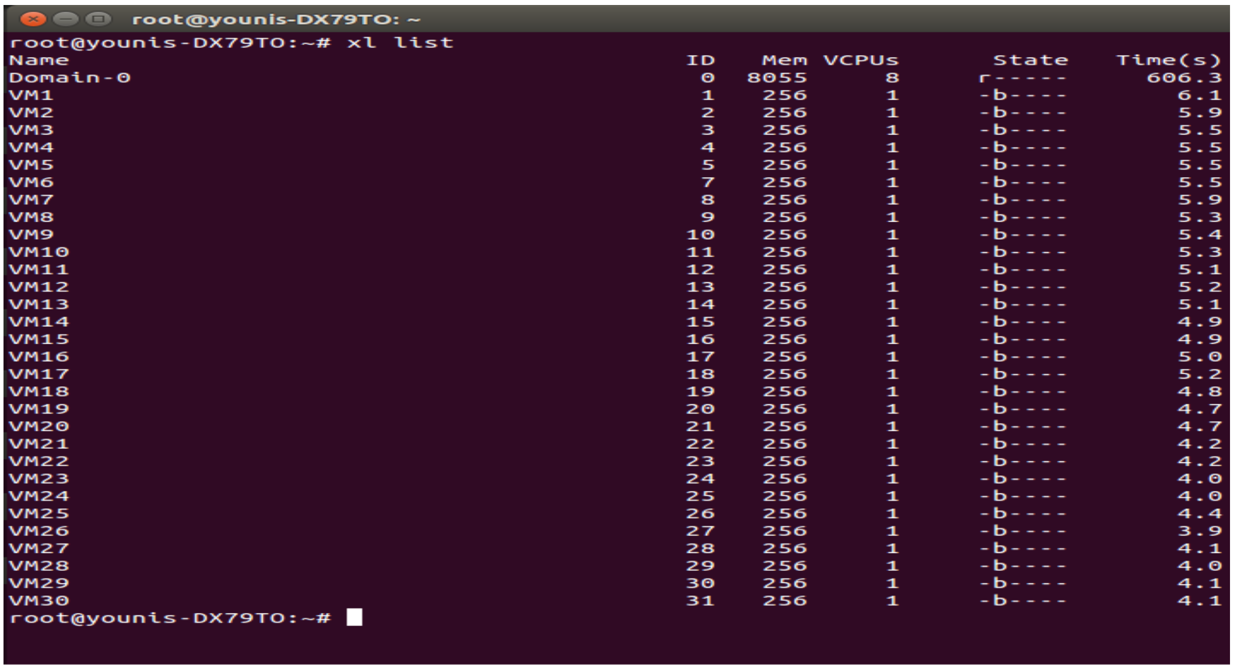


Figure 34: The CPU's physical cores and cache levels

6.1.2 The Host Server

We conducted performance and security evaluations for both solutions on a host machine that had an *x86-64* architecture and Ubuntu desktop 14.04, which was the host operating system. The host machine (server) used in our experiments is a comparable to cloud environment servers. In addition, we use the aforementioned points presented in our goals sections 5.2.1 and 5.3.1 in chapter 5 as research questions to evaluate the solutions. As presented in figure 34, the server was equipped with quad-core (2 logical cores per physical) Intel i7-3820 processor with an operating frequency of 3.60GHz. The server had three levels of caches. Each physical core had L1 and L2, yet all cores shared L3. Furthermore, L1 data and instruction caches were 32KB in size and 8-way set-associative. The unified L2 was 256KB in size and 8-way set-associative. The shared unified L3 was 10MB and 20-way set-associative. Moreover, all three levels had 64-byte cache lines and 64B cache lines. The server that represents the management domain in Xen (*Dom0*) runs Ubuntu 14.04. The hypervisor used in our experiments was Xen 4.4 hypervisor. In addition, we used various numbers of virtual machines and each VM had one VCPU. RAM size assigned to each VM was varied due to the number of VMs running in the server at the same time and the size of the RAM was 16GB. However, the minimum amount of RAM assigned to a VM was 256 KB when the number of VMs was 30 as shown in figure 35.



```
root@younis-DX79TO: ~
root@younis-DX79TO:~# xl list
Name                                        ID   Mem VCPUs      State   Time(s)
Domain-0                                     0  8055     8     r-----     606.3
VM1                                          1   256     1     -b----       6.1
VM2                                          2   256     1     -b----       5.9
VM3                                          3   256     1     -b----       5.5
VM4                                          4   256     1     -b----       5.5
VM5                                          5   256     1     -b----       5.5
VM6                                          7   256     1     -b----       5.5
VM7                                          8   256     1     -b----       5.9
VM8                                          9   256     1     -b----       5.3
VM9                                         10   256     1     -b----       5.4
VM10                                        11   256     1     -b----       5.3
VM11                                        12   256     1     -b----       5.1
VM12                                        13   256     1     -b----       5.2
VM13                                        14   256     1     -b----       5.1
VM14                                        15   256     1     -b----       4.9
VM15                                        16   256     1     -b----       4.9
VM16                                        17   256     1     -b----       5.0
VM17                                        18   256     1     -b----       5.2
VM18                                        19   256     1     -b----       4.8
VM19                                        20   256     1     -b----       4.7
VM20                                        21   256     1     -b----       4.7
VM21                                        22   256     1     -b----       4.2
VM22                                        23   256     1     -b----       4.2
VM23                                        24   256     1     -b----       4.0
VM24                                        25   256     1     -b----       4.0
VM25                                        26   256     1     -b----       4.4
VM26                                        27   256     1     -b----       3.9
VM27                                        28   256     1     -b----       4.1
VM28                                        29   256     1     -b----       4.0
VM29                                        30   256     1     -b----       4.1
VM30                                        31   256     1     -b----       4.1
root@younis-DX79TO:~#
```

Figure 35: The number of VM used in the experiments

6.2 The Implementation of the Lightweight Prevention Solution

The proposed solution has been implemented using the open source Xen Hypervisor version 4.4. Specifically, we implemented and wrote the proposed solution in three different places in the Xen's kernel in order to deal with all types of overlapping access that are presented at section 5.2.3 in section 5 . It is worth mentioning that the Xen hypervisor creates a scheduling unit called Virtual CPU (VCPU), which is assigned to one of the physical CPU cores or a time slot on the physical CPU. Thus, we focused on VCPUs' behaviours more than VMs as it's the medium component between a VM and a CPU core. Furthermore, most of the overlapping happens between VCPUs and CPU cores. The results of our implementation are a tracing algorithm and an erasing algorithm as shown in figure 36.

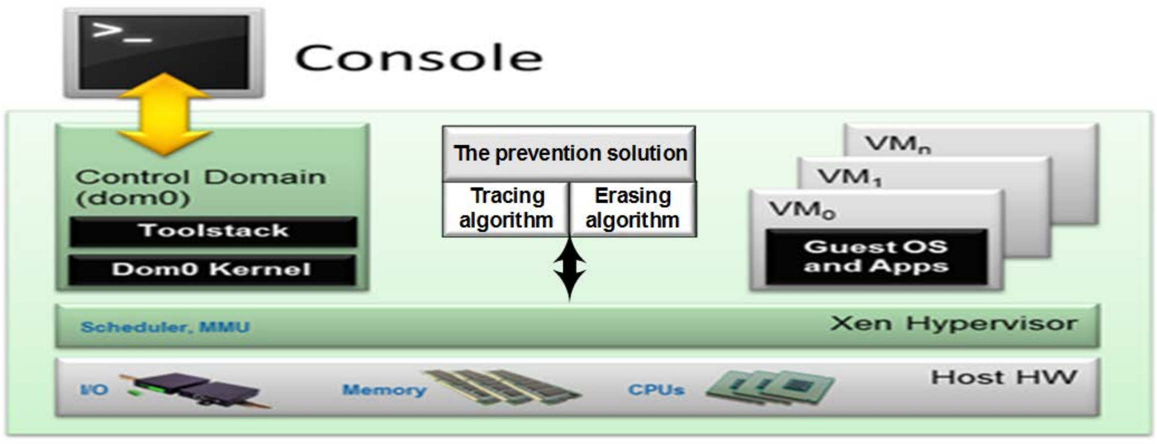


Figure 36: Implementing the solution's algorithms on the Xen hypervisor

6.2.1 Tracing Algorithm

The tracing algorithm aims to give the precise decision when a VM's data should be searched and removed from the CPU caches. It starts with tracing the run state of the running VCPU. If any changes happen to the VCPU values such as migrating it to another core or changes to the VM using the VCPU, the trace algorithm will examine the changes whether they affect the domain occupying the VCPU and the core utilised or not. If any changes happen to the domain id or the core id in the VCPU date structure, the erasing algorithm will be called and passed to the VCPU data structures. In addition, the tracing algorithm operates in the kernel of the Xen hypervisor as long as it runs.

Tracing algorithm (Tracing VCPUs run state)

```
Trace_VCPU_Runstate(VCPU)
{
    If (the run state of (VCPU) is changed)
      If (VCPU->domain is changed) OR (VCPU->CPU.core is changed)
      Eras_cache_lines(VCPU->domain.data);
      Else  Return;
    Return;
} End
```

6.2.2 Erasing Algorithm

It mainly focuses on identifying data and how data should be removed.

Erasing algorithm (Erase a VM's data from all CPU cache levels)

```
Erasing_cache_lines(VCPU)
{
    While (VCPU->domain.page_list!=NULL)
    {
      Erase(Read_address(domain.page_list));
      Erase(domain.VA);
      Erase(translate_VA_to_page(domain.VA));
      domain.page_list= domain.page_list.next;
     }
    Return;
}End
```

In order to deal with the data leakage caused by different types of overlapping access to VCPU or CPU's cores, at least one of the aforementioned algorithms was coded in three different places in the Xen's kernel. Firstly, the erasing algorithm was added to Xen credit scheduler file, which balances and schedules accessing the CPU resources. Thus, the erasing algorithm was coded in a function that removes a VCPU from the Xen run-queue when its execution time ends in order to prevent deriving data from cache lines were accessed by the VCPU. The second place is the Xen scheduler file. The tracing and erasing algorithms are coded in the VCPU run-state function to look for any changes in a VCPU run state. For example, the VCPU migrates to another CPU's core. Hence, the solution's algorithm will prevent launching a cache side-channel attack. Finally, the erasing algorithm was added in the Xen domain file, which manages context switching between domains. In addition, figure 37 shows the kernel's structure of the Xen hypervisor. Words in bold illustrate where the solution's algorithms were coded.

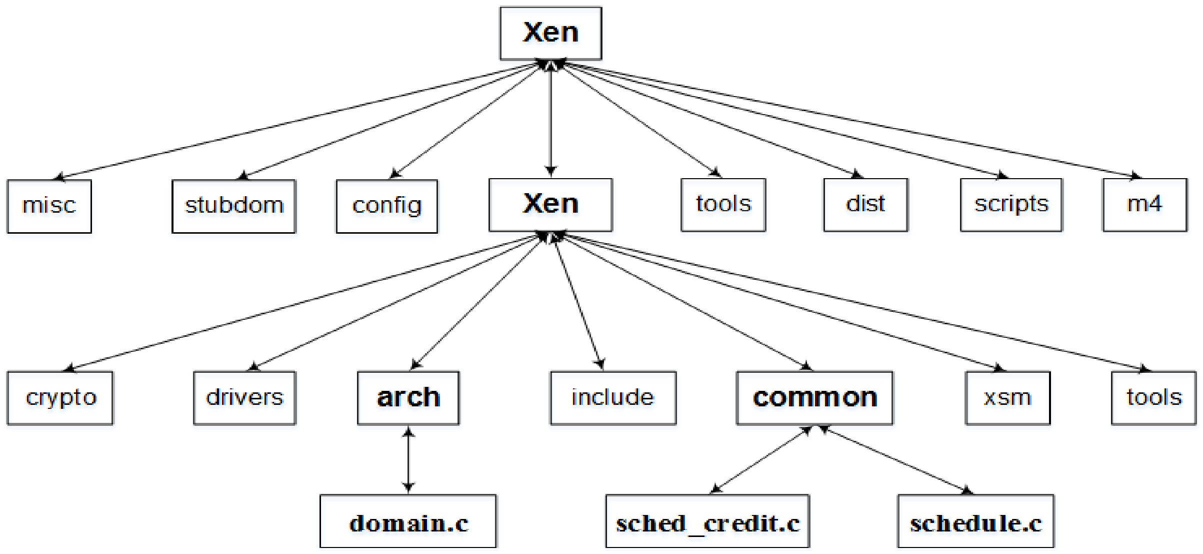


Figure 37: The kernel's structure of the Xen hypervisor

6.2.3 Clflush

The Clflush instruction invalidates the cache line that contains the linear address specified with the source operand from all levels of the processor cache hierarchy (data and instruction). The invalidation is broadcast throughout the cache coherence domain. If, at any level of the cache hierarchy, the line is inconsistent with memory (dirty) it is written to memory before invalidation [110].

There are three different addresses supported in the Intel processors:

1. Virtual address (logical address)

   In operating systems, a range of virtual addresses for each process is used for security and isolation proposes, since the 8086 architecture forces the programs to be divided in segments. Thus, these virtual addresses tell which segment is used and the address in segment holding the value of an operand or instruction.

2. Linear address

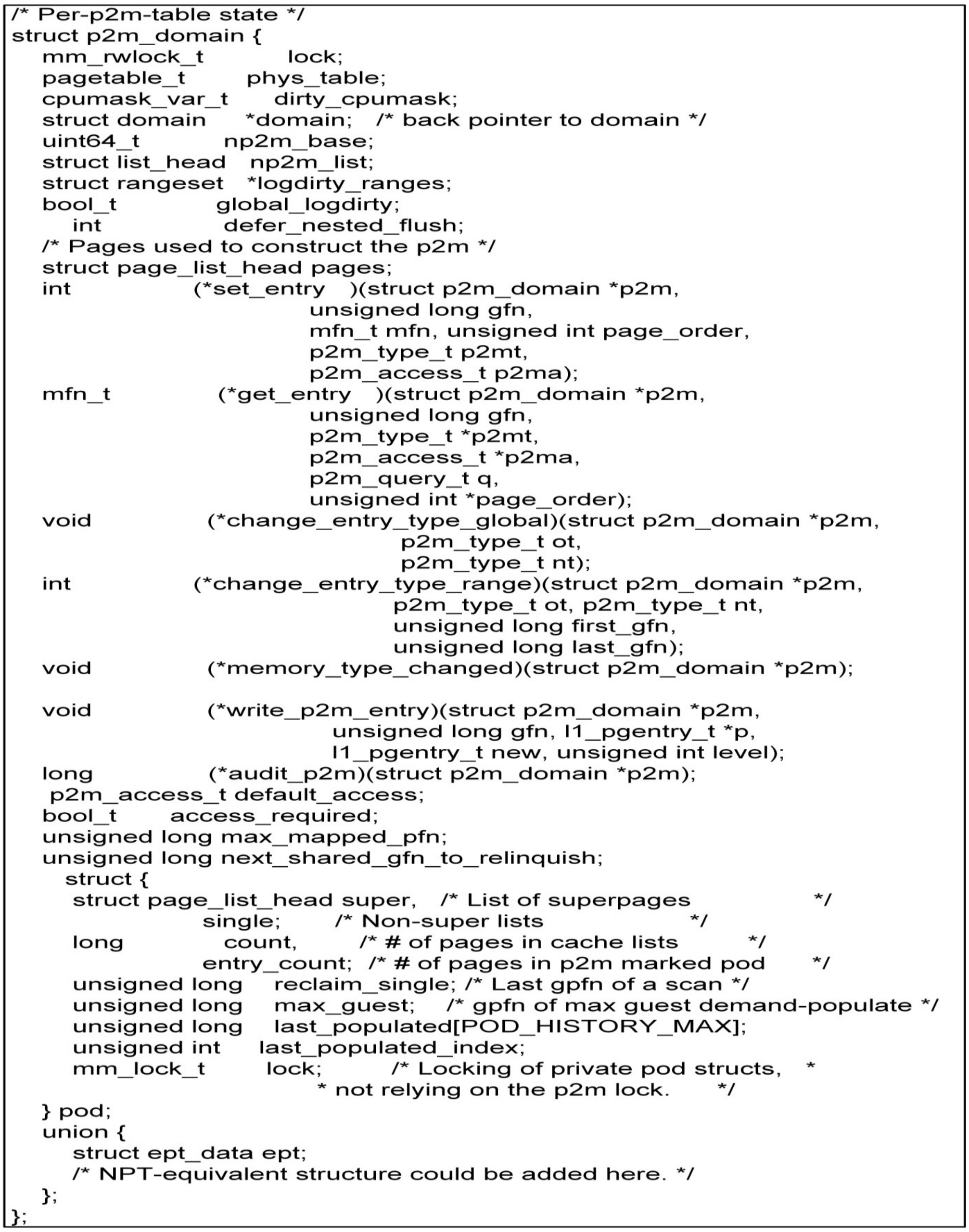
   It is an address derived from virtual addresses by segment translation and segment descriptor tables. These addresses are a part of the memory segment of a program. They are used to get physical addresses by using page directory and page tables.

3. Physical address

   They are calculated from linear addresses through paging and correspond to addresses in RAM. It is the value that the processor places on its address lines in order to access a value in chip-based memory. If the paging is not enabled in the operating system, linear addresses are equal to physical addresses.

As described in the Intel document, the Clflush teaks and flushes a linear address attended to be flushed from all CPU cache levels. However, by using experiments, we prove flushing a linear address will remove the physical addresses that are mapped to it. In the L1 all aliased virtual to physical translations map to the same cache set on Intel processors. Thus, evicting the linear address removes the physical address. Moreover, in the rest of cache levels (L2 and L3), all addresses used by the operating system are physical addresses. Hence, the physical address that gets broadcast globally for eviction.

In Linux Ubuntu, segmentation and linear addresses are not supported. They are only used for permission control and the kernel in Ubuntu configures each segment's offset value to zero. Thus, linear addresses are not used in a kernel, and the kernel directly uses virtual address on paging units. After getting the virtual address, the MMU paging unit uses CR3 register to get the base of paging table to generate physical address.



```
/* Per-p2m-table state */
struct p2m_domain {
    mm_rwlock_t          lock;
    pagetable_t          phys_table;
    cpumask_var_t        dirty_cpumask;
    struct domain        *domain;   /* back pointer to domain */
    uint64_t             np2m_base;
    struct list_head     np2m_list;
    struct rangeset      *logdirty_ranges;
    bool_t               global_logdirty;
        int              defer_nested_flush;
    /* Pages used to construct the p2m */
    struct page_list_head pages;
    int                  (*set_entry   )(struct p2m_domain *p2m,
                                   unsigned long gfn,
                                   mfn_t mfn, unsigned int page_order,
                                   p2m_type_t p2mt,
                                   p2m_access_t p2ma);
    mfn_t                (*get_entry   )(struct p2m_domain *p2m,
                                   unsigned long gfn,
                                   p2m_type_t *p2mt,
                                   p2m_access_t *p2ma,
                                   p2m_query_t q,
                                   unsigned int *page_order);
    void                 (*change_entry_type_global)(struct p2m_domain *p2m,
                                          p2m_type_t ot,
                                          p2m_type_t nt);
    int                  (*change_entry_type_range)(struct p2m_domain *p2m,
                                          p2m_type_t ot, p2m_type_t nt,
                                          unsigned long first_gfn,
                                          unsigned long last_gfn);
    void                 (*memory_type_changed)(struct p2m_domain *p2m);

    void                 (*write_p2m_entry)(struct p2m_domain *p2m,
                                    unsigned long gfn, l1_pgentry_t *p,
                                    l1_pgentry_t new, unsigned int level);
    long                 (*audit_p2m)(struct p2m_domain *p2m);
     p2m_access_t default_access;
    bool_t        access_required;
    unsigned long max_mapped_pfn;
    unsigned long next_shared_gfn_to_relinquish;
       struct {
        struct page_list_head super,   /* List of superpages                */
                         single;       /* Non-super lists                   */
        long               count,      /* # of pages in cache lists         */
                         entry_count;  /* # of pages in p2m marked pod      */
        unsigned long    reclaim_single; /* Last gpfn of a scan */
        unsigned long    max_guest;    /* gpfn of max guest demand-populate */
        unsigned long    last_populated[POD_HISTORY_MAX];
        unsigned int     last_populated_index;
        mm_lock_t        lock;         /* Locking of private pod structs,   *
                                        * not relying on the p2m lock.      */
    } pod;
    union {
        struct ept_data ept;
        /* NPT-equivalent structure could be added here. */
    };
};
```

Figure 38: Data structure of P2M table

6.2.4 A VCPU's Memory Addresses to Be Erased

As described in the memory structure of the Xen hypervisor, every VM has a table called Physical to Machine (P2M) mapping table. This table has full details about the number of pages, the structure of the page list and mapping from physical to machine addresses, etc. (see figure 38). Therefore, the erasing algorithm removes the memory page addresses kept in the P2M table and addresses point to its data structure. Moreover, to reduce the overload induced from erasing all the page addresses from the CPU cache, the proposed solution searches for memory pages that host P2M data structure and VM's memory addresses and removes them from the CPU caches.

6.2.5 Implementing the Solution's Code on the Xen's Kernel

Writing code of the proposed solution inside Xen's kernel was a risky and challenging task due to the complex nature of Xen architecture. The Xen architecture has been written in C language including hundreds of various functions and procedures, which are used for different purposes. A part of our implementation was to investigate and identify the correct functions and locations for the proposed solution.

After conducting a considerable number of experiments, we focused on data structure on architecture and common Xen's kernel files, which control the entire context switch happening between VMs and VCPUs and the scheduling of tasks.

6.2.5.1 The Arc Xen's Kernel Data Structure

The Arc Xen's kernel data structure has two architectures, which are ARM and X86. It was easy to choose the X86 as it is the backbone architecture of machines used around the world. Inside the structure of X86, there are 10 other data structures and 55 kernel's files. They control all the hardware components such as CPU and memories. Inside the structure of X86, there is a domain.c file, which manages the context switch happening between virtual machines (VMs). The domain.c file has been used to implement the tracing algorithm in the function (paravirt_ctxt_switch_from). This function would be used when a context switch happens and used to save VMs' data. In the proposed solution the tracing algorithm will examine when this function is executed, then the erasing algorithm will be triggered to erase all the VM's addresses from all CPU cache levels.

6.2.5.2 The Common Xen's Kernel Data Structure

The proposed solution (tracing and erasing algorithms) was implemented in two important files (sched_credit.c and schedule.c in common data structure). They control VCPUs' running queues and scheduling tables.

1. The sched_credit.c represents the fair CPU scheduler, which deals with Symmetric MultiProcessing (SMP) on the host machines. More details about the credit scheduler are demonstrated on the section 6.1.1. In the file, we looked for functions and procedures that deal with credit given to VCPUs in order to finish their tasks. The runq_remove is a function that removes any VCPU from the credit scheduler queue when its time is finished. Thus, the erasing algorithm is added to the code of this function.

2. The Xen hypervisor under certain circumstances gives VMs ability to wake up, performing some tasks, and going back to idle state [111]. Thus, a number of VMs and VCPUs used by them can perform some hidden tasks and operations that are not added on the running queue or credit schedulers. Thus, the solution's algorithms were added to the schedule.c file to trace any change on the run state of VCPUs. The schedule.c file has a large number of procedures and functions that are responsible for creating and destroying VCPUs, tracing run state of VCPUs, creating and destroying domains (VMs) and etc. Therefore, the tracing algorithm was added to this file, specifically the code was injected at the trace run state function. In this function, all parameters assigned to a VCPU, a domain that uses the VCPU and CPU's core which is occupied by it, are tested constantly. If any alteration happens to the state of the VM that uses the VCPU or the CPU core utilised by it, the Tracing algorithm will trigger the Erasing algorithm.

6.3 Implementation of the New Detection Solution

We have implemented the new proposed cache side-channel detection solution using the open source Xen Hypervisor version 4.4. The proposed solution has two main algorithms (measuring and analysis algorithms) which were coded and embedded in different places in the Xen's kernel in order to detect any cache side-channel attack that causes any abnormality in the CPU caches' behaviour or CPU cache misses.

6.3.1 Measuring Algorithm

It focuses on calculating the time taken in every VCPU's fetch cycle and recording the occurring time of that fetch. The executing time of the fetch cycle and happening time are added to a new data structure that is linked to every VCPU running in the host machine.

Measuring algorithm

```
Void the measuring_function (struct vcpu *v)/*injected in the fetch step*/
{
   Unsigned long str, r;
   str=rdtsc();
   fetch ();
   r=rdtsc();
   v->cache_miss_time=(r-str);
}End
```

6.3.2 Analysis Algorithm

This algorithm concentrates on analysing the collected data of fetch cycles' time and tells whether a cache side-channel attack has been launched or not. When a VCPU overlapping access to a CPU core happens, the analysis algorithm starts counting cache misses and compares it with a predefined threshold of the number of CPU cache misses in normal conditions. If the whole number of CPU cache misses is greater than threshold, it is likely abnormal behaviour and the standard deviation of the time of CPU cache misses reoccurring will be calculated. If the standard deviation is greater than the threshold's standard deviation, it is definitely a cache side-channel attack lunched by a VM.

Analysis algorithm

```
Void analysis_function (struct vcpu *v)/*injected where the VCPU release the control of
PCPU*/
{
        If (v->cache_miss_time>threshold1)
        Count cache misses ();
        Calculate percentage of cache misses ();
        If (percentage>threshold2)
         {
                Calculating the summation of cache misses times ();
                Calculating the mean of cache misses times ();
                Calculating the sum of deviation of cache misses times ();
                Calculating the standard deviation of cache misses times ();
        If (standard_deviation<threshold3)
        Printk (" The VCPU %d which uses by Domain %d launched CSSA\n",
v->vcpu_id, v->domain->domain_id);
        }
        Empty the list after the VCPU release the control of PCPU ();
}End
```

6.3.3 Implementing the Detection Solution in the Kernel of the Xen Hypervisor

The solution’s analysis algorithm was implemented on the same kernel’s data structures, files and functions mention in section 6.2.1 to implement the novel lightweight solution for preventing cache side-channel attacks. However, there were two crucial steps for this solution to work perfectly. Firstly, creating a new data structure to VCPUs in the kernel of Xen hypervisor and that was not an easy task with considerable restrictions on modifying the VCPUs’ data structure. For example, the VCPU’s structure in the Xen’s kernel has limited memory to be constructed. Hence, we had to change types of number of parameters (form uint64_t to uint8_t or other types) and accommodate these changes in other linked data structure such as domain and timer data structures. Secondly, finding the code used by VCPUs to execute Fetch cycles and adding the measuring algorithm to it.

6.3.3.1 Creating a New Data Structure

In order to create a new data structure for every VCPU to save all the fetch cycles’ execution time, we looked for the Xen’s kernel files that are responsible for creating VCPUs’ data structure. It was difficult to identify the components of Xen which can help implementing the detection solution as most of the codes are nested and linked to each other. Thus, we have examined various components and functions of the Xen’s kernel to find the file that creates VCPUs’ data structure. The Xen’s kernel has a folder named include. The include data structure has most of the various libraries and files that responsible for creating memory pages and assigning them, controlling the low levels hardware and structures of VCPUs, etc. After examining the include’s data structures and files, the file responsible for creating VCPUs’ structure was found in the xen folder. It is called sched.h. In this file, a data linked list structure was implemented inside the data structure of the VCPU to keep track of all CPU cache misses caused by a VCPU. The code used to create it, is presented in a first part the appendix B.

6.3.3.2 Fetch Instruction

Looking for the fetch instruction used by VCPUs was really the toughest part and time consuming task of implementing the solution. We had to dig deep for this instruction in every data structure, file, and code lines in functions used in the Xen's kernel. Finally, we found the fetch instruction that is used by VCPU to get data either from CPU caches or RAM. It was found in file traps.c inside the x86 data structure. In this file, we wrote the measuring algorithm, which will be responsible for recording every time taken to execute a VM's fetch cycle.

6.4 Summary

This chapter described how the prevention and detection solutions to cache side-channel attacks are implemented. Both of the proposed solutions were implemented in the Xen hypervisor. The Xen hypervisor is an open-source bare-metal or type-1 hypervisor implementation that resides between the hardware and the operating system in order to grant virtual averment. It is evolved by a world-wide community of researchers, employees of companies and individuals. It is being used on contemporary hardware architectures of public clouds including Amazon's EC2. Detailed steps and algorithms developed to implement the solutions were fully explained here. The proposed prevention cache side-cannel attacks solution has two algorithms, which are tracing algorithm and an erasing algorithm. They were implemented in the kernel of the Xen. Furthermore, data structures and files that the new detection solution where implemented in were also demonstrated in this chapter. We have implemented the new proposed cache side-channel detection solution in the open source Xen Hypervisor version 4.4. The proposed solution has two main algorithms (measuring and analysis algorithms) which were coded. The next chapter will show in details how the proposed prevention and detection solutions will be evaluated. We conducted two evaluations, which are security and performance evaluations. The evolutions' results are compared with results from other proposed prevention and detection cache side-channel attacks solutions implemented in the same testbed.

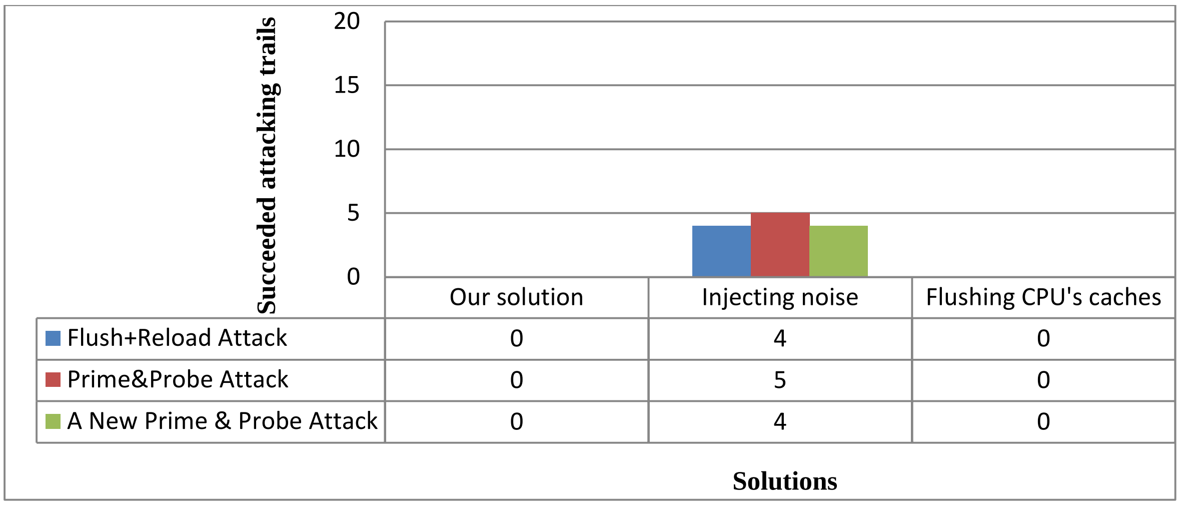
CHAPTER 7 Results and Evaluation of the Prevention and Detection Solutions to Cache Side-Channel Attacks

The proposed prevention and detection cache side-channel attacks solutions are evaluated in this chapter. This chapter presents the results and evaluation of both solutions. Two different evaluations were conducted, which are security and performance evaluations. The security evaluation includes test the proposed solution against three types of cache-side channel attacks, which are Prime & Probe, Flush+Reload and our developed new prime and probe cache side-channel attack. Furthermore, the performance evaluation for the proposed solutions will cover the following aspects: the overload induced by the new prevention and detection cache side-channel attacks solutions and compare it with overload generated by other proposed detection and prevention solution to cache side-channel attacks. Both of the new proposed solutions are tested against different factors and conditions such as the number of virtual machine, the Xen's parameters and RAM.

7.1 Results and Evaluation of the Novel Lightweight Prevention Solution

7.1.1 Security Evaluation

The proposed prevention solution was evaluated using three types of access-driven cache side-channel attacks, which are Prime & Probe, Flush+Reload and our developed new prime and probe cache side-channel attack. All of the attacks were given ideal conditions to work. In our security evaluation experiments, we have used only two VMs with *Dom0* and each VM had one VCPU. The VCPUs were pinned to a single CPU core in the experiments. Thus, the VCPUs of VMs would overlap access to the CPU core's cache levels (L1and L2) and that would generate the perfect conditions for cache side-channel attacks to work. Moreover, the results obtained are compared with results from unmodified Xen hypervisor, which had nothing changed on its code or working conditions. In order to achieve consistency of the results, we launched the three attacks against the proposed lightweight prevention solution and the normal unmodified Xen hypervisor 20 times. Furthermore, in order to compare our solution's security evaluation results with other well-known proposed solutions, we implemented two proposed solutions for preventing cache side-channel attacks, which are flushing the CPU caches solution [18] and injecting noise to cache timing approach [19]. They were tested against the three types of cache side channel attacks with the same conditions used to test the lightweight prevention solution. As demonstrated in figure 39, we lunched the three attacks against the proposed solutions 20 times. Our proposed solution and the CPU caches flushing solution had the best result and prevents the three types of attacks from gaining any information in 20 times out of 20. However, as described in the section 2.3.2 (background chapter) and in section 7.1.2.1, the CPU caches flushing solution cannot be applied in the cloud computing due to its big overload that exceed 1,000,000 CPU cycles (about 0.025 seconds). In addition, the injecting noise solutions failed at least 4 times out of 20 to prevent the three types of cache side-channel attacks.



	Our solution	Injecting noise	Flushing CPU's caches
🟦 Flush+Reload Attack	0	4	0
🟥 Prime&Probe Attack	0	5	0
🟩 A New Prime & Probe Attack	0	4	0

Figure 39: A comparison of attack trails between our prevention solution and other solutions

7.1.1.1 Flush+Reload Cache Side-Channel Attack

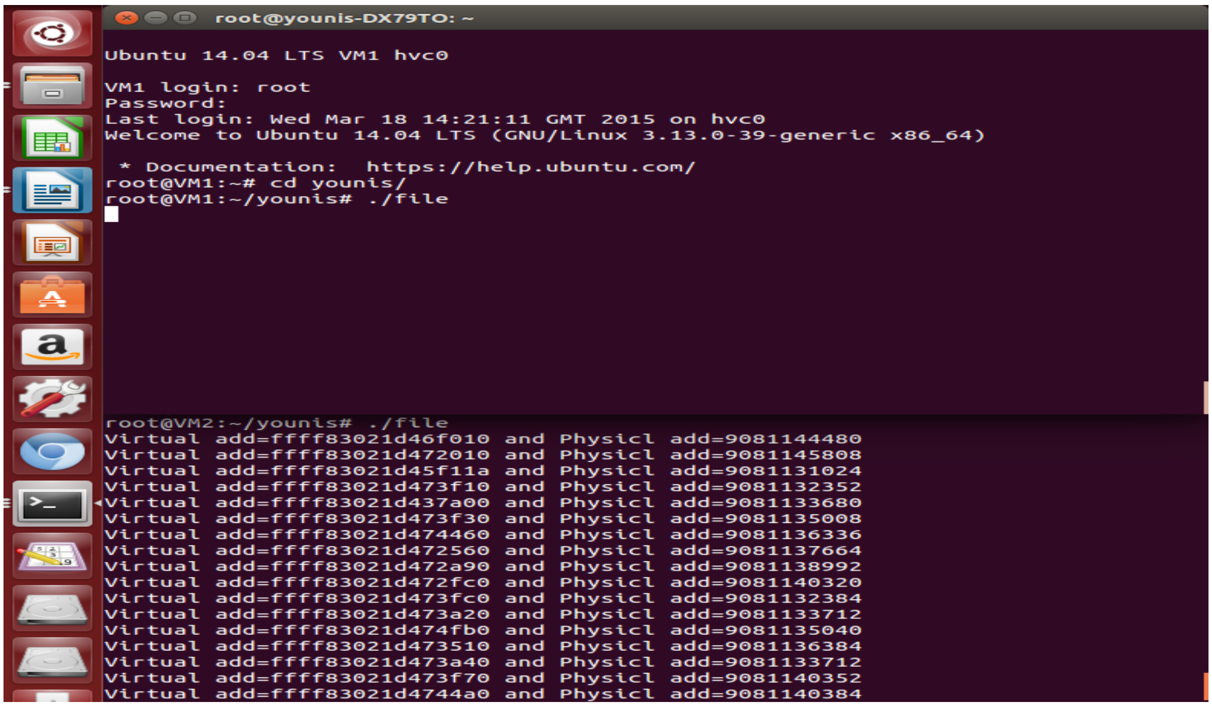
In order to perform Flush+Reload attack, we followed the same steps which were presented at the background chapter in section 2.3.1. One of the VMs was the attacking machine, which executed a small program written in C language to flush the number of memory addresses which were predefined and monitored. The pre-identified addresses were accessed by the victim machine in an early stage. When the VCPU of victim machine released control of the CPU core, the attacker would use the C language code to flush the memory address from the all CPU cache levels and wait to allow the victim machine to access the monitored lines [67]. The attacker would measure the time to access the monitoring lines after the victim machine relinquished CPU resources. If the time was less than the pre-computed threshold (120 CPU cycles), the monitored lines were accessed by the victim machine. Otherwise, the monitored lines were not accessed by the victim machine.

In the normal hypervisor, the attackers succeeded in 20 trials to tell the victim accessed the monitored lines. However, in our solution, the attack did not succeed to derive any information about the accessed memory lines. Finally, the attacker did not have any information about memory lines accessed by the victim.

7.1.1.2 Prime & Probe Cache Side-Channel Attack

The second type of side-channel attacks used in our security evaluation is the Prime & Probe attack. The attacker VM primed the CPU core's caches by using the same technique introduced by Osvik et al. [68]. We primed a number of cache lines with our data. Having waited for the predefined time to allow the victim machine to access the CPU core's caches, the probe stage started. It observed the victim's activity on cache sets. If the victim accesses a primed line, data on the line will be evicted and caused a cache miss. This will yield a higher time to read this line than if it is still untouched.

The Prime & Probe attack was applied on our novel lightweight solution and normal hypervisor 20 times. By using the proposed prevention solution, the attacker did not have any information about the primed cache lines as all of its primed lines and data were erased after relinquishing the CPU core. On the other hand, the attack in the normal hypervisor succeeded in 20 out of 20 trials. The attacker could see the activity of the victim and which primed line had been accessed.



```
root@younis-DX79TO: ~
Ubuntu 14.04 LTS VM1 hvc0

VM1 login: root
Password:
Last login: Wed Mar 18 14:21:11 GMT 2015 on hvc0
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-39-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
root@VM1:~# cd younis/
root@VM1:~/younis# ./file
```

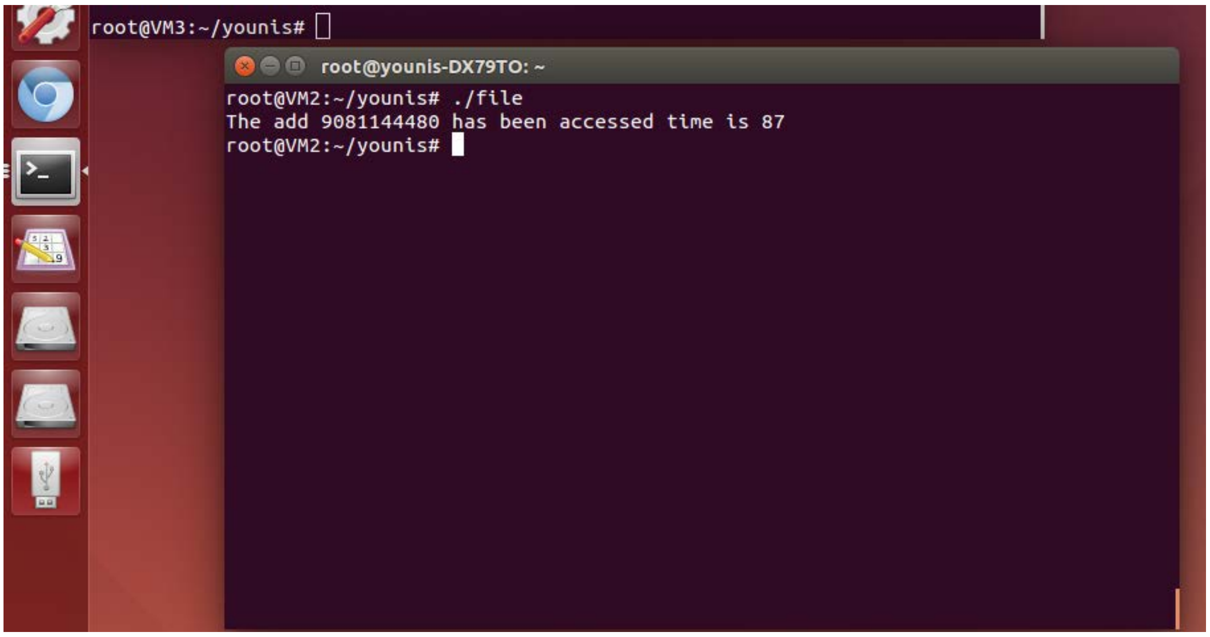
```
root@VM2:~/younis# ./file
Virtual add=ffff83021d46f010 and Physicl add=9081144480
Virtual add=ffff83021d472010 and Physicl add=9081145808
Virtual add=ffff83021d45f11a and Physicl add=9081131024
Virtual add=ffff83021d473f10 and Physicl add=9081132352
Virtual add=ffff83021d437a00 and Physicl add=9081133680
Virtual add=ffff83021d473f30 and Physicl add=9081135008
Virtual add=ffff83021d474460 and Physicl add=9081136336
Virtual add=ffff83021d472560 and Physicl add=9081137664
Virtual add=ffff83021d472a90 and Physicl add=9081138992
Virtual add=ffff83021d472fc0 and Physicl add=9081140320
Virtual add=ffff83021d473fc0 and Physicl add=9081132384
Virtual add=ffff83021d473a20 and Physicl add=9081133712
Virtual add=ffff83021d474fb0 and Physicl add=9081135040
Virtual add=ffff83021d473510 and Physicl add=9081136384
Virtual add=ffff83021d473a40 and Physicl add=9081133712
Virtual add=ffff83021d473f70 and Physicl add=9081140352
Virtual add=ffff83021d4744a0 and Physicl add=9081140384
```

Figure 40: Translating virtual to physical address

7.1.1.3 A New Prime & Probe Cache Side-channel Attack

The third type was our new Prime & Probe attack. The attacker got the virtual addresses used by its VM and translated them to physical addresses (see figure 40). The physical addresses were primed with data and left to let the victim access them. After waiting for a predefined time, the attacker machine lunched the probe stage and accessed the physical addresses. It measured the time to access the physical addresses and data on the physical address.

In our solution, the attacker did not have any information about the accessed virtual and physical addresses as all of its addresses were erased after relinquishing the CPU core. On the other hand, the attack in the normal hypervisor succeeded in 20 out of 20 trials. The attacker could see the activity of the victim and which addresses had been accessed. In the figure 41, the attacker in VM2 launched the attack and waited for the victim to access the monitored address. Then, the new Prime & Probe attack presented a message to the attacker stated an address had been accessed by another VM.



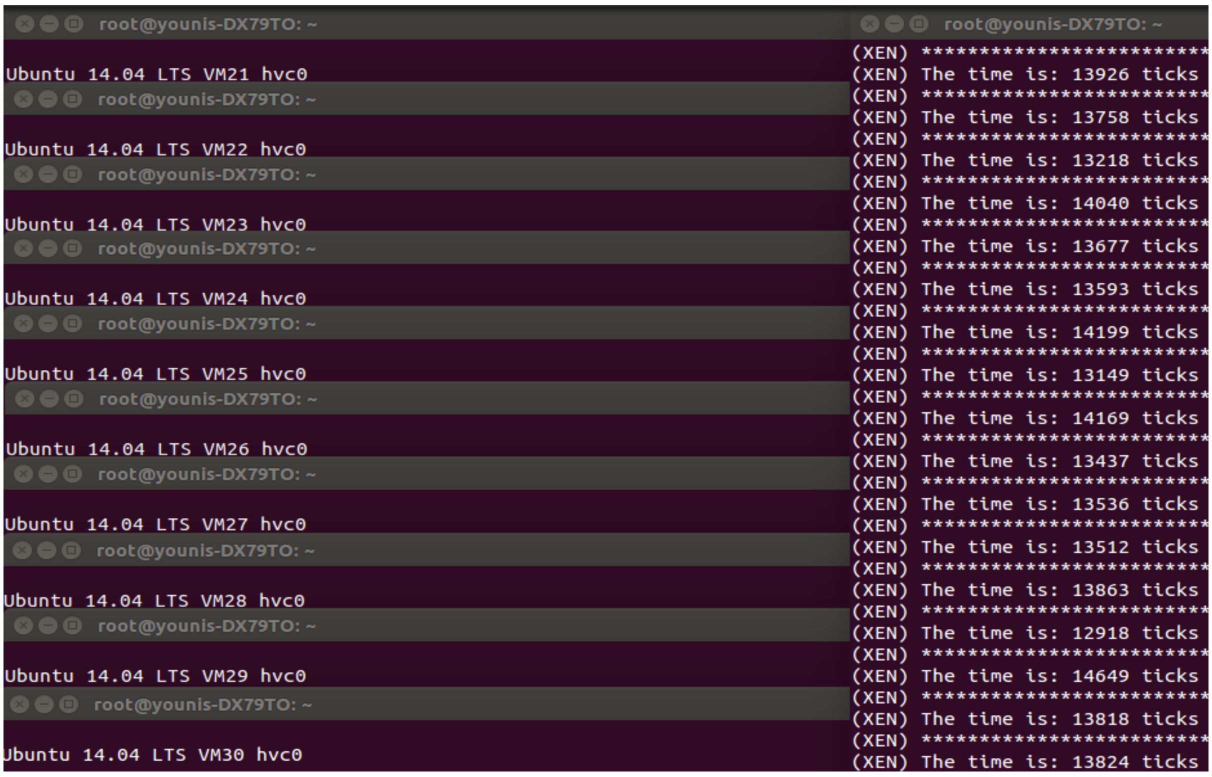
```
root@VM3:~/younis#
```

```
root@VM2:~/younis# ./file
The add 9081144480 has been accessed time is 87
root@VM2:~/younis#
```

Figure 41: A successful trial to attack on a normal hypervisor

7.1.2 Performance Evaluation

In this subsection, we present our performance evaluation experiments of the novel lightweight solution under a variety of Xen credit scheduler parameters and virtual machines. We calculated the overhead introduced by the proposed prevention solution and compared it with two other approaches that proposed to prevent cache side-channel attacks [18,19]. We conducted three different performance evaluation experiments. The first one was comparing the overload introduced by the two other approaches with our solution. This experiment was conducted with 30 VMs working at the same time on the server. The second experiment was identical to the first one, but it examined the three proposed approaches for different numbers of hosted VMs. The final experiment was testing the proposed solution and a normal Xen hypervisor for a heavy workload and comparing the results. Furthermore, it is worth mentioning that RDTSC command was used to measure the time taken for performing all conducted evaluation experiments (see section 5.1 in chapter 5).



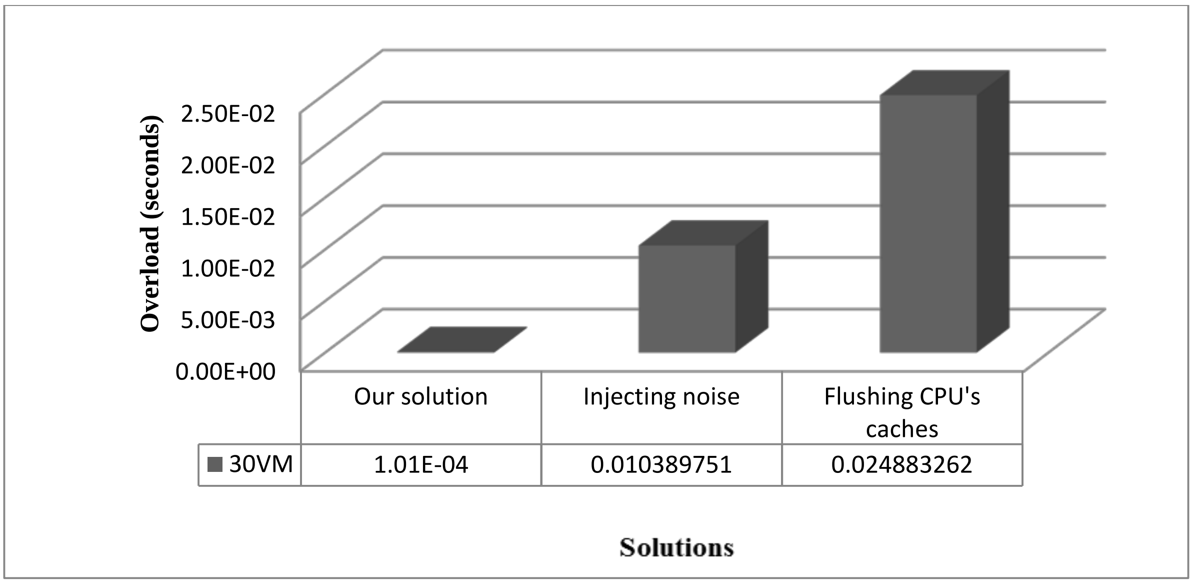
```
root@younis-DX79TO: ~
Ubuntu 14.04 LTS VM21 hvc0
root@younis-DX79TO: ~
Ubuntu 14.04 LTS VM22 hvc0
root@younis-DX79TO: ~
Ubuntu 14.04 LTS VM23 hvc0
root@younis-DX79TO: ~
Ubuntu 14.04 LTS VM24 hvc0
root@younis-DX79TO: ~
Ubuntu 14.04 LTS VM25 hvc0
root@younis-DX79TO: ~
Ubuntu 14.04 LTS VM26 hvc0
root@younis-DX79TO: ~
Ubuntu 14.04 LTS VM27 hvc0
root@younis-DX79TO: ~
Ubuntu 14.04 LTS VM28 hvc0
root@younis-DX79TO: ~
Ubuntu 14.04 LTS VM29 hvc0
root@younis-DX79TO: ~
Ubuntu 14.04 LTS VM30 hvc0
```

```
root@younis-DX79TO: ~
(XEN) *************************
(XEN) The time is: 13926 ticks
(XEN) *************************
(XEN) The time is: 13758 ticks
(XEN) *************************
(XEN) The time is: 13218 ticks
(XEN) *************************
(XEN) The time is: 14040 ticks
(XEN) *************************
(XEN) The time is: 13677 ticks
(XEN) *************************
(XEN) The time is: 13593 ticks
(XEN) *************************
(XEN) The time is: 14199 ticks
(XEN) *************************
(XEN) The time is: 13149 ticks
(XEN) *************************
(XEN) The time is: 14169 ticks
(XEN) *************************
(XEN) The time is: 13437 ticks
(XEN) *************************
(XEN) The time is: 13536 ticks
(XEN) *************************
(XEN) The time is: 13512 ticks
(XEN) *************************
(XEN) The time is: 13863 ticks
(XEN) *************************
(XEN) The time is: 12918 ticks
(XEN) *************************
(XEN) The time is: 14649 ticks
(XEN) *************************
(XEN) The time is: 13818 ticks
(XEN) *************************
(XEN) The time is: 13824 ticks
```

Figure 42: The overload induced by the lightweight solution

7.1.2.1 Overload

One of the most important features in the proposed prevention solution is the very small overhead introduced, which does not exceed 15,000 cycles as illustrated in figure 42. In order to validate our solution, we implemented two other proposed solutions, which are flushing the CPU caches [18] and injecting noise to cache timing[19]. Each one of them was implemented separately and tested for 30 VMs running at the same time in the server. Furthermore, the biggest overload was induced by flushing the whole CPU caches, which induced 0.0249 seconds every time a switch happens from one VM to another as demonstrated in figure 43. This overload will be added to the workload the CPU core has to do. Injecting noise to cache timing approach induced 0.0104 seconds overload every time a context switch happens. It is less than the whole flushing approach, yet it is still very high and unacceptable. Moreover, the novel lightweight solution induced 1.01e^{-04} seconds, which was the lowest value compared with others and more realistic to be used in the cloud. Although, the results are highly dependent on the cache hardware in the server's CPU and size of data to be erased from the CPU caches, the proposed lightweight solution removes only used memory address from the cache. It also erases memory pages' addresses that are currently used by the VM not the whole assigned memory pages.



	Our solution	Injecting noise	Flushing CPU's caches
■ 30VM	1.01E-04	0.010389751	0.024883262

Figure 43: The induced overload from each solution with 30 VM

It is apparent from figure 44, that using a translation function to get a page's address that has a VM's virtual address, reduced the overload dramatically as passing all VM's pages address can cause more than 100,000 cycles. We used a new approach to get the page address used by the VM. This page address is translated from a virtual address used by the VM. After getting the page address, both the page address and virtual address are erased by using the Clflush CPU command that presented at chapter 6 in section 6.2.3. The addresses passed to the Clflush command and it instantly flushed them from the cache levels. Moreover, calculating the Fibonacci sequence with 10^6 iterations was used to examine the overload induced by either deleting the whole VM's pages or just pages that have virtual address. A security evolution was conducted to examine the effect of erasing pages that have virtual address used by the VM only and deleting the whole VM's pages. It was found that removing pages that have virtual address technique from the caches does not affect the level of security provided by the proposed solution.

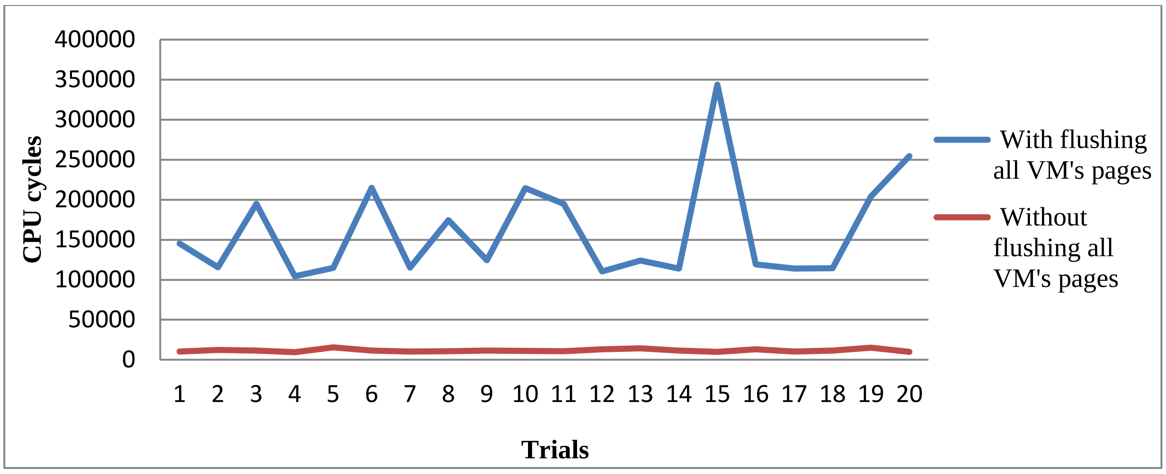
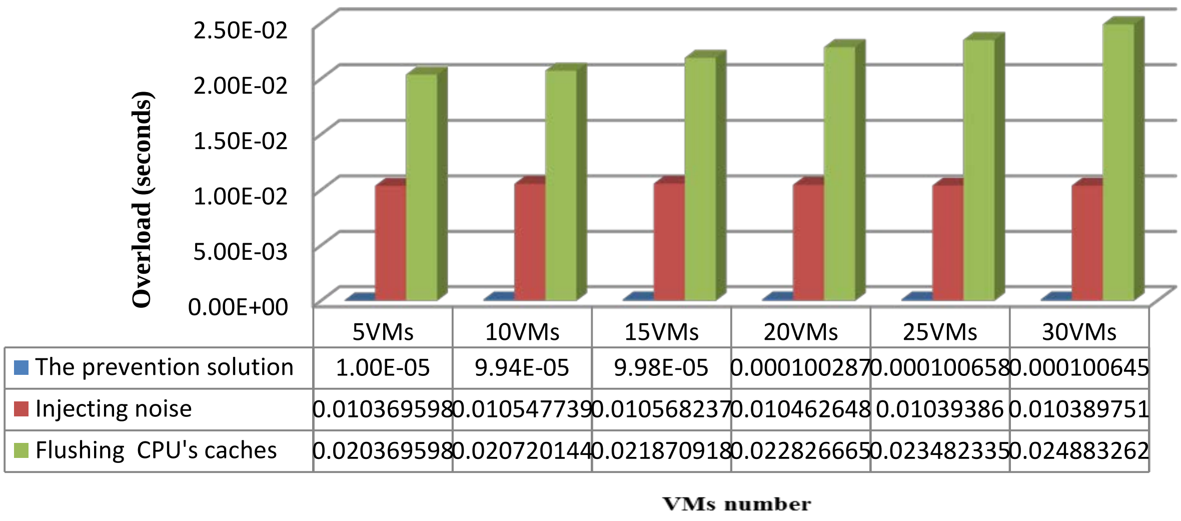


Figure 44: The overload induced with and without translating function

7.1.2.2 VM Number

Due to the lack of server resources, we could not run more than 30 VMs at the same time in the server. We used the same configuration of overload experiment with varying numbers of VMs. In figure 45, we started with 5 VMs till reaching 30 VMs. All of the virtual machines were running concurrently within the server. The most important observation from this experiment was increasing the number of VMs did not increase the overload induced from the proposed prevention solution.



	5VMs	10VMs	15VMs	20VMs	25VMs	30VMs
🟦 The prevention solution	1.00E-05	9.94E-05	9.98E-05	0.000100287	0.000100658	0.000100645
🟥 Injecting noise	0.010369598	0.010547739	0.010568237	0.010462648	0.01039386	0.010389751
🟩 Flushing CPU's caches	0.020369598	0.020720144	0.021870918	0.022826665	0.023482335	0.024883262

Figure 45: The induced overload from each solution with various VM numbers

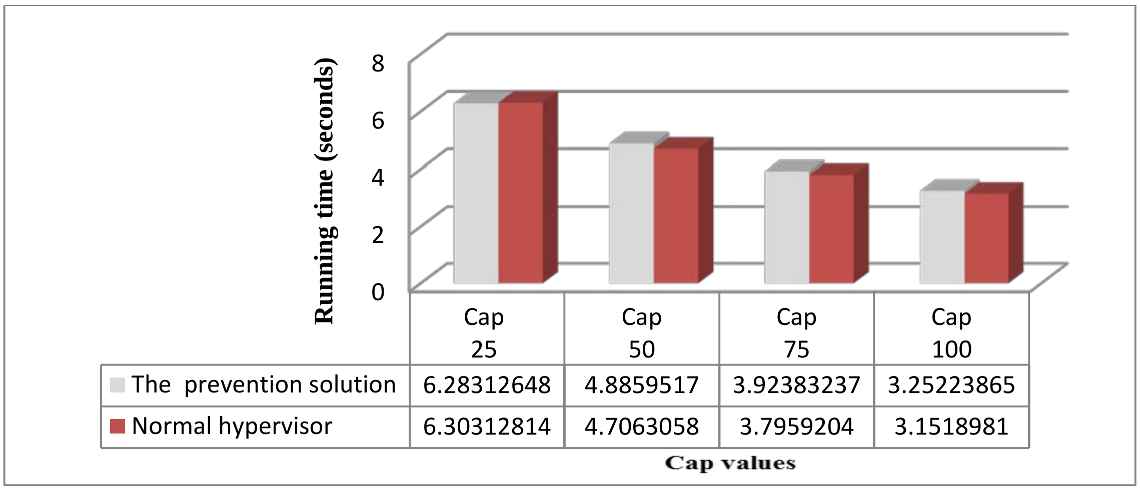
7.1.2.3 Heavy Workload

In this experiment, we tested the proposed solution for heavy workloads with varying values of Xen credit scheduler parameters. The used workload was a program calculating the

Fibonacci sequence with 10^9 iterations. This workload simulates heavy workload found in the cloud such as web servers. Two virtual machines were used for all conducted experiments. Furthermore, the proposed prevention solution added less than 0.0031% to the total execution time of the used program to test the solution in the all conducted experiments.

1. Cap

   As aforementioned in the introduction of the Xen hypervisor at chapter 6 in section 6.1.1, changing a Cap value of a VM can affect the time needed for executing the VM tasks and operations (see figure 46). In the figure, we changed the Cap value to examine its effect on the proposed solution. Reducing the Cap value to 25, can double the time to calculate the Fibonacci sequence (10^9 iterations) from 3 to 6 seconds. However, execution results from our solution are almost equal to the normal hypervisor's execution results.

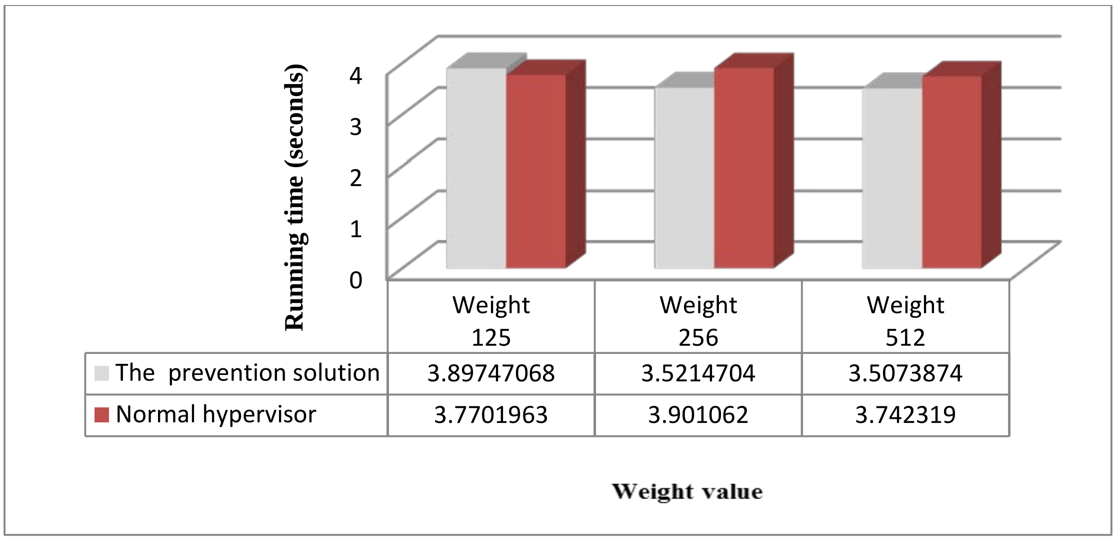


	Cap 25	Cap 50	Cap 75	Cap 100
⬜ The prevention solution	6.28312648	4.8859517	3.92383237	3.25223865
🟥 Normal hypervisor	6.30312814	4.7063058	3.7959204	3.1518981

Figure 46: Heavy workload executed on the proposed prevention solution and normal hypervisor with different Cap values

2. Weight

   The credit scheduler in the Xen uses this parameter to manage every VM weight. Thus a VM with a weight that is as twice as another VM's weight can have double execution time. As presented in figure 47, we altered the weight parameter of the VMs in order to test the performance of the prevention proposed solution, yet no considerable observations can be reported. Furthermore, some fluctuations in the results can be seen (the first result), yet that were caused by other operations executed by other processed in the CPU caches.

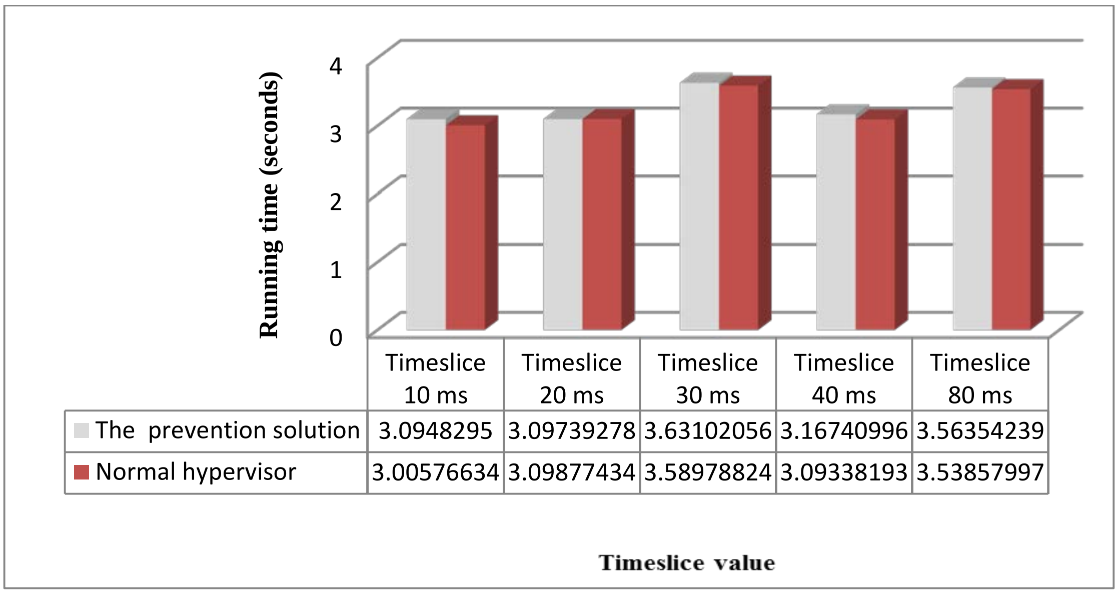


	Weight 125	Weight 256	Weight 512
⬜ The prevention solution	3.89747068	3.5214704	3.5073874
🟥 Normal hypervisor	3.7701963	3.901062	3.742319

Figure 47: Heavy workload executed on the prevention solution and normal hypervisor with different Weight values

3. Timeslice

Results of changing the Timeslice values were expected. In figure 48, we tested the calculation code with 5 different values of Timeslice from 10 to 80. However, the program's execution time in both hypervisors is almost identical.

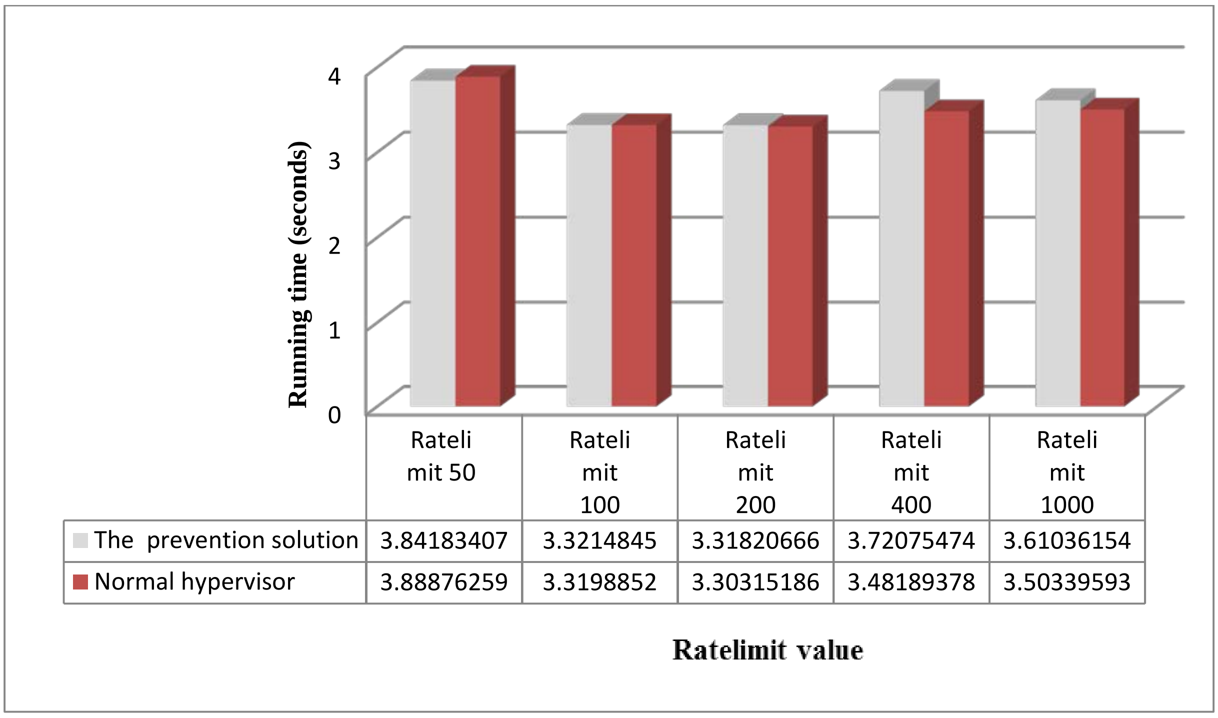


	Timeslice 10 ms	Timeslice 20 ms	Timeslice 30 ms	Timeslice 40 ms	Timeslice 80 ms
⬜ The prevention solution	3.0948295	3.09739278	3.63102056	3.16740996	3.56354239
🟥 Normal hypervisor	3.00576634	3.09877434	3.58978824	3.09338193	3.53857997

Figure 48: Heavy workload executed on the prevention solution and normal hypervisor with different Xen Timeslice values

4. Ratelimit

Varying values of the Ratelimit were tested in normal hypervisor and in the proposed solution. The calculation time of the program almost equalled that on our solution and the normal hypervisor as illustrated in the figure 49. Moreover, as the CPU caches are shared among various processes and controlled directly by the CPU, there were some fluctuations on the results due to other operations executed on the CPU and increased the value of the RDTSC that was used to measure the time taken for conducting the experiments.



	Ratelimit 50	Ratelimit 100	Ratelimit 200	Ratelimit 400	Ratelimit 1000
■ The prevention solution	3.84183407	3.3214845	3.31820666	3.72075474	3.61036154
🟥 Normal hypervisor	3.88876259	3.3198852	3.30315186	3.48189378	3.50339593

Figure 49: Heavy workload executed on the prevention solution and normal hypervisor with different Xen Ratelimit values

7.2 Results and Evaluation of the Novel Infrastructure Detection Solution

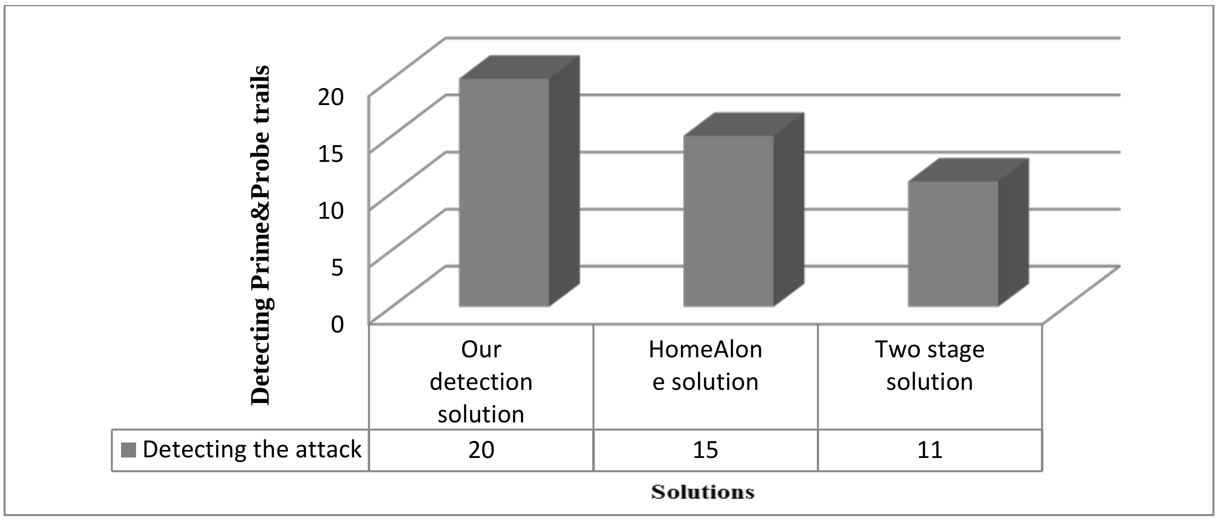
In this section, we evaluate the proposed detection solution and report the results of our evaluation. We conducted performance and security evaluation. We also illustrate some additional implementation details that were needed to implement the solution in *x86* architecture and Ubuntu 14.04, which was the host operating system. In addition, we used the aforementioned points presented in our goals the section 5.3.1 in chapter 5 as research questions to evaluate the proposed solution. Our experiments were conducted using the same testbed utilised to evaluate the novel lightweight solution.

7.2.1 Security Evaluation

The proposed detection solution and normal hypervisor were evaluated using three types of access-driven cache side-channel attacks, which are Prime & Probe, Flush+Reload and the new

Prime & Probe attacks. The attacks were given ideal conditions. Hence, only two VMs with *Dom0* were used and were pinned to a single CPU core to overlap access to the CPU core's cache levels (L1and L2). Moreover, we run up to 30 VM at the same time to get the average number of CPU cache misses in normal conditions. The average number of CPU cache misses is the first threshold used in our solution. Its value was 60.41% CPU cache misses out of the total number of fetch cycle times for a VM. We also calculated the threshold standard deviation for CPU cache misses times; the variation was between 9500 and 20,000 cycles. Thus, threshold standard deviation of 500 values of cache misses times was 200.314. With applying these thresholds' values 100 times in our full implemented solution and lunching a cache side-channel attack (Prime and Probe attack), we get 0% false negative and 15% false positive. Furthermore, the 15% false positive is caused by VMs when they are just implemented by the hypervisor. Therefore, VCPUs assigned to them will cause a considerable number of CPU cache misses.

In order to compare results obtained from our detection solution with results from other detection solutions, we implemented two detection solutions, which are HomeAlone detection solution [21] and the two stage detection solution [20]. Both of them were implemented and tested against Prime and Probe cache side-channel attack in the same testbed with 2 VMs. As presented in figure 50, the novel infrastructure detection solution has the best results and detects Prime and Probe attack in 20 out of 20. Furthermore, HomeAlone detection solution succeeded 14 times out of 20 to detect the attack, yet the two stage solution detected only 11 out of 20.



	Our detection solution	HomeAlone solution	Two stage solution
■ Detecting the attack	20	15	11

Figure 50: A comparison of attack trails detection between our detection solution and others

7.2.1.1 Flush+Reload Attack

In order to perform Flush+Reload attack, we followed the same steps which were presented in the section 2.3.1(background chapter). One of the VM was the attacking machine, which executed a small code to flush the number of memory addresses which were predefined and monitored [67]. The pre-identified addresses were accessed by the victim machine in an early stage. When the VCPU of victim machine released control of the CPU core, the attacker would use the C code to flush the memory address from the all CPU cache levels and wait to allow the victim machine to access the monitored lines. The attacker would measure the time to access the monitoring lines after the victim machine relinquished CPU resources. If the time was less than the pre-computed threshold, the monitored lines were accessed by the victim machine. Otherwise, the monitored lines were not accessed by the victim machine. Moreover, the used threshold was 120 CPU cycles and it was calculated by using the same code used in [67].

Furthermore, we ran the attack against the novel infrastructure solution 50 times to achieve consistency of the results. The attack was detected every time it was launched. Figure 51 shows one of our trials when VCPU1 was working then VCPU2 took control of the shared CPU core and launched a Flush+Reload attack. In the figure, the normal VM was executing some operations by using VCPU1, which relinquished control over a CPU core. Then, the attacking VM (using VCPU2) started to generate a considerable number of CPU cache misses by performing a Flush stage. These CPU cache misses were already counted by the measurement stage in the detection solution. Instantly, when the attacking machine (VCPU2) released the control of the CPU core, the number of CPU cache misses was compared against the total cache misses threshold. In the figure, the number of cache misses was over 60% of the total cache accessed time by the VM2. Thus, the detection solution calculated the sequence of CPU cache misses. The differences between the cache misses were almost constant. Therefore, the detection solution successfully reported the attack in its first stage (Flush stage) before launching the second stage (Reload stage).

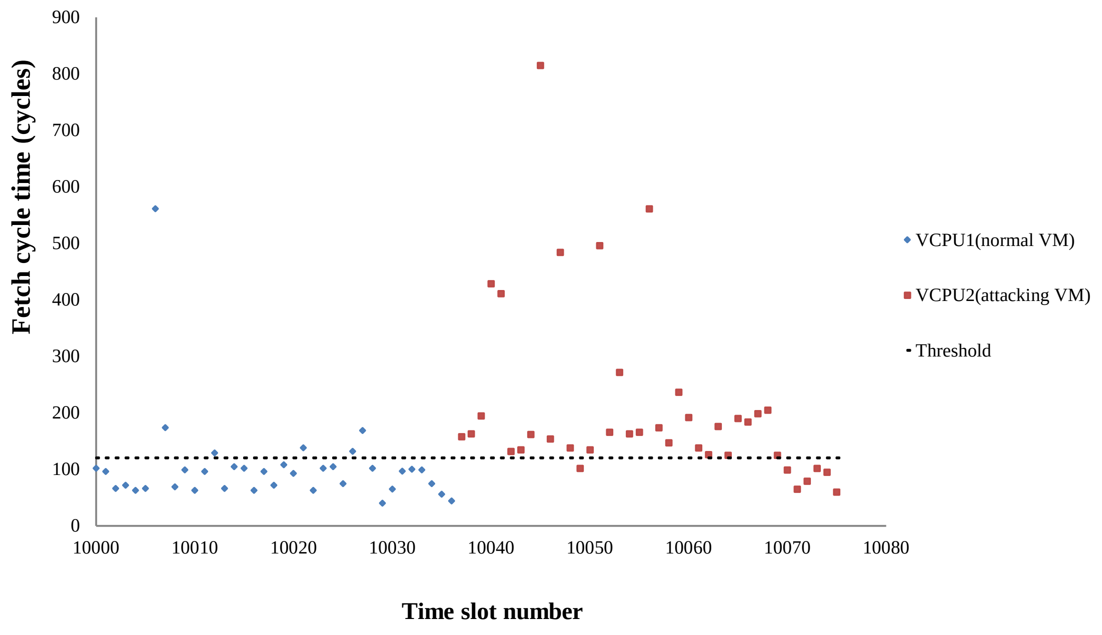


Figure 51: Detecting a Flush+Reload cache side-channel attack

7.2.1.2 The Prime & Probe Attack

The second type of side-channel attacks was used in our security evaluation was the Prime & Probe attack. We follow the same technique introduced by Osvik et al. [68] and explained in section 7.1.1.2 and it was applied on our solution 50 times. The attack was successfully detected in the 50 attacking trails by the infrastructure detection solution. A shown in figure 52, the detection proposed solution was tested with three virtual machines (VM1, VM2 and VM3) where VM2 was the attacking machine. The three machine overlapped access to the cache and every cache access time of the VMs were recorded. When VM1 finished its slot time on the Xen's run queue, its collected cache misses' number compared with the total cache misses threshed, which was greater than the VM1's total number of cache misses. Furthermore, VM2 started to prime a number of CPU cache lines with its data, which caused a big number of cache misses. The detection solution already recorded every time taken by VM2's fetch cycles. When the VM2 finished its execution time slot, the detection solution found VM2 had over 60% cache misses. Therefore, the detection solution's analysis stage started and reported that VM2 launched a cache side-channel attack. Finally, the VM3's fetch cycles times were also recorded by measurement stage in the detection solution. However, after the VM3 relinquished the CPU core, the total of its cache misses time was less than the threshold.

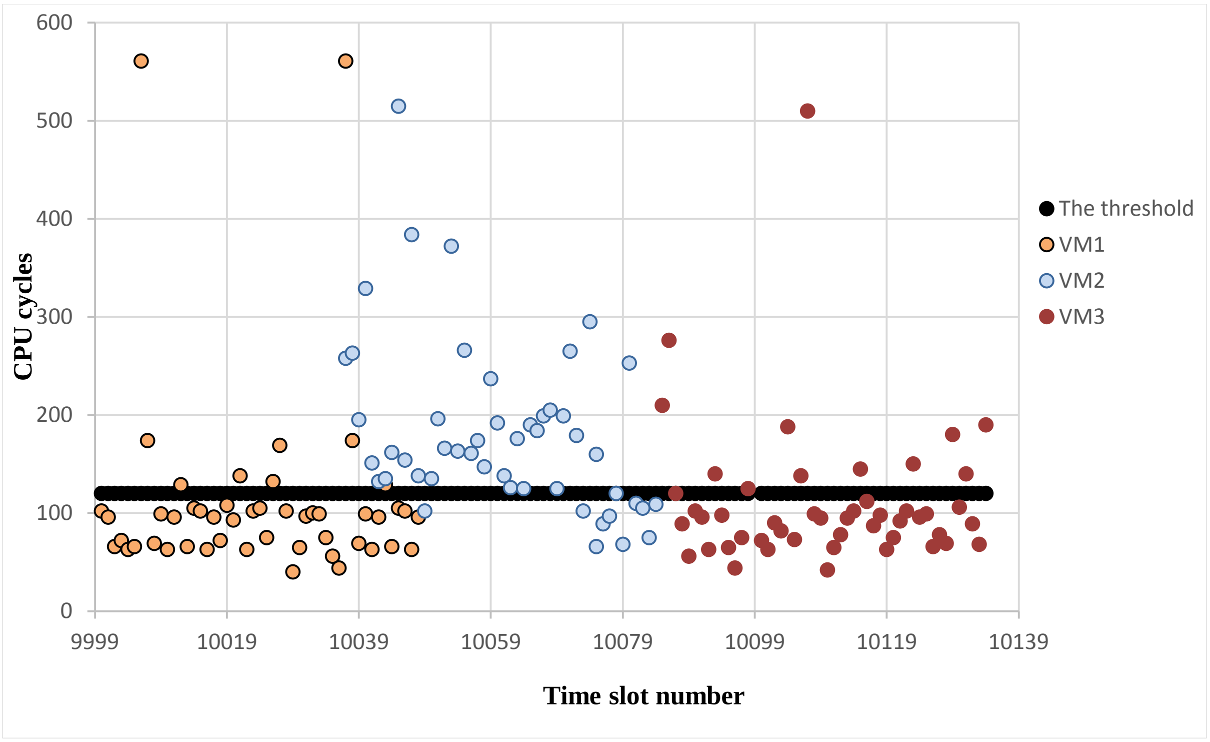
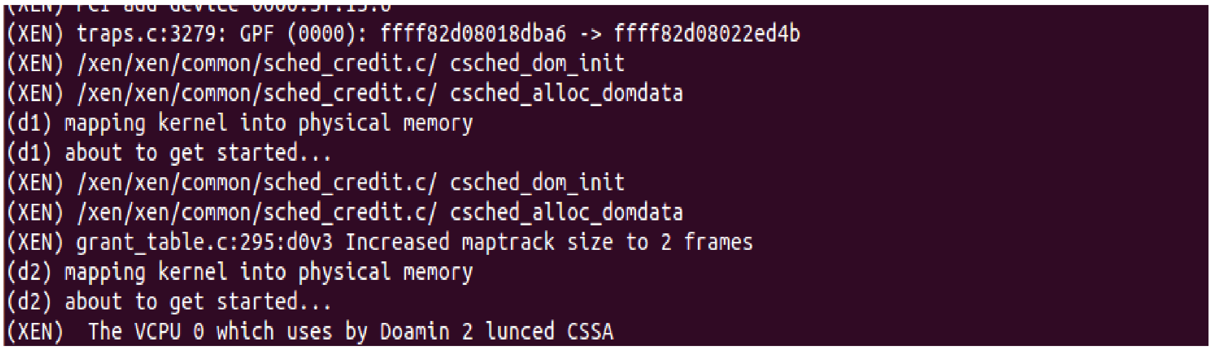


Figure 52: Detecting a Prime & Probe attack

7.2.1.3 The New Prime and Probe Attack

The new Prime & Probe attack was launched by following the same steps explained in section 5.1 (chapter 5) and the security evaluation of the novel lightweight solution 7.1.1.3. The proposed solution was able in 50 times out of 50 trails to detect the attack. Figure 53 presents one of the experiment's trials. In this trial, there were two VMs or domains (domain1 and domain 2). The VM2 in an early stage defined its virtual address, translated them to physical addresses and waited for VM1 to access the CPU caches. When VM1 started executing its operations, the detection solution monitored these operations and recorded all cache misses caused by VM1. After VM1 finished its time slot on the Xen's run queue, the detection solution found out that no suspicious action (a big number of cache misses) generated by VM1. Moreover, VM2 begun to prime a number of targeted physical lines, which caused a considerable number of cache misses recorded by the detection solution's analysis stage. When the measurement stage of the detection solution got a message from the analysis stage reported VM2 had total cache misses greater than the threshold, it found out VM2 carried out a cache side-channel attack. Finally, the detection solution printed a kernel message, which reported a cache side-channel attack carried out by VM 2 that used VCPU0.



```
(XEN) traps.c:3279: GPF (0000): ffff82d08018dba6 -> ffff82d08022ed4b
(XEN) /xen/xen/common/sched_credit.c/ csched_dom_init
(XEN) /xen/xen/common/sched_credit.c/ csched_alloc_domdata
(d1) mapping kernel into physical memory
(d1) about to get started...
(XEN) /xen/xen/common/sched_credit.c/ csched_dom_init
(XEN) /xen/xen/common/sched_credit.c/ csched_alloc_domdata
(XEN) grant_table.c:295:d0v3 Increased maptrack size to 2 frames
(d2) mapping kernel into physical memory
(d2) about to get started...
(XEN)  The VCPU 0 which uses by Doamin 2 lunced CSSA
```

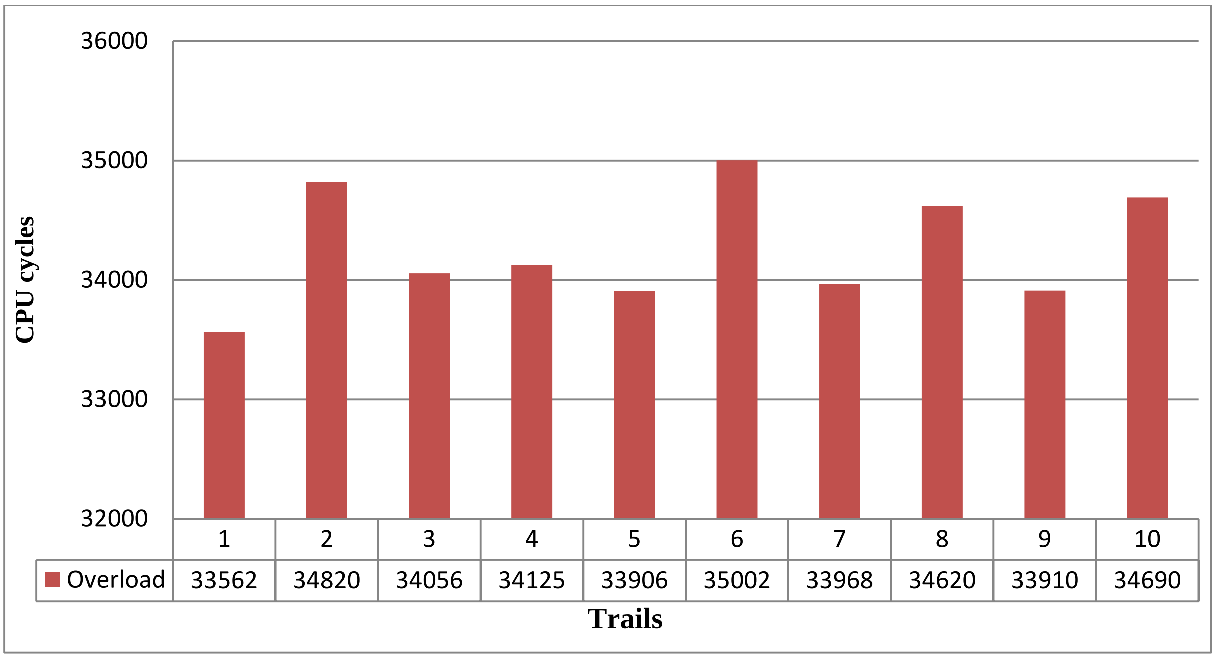
Figure 53: Detecting the Prime & Probe attack in one of experiments trials

7.2.2 Performance Evaluation

In this subsection, we present our performance evaluation experiments of the novel infrastructure detection solution under a variety of Xen credit scheduler parameters and virtual machines. We calculated the overhead introduced by our solution and compared it with normal Xen hypervisor. We conducted three different performance evaluation experiments during the detection processes. The first one was calculating the overload introduced by our solution and comparing it with the two other detection proposed solutions. This experiment was conducted with 10 VMs working at the same time on the server. The second experiment was identical to the first one, but it examined the novel infrastructure solution for different number of hosted VMs. The final experiment was testing the proposed solution and a normal Xen hypervisor for a heavy workload and comparing the results.

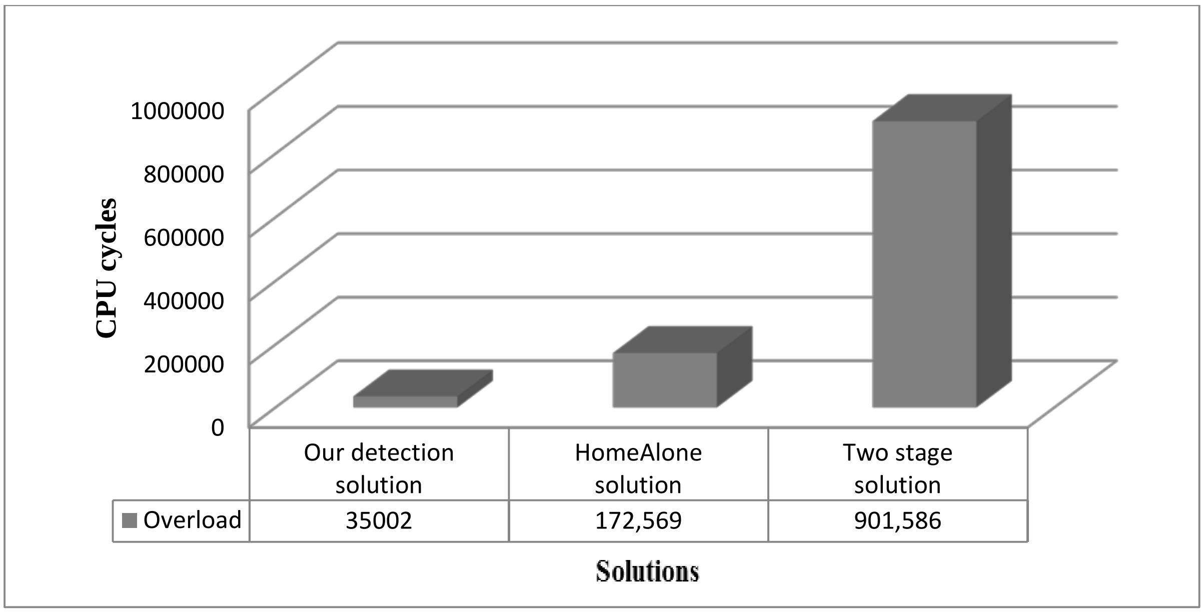
7.2.2.1 Overload

One of the most important features in the novel infrastructure solution is the very small overhead introduced during the detection process, which does not exceed 35,000 cycles as illustrated in figure 54. In the figure, the overload of the detection solution was obtained when the Xen hypervisor hosted 10 VMs running at the same time. Moreover, the overload infrastructure detection solution did not exceed 35,000 cycles due to no applications or software being used to measure CPU caches misses. For example, OProfile software was used in a two-stage mode technique for detecting cache side-channel attacks in cloud computing to count CPU cache misses [21]. OProfile can generate at least 2,000 interrupts per second to the CPU and add 20% extra to the total overload to complete a task [80]. Therefore, when the two stage solution implemented in the testbed, we got at least 900,000 cycles for every time the solution run. Furthermore, HomeAlone solution with single-probe classifier induced at least 150,000 cycles when was implemented in our testbed as shown in figure 55.



	1	2	3	4	5	6	7	8	9	10
🟥 Overload	33562	34820	34056	34125	33906	35002	33968	34620	33910	34690

Figure 54: The generated overload by the detection solution with 10 VM

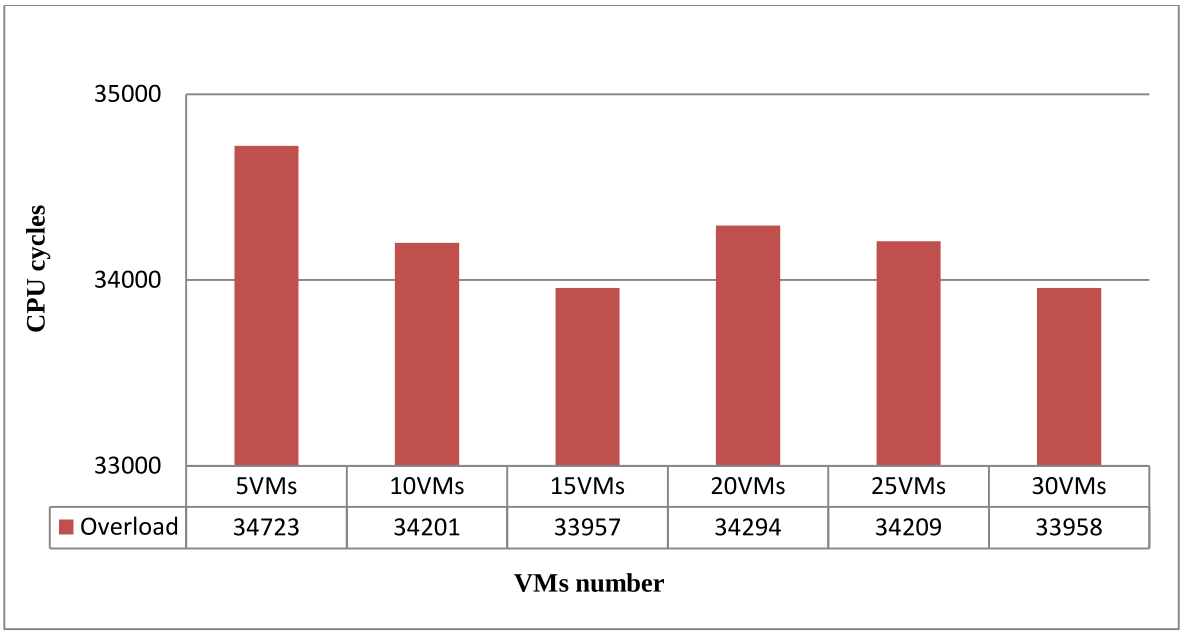


	Our detection solution	HomeAlone solution	Two stage solution
■ Overload	35002	172,569	901,586

Figure 55: Comparing overload generated by our solution with other proposed solutions

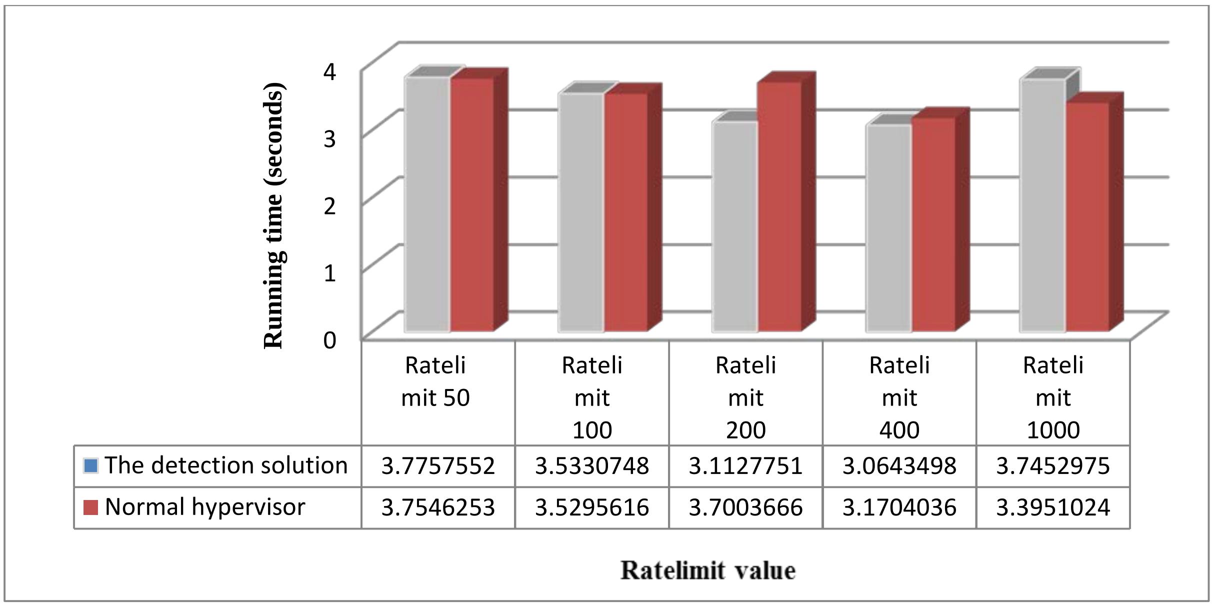
7.2.2.2 VM Number

We used the same configuration of overload experiment with varying numbers of VMs. In figure 56, we started with 5 VMs till reaching 30 VMs. All of the virtual machines were running concurrently within the server. The most important observation from this experiment was increasing the number of VMs did not increase the overload induced from the detection solution.



	5VMs	10VMs	15VMs	20VMs	25VMs	30VMs
🟥 Overload	34723	34201	33957	34294	34209	33958

Figure 56: The induced overload from the detection solution with various VM numbers



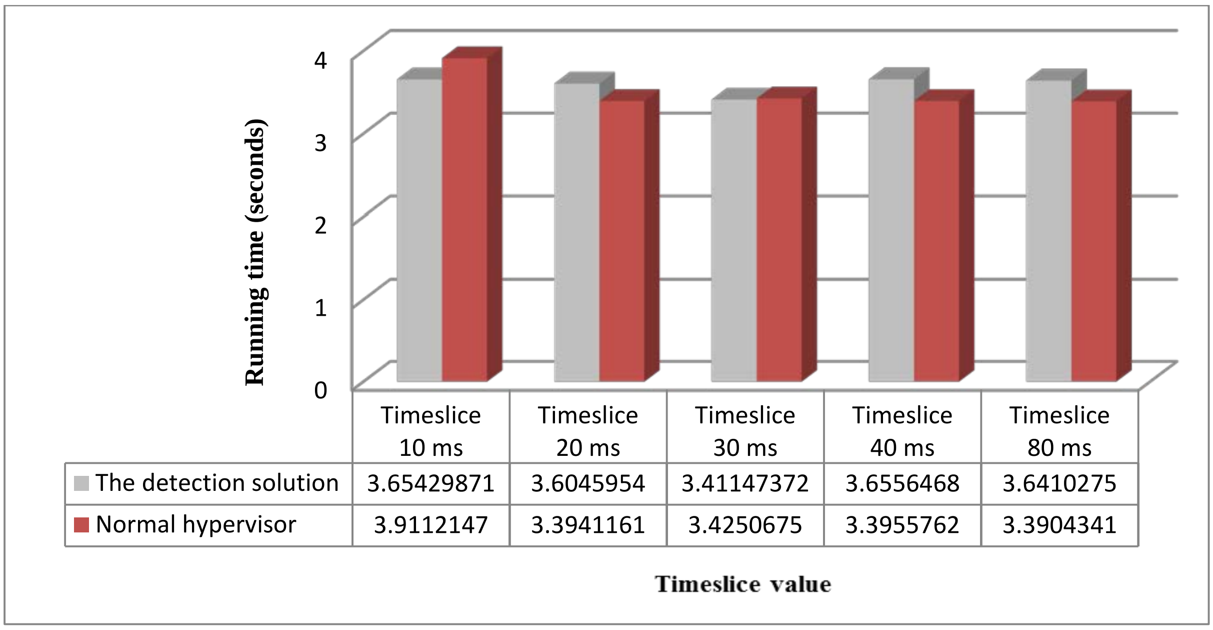
	Rateli mit 50	Rateli mit 100	Rateli mit 200	Rateli mit 400	Rateli mit 1000
🟦 The detection solution	3.7757552	3.5330748	3.1127751	3.0643498	3.7452975
🟥 Normal hypervisor	3.7546253	3.5295616	3.7003666	3.1704036	3.3951024

Figure 57: Heavy workload executed on the detection solution and normal hypervisor with different Xen Ratelimit values

7.2.2.3 Heavy Workload

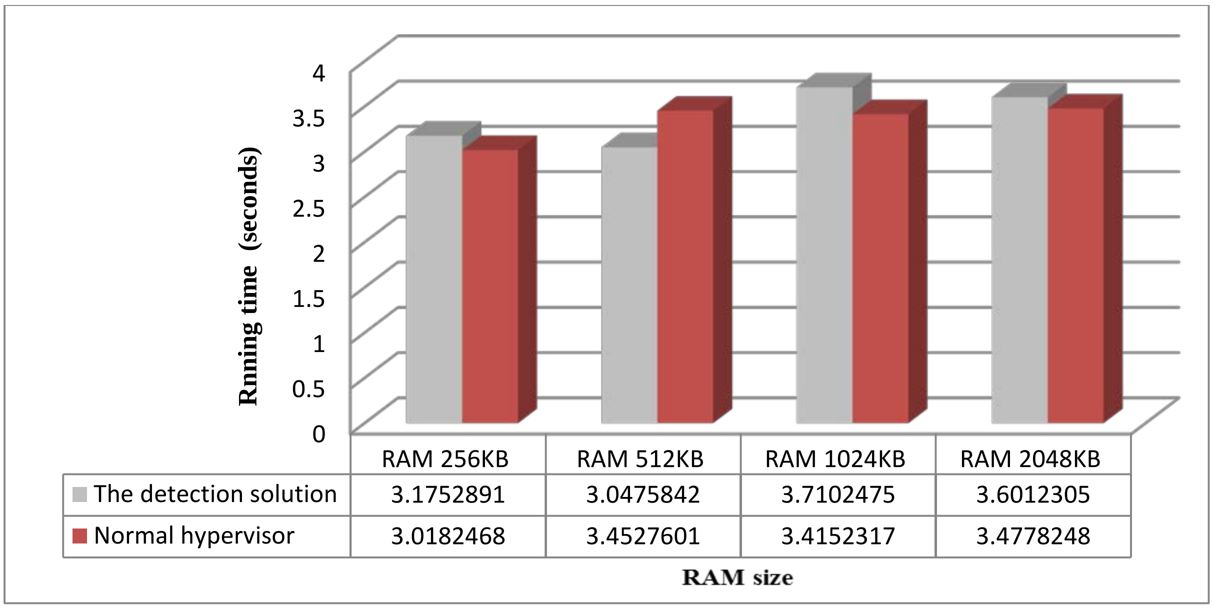
In this experiment, we tested the proposed detection solution for heavy workloads with varying values of Xen Credit scheduler parameters such as Ratelimit and Timeslice as illustrated in figure 57 and 58 respectively. The used workload was a program calculating the Fibonacci sequence with 10^9 iterations. This workload simulates heavy workload found in the cloud such as web servers. Furthermore, we increased the RAM's size that is assigned for the VM from

256KB to 2048 KB (see figure 59) with no major changes to be reported. The workload was tested with different values of Ratelimit and Timeslice. However, the calculation time of the program was almost equal on our solution and the normal hypervisor as illustrated in the figures. As the detection solution was tested in real scenarios, the CPU was performing uncontrolled tasks, which caused some fluctuations on the results. Moreover, the new detection solution added less than 0.0093% to the total execution time of the used program to test the proposed solution in the all conducted experiments.



	Timeslice 10 ms	Timeslice 20 ms	Timeslice 30 ms	Timeslice 40 ms	Timeslice 80 ms
⬜ The detection solution	3.65429871	3.6045954	3.41147372	3.6556468	3.6410275
🟥 Normal hypervisor	3.9112147	3.3941161	3.4250675	3.3955762	3.3904341

Figure 58: Heavy workload executed on the detction solution and normal hypervisor with different Xen Timeslice values



	RAM 256KB	RAM 512KB	RAM 1024KB	RAM 2048KB
⬜ The detection solution	3.1752891	3.0475842	3.7102475	3.6012305
🟥 Normal hypervisor	3.0182468	3.4527601	3.4152317	3.4778248

Figure 59: Testing the proposed detection solution with various VM's RAM sizes

7.3 Summary

In this chapter several tests have been made and illustrated. We have shown how effectively the new proposed prevention and detection cache side-channel attacks solutions can prevent and detect cache side-channel attacks. The proposed prevention solution was evaluated against Prime & Probe, Flush+Reload and our developed new prime and probe cache side-channel attack. The results showed that the attacked could not gain any information in 20 trails and the overload did not exceed 15,000 CPU cycles ($1.01e^{-04}$seconds). Furthermore, we used varied number of virtual machines to test the solution, but no considerable changes had been noticed. The proposed prevention and detection cache side-channel attacks solution were tested against set Xen credit scheduler parameters. However, the execution results from our solution are almost equal to the normal Xen hypervisor's execution results.

The proposed detection cache side-channel attacks solution was evaluated against Prime & Probe, Flush+Reload and the new prime and probe cache side-channel attack. These attacks were launched 20 times. The proposed solution was able to detect the attacks in 20 times out of 20. Moreover, the overload induced by the proposed detection solution did not exceed 35,000 CPU cycles (0.00035 seconds). In the performance evaluation, we tested the new detection solutions under a variety of Xen credit scheduler parameters, virtual machines and RAM size. The most important observation from the evaluation experiment was almost no changes to the solutions overload.

In the thesis, all parts of the research have been presented. In the next chapter, we will have an overall view and point out any potential topics that could be further developed in the proposed AC3 or the new prevention and detection cache side-channel attacks solutions.

CHAPTER 8 Conclusion and Future Work

8.1 Conclusion

Access control is one of the fundamental requirements in order to avoid unauthorized access to systems and protect organizations assets. Although, various access control models and policies have been developed such as Mandatory Access Control (MAC) and Role Based Access Control (RBAC) for different environments, these models cannot fulfil cloud's access control requirements. This is because cloud computing has a diverse set of users with different sets of security requirements. It also has unique security challenges such as multi-tenant hosting and heterogeneity of security policies, rules and domains. This project aimed to propose a cloud based access control model that can meet the security requirements for either cloud service providers or their tenants such as critical infrastructure providers. Furthermore, we aimed to investigate how the proposed access control model can deal with leakage of information in lower levels such as CPU caches and enhance level of security at these levels. More specifically, looking at the leakage of data caused by cache side-channel attacks and how it can be detected and prevented.

The key objectives of this project were investigating in details cloud computing security challenges, critical infrastructures providers' security requirements and gaps in conventional access control models. These key objectives were used to propose a cloud based access control model, which satisfies security requirements for cloud tenants. One of our key objects was looking at the leakage of data in lower levels caused by cache side-channel attacks and proposing prevention and detection solutions to them.

The findings from this study make several contributions to the current literature:

1. A new list of security requirements for different critical infrastructure providers to be moved to cloud computing. The study has gone some way towards enhancing our understanding of various critical infrastructure services to find the common security requirements such as data security, compliance and audit, cryptography and access control. The results of this investigation show that an access control system has been found as one of the core requirements.

2. An in-depth requirement analysis to access control requirements for cloud computing. Finally, this study has demonstrated, for the first time, a new guidelines list of factors should be taken into account for proposing a cloud based access control model.

3. One of the more significant findings to emerge from this project is a novel access control model for cloud computing. The proposed model can fulfil the access control requirements in cloud computing. It facilitates the role and task principles to make assigning privileges very dynamic and easy. It also utilises temporal (time and location) and delegation constraints. In our model, users are assigned to security domains that relate to their roles and actual jobs. Every role within the model is assigned the relevant tasks that allow them to practice their roles. The model secures access and flow of data by marking the data with security labels, which states the data sensitivity. Every task will have a security classification for accessing the data or assets, and the exact permissions needed for accomplishing this task. Thus, any task or process attempting to access the data has to have a classification that dominates the targeted data's or assets' security labels. A risk engine is utilised to deal with dynamic and random behaviours of users. It credits consumers according to their access behaviours. A security tags engine is also used for issuing security tags in semi or untrusted environments and processes. The security tags are utilised in some circumstances and according to the level of trust and security of an environment the system is deployed in. The security tag proposed is to consist of only the information needed for securing access in order to be lightweight.

4. A new Prime and Probe attack is a new way to attack CPU caches and derive some information from them about addresses and data kept on them. In this attack, there is no need for any information about exact CPU cache sets that were accessed by a VM or linked to a memory page. The attack gets virtual addresses used by a VM and translates them to physical addresses. Finally, the attack primes the physical addresses and looks for any changes happening to the time to access them after waiting for a predefined time. Furthermore, the experiments' results shown the attack can tell exactly which addresses have been accessed and by whom.

5. A novel lightweight solution that prevents cache side-channel attacks in cloud computing without interfering with cloud tenants and models.

Cache side-channel attacks are well-known micro architectural attacks, which benefit from correlation between the higher level functionalities of software and the underlying hardware phenomena. Their effect gets worse with cloud computing multi-tenancy's unique vulnerabilities such as clients co-residence and virtual machine physical co-residency. They enable adversaries to interfere with victims on the same physical machine and exfiltrate sensitive information. In this report, we illustrated a novel lightweight solution to prevent cache side-channel attacks in cloud computing. It enables cloud service providers to prevent cache side-channel attacks without any modification to the guests' operating systems or applications. The proposed solution was implemented in identical cloud environments that run the up-to-date version of Linux Ubuntu 14.04 and Xen hypervisor 4.04 with 30 PV guests. It was evaluated in various conditions and circumstances such as testing it for different Xen credit scheduler parameters values and heavy workload. Moreover, our evaluation results proved that the solution's overload induced a maximum of 15,000 CPU cycles. The solution's overload was the lowest among other compared approaches even with heavy workload or hosting a large number of virtual machines.

6. A new infrastructure solution detecting cache side-channel attacks in cloud computing.

   We presented a novel infrastructure solution to detect cache side-channel attacks in cloud computing. The solution attempts to detect cache side-channels in time-shared caches (e.g. L1 instruction/data and L2 per core, and L3 if it is used), particularly Prime and Probe, Flush+Reload and new Prime&Probe attacks. This solution has several practical applications. Firstly, it builds the code in the host kernel to measure time taken in every fetch cycle. Secondly, this is the first study enabling cloud service providers to detect cache side-channel attacks without any modification to the guest's operating system or applications. The solution implemented in identical cloud environments that run the up-to-date version of Linux Ubuntu 14.04 and Xen hypervisor 4.04 with 30 PV guests. It evaluated in various conditions and circumstances such as testing it for different Xen credit scheduler parameters values and heavy workload. Moreover, our evaluation results proved that the proposed solution's overload induced a maximum of 35,000 cycles.

8.2 Future Work

More information on implementing the proposed access control model (AC3) would help us to establish a greater degree of accuracy on this matter. If the debate is to be moved forward, a better understanding of authentication mechanism that can deal with high time and huge space complexity needs to be developed. Future trials should assess the impact of implementing the risk engine and its components, which are used to deal with dynamic behaviours. Further research needs to examine more closely the links between risk engine and the proposed access control model. The issue of implementing and evaluating is an intriguing one which could be usefully explored in further research. The AC3 should be implemented in one of the well-known environments (Windows or Linux) and evaluated in terms of how the security tags can be generated and attached to data, the overload induced by using security tags and risk engine, how the risk engine can cope with heterogeneity, the time taken for grating access to a system deployed the AC3 and the complexity of adding or deleting access policies.

Future research should therefore concentrate on the investigation of the novel lightweight solution to prevent cache side-channel attacks. It should be tested in other architecture such as AMD and ARM. A number of possible future studies using the same experimental set up are apparent to be conducted with various CPU cache sizes, levels and cache lines. Moreover, the proposed solution can be extended by tainting CPU cache lines accessed by or have data related to a VM. These lines will be kept in a new data structure, and every overlapping happen the addresses will be erased from all CPU cache levels. Future trials should assess the impact of implementing the solution in other virtualization environments and operating systems.

A greater focus on linking both of the proposed solutions could produce interesting findings that account more for preventing and detecting cache side-channel attacks. The detection infrastructure solution may take other characteristics into account to improve its efficiency and ability to detect other types of cache side-channel attacks. This research has thrown up many questions in need of further investigation. It is recommended that further research be undertaken in the following areas:

1. The proposed model needs to be implemented and evaluated in a real cloud computing environment.

2. More research is needed to better understand how the proposed model can be used in mobile devices.
3. A good quality policy is needed to define allowed roles and tasks which have to be assigned to each role.
4. It would be interesting to assess the effects of giving security classification to processes and how that may affect the system performance.
5. Further work needs to be done to establish whether the security labels can be enforced in the lower level layers such as CPU caches.
6. Further research regarding the role of risk engine would be worthwhile. particularly, how risk of accessing data and available resources can be calculated with taking into account:
    - The movement of data from one datacentre or place to another.
    - Multiple roles for the same user.
    - What kind of actions should be used to credit users' behaviours.
    - Dealing with heterogeneity.
7. More broadly, research is also needed to determine how access control models can be linked to mechanisms used by operating systems or CPUs to control access to data in lower levels such as CPU caches and RAM.
8. The novel lightweight solution to prevent cache side-channel attacks needs to be extended to cover other types of cache side-channel attacks such as time and trace driven side-channel attacks.
9. It would be interesting to assess the effects of increasing the size of cache size and levels.
10. The novel infrastructure solution may need to be developed for detecting more types of cache side-channel attacks.

Bibliography

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," *NIST special publication*. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf. 2011.

[2] J. Bardin, J. Callas, S. Chaput, and P. Fusco, "Security guidance for critical areas of focus in cloud computing," *Cloud Security Alliance*, 2009. [Online]. Available: https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf. 2009.

[3] M. Merabti, M. Kennedy, and W. Hurst, "Critical infrastructure protection: A 21 st century challenge," in *International Conference on Communications and Information Technology (ICCIT)*, pp. 1–6, 2011.

[4] L. Billings, "Turkey power cut," *The Independent*. [Online]. Available: http://www.independent.co.uk/news/world/nationwide-blackout-throws-turkey-into-chaos-10145504.html. 31-Mar-2015.

[5] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *J. Netw. Comput. Appl.*, vol. 34, no. 1, pp. 1–11, Jan. 2011.

[6] S. Lar, X. Liao, and S. Abbas, "Cloud computing privacy & security global issues, challenges, & mechanisms," in *The 6th International ICST Conference on Communications and Networking*, pp. 1240–1245, 2011.

[7] F. Lombardi and R. Di Pietro, "Secure virtualization for cloud computing," *J. Netw. Comput. Appl.*, vol. 34, no. 4, pp. 1113–1122, Jul. 2011.

[8] Z. Wang, "Security and Privacy Issues within the Cloud Computing," in *The International Conference on Computational and Information Sciences*, pp. 175–178 2011.

[9] W. Dan Hubbard and Z. Michael Sutton, "Top Threats to Cloud Computing V1.0," *Cloud Security Alliance*. [Online]. Available: https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf. 2010.

[10] K. Popovic and Z. Hocenski, "Cloud computing security issues and challenges," in *The 33rd International Convention (MIPRO)*, pp. 344–349, 2010.

[11] R. Masood and M. A. Shibli, "Comparative Analysis of Access Control Systems on Cloud," in *The 13th ACIS International Conference on Software Engineering, Artificial*

Intelligence, Networking and Parallel/Distributed Computing, pp. 41–46, 2012.

[12] H. Takabi, J. Joshi, and G. Ahn, “Security and privacy challenges in cloud computing environments,” *IEEE Secur. Priv.*, vol. 8, no. 6, pp. 24–31, 2010.

[13] W. Wang, J. Han, M. Song, and X. Wang, “The design of a trust and role based access control model in cloud computing,” in *The 6th International Conference on Pervasive Computing and Applications*, pp. 330–334, 2011.

[14] Y. Zhou and D. Feng, “Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing.,” *IACR Cryptology ePrint Archive*. [Online]. Available: http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-3/physec/papers/physecpaper19.pdf. 2005.

[15] O. Acıiçmez, W. Schindler, and Ç. Koç, “Cache based remote timing attack on the AES,” *Cryptogr. Track RSA Conf.*, vol. 4377, pp. 271–286, 2006.

[16] O. Acıiçmez and W. Schindler, “A Vulnerability in RSA Implementations Due to Instruction Cache Analysis and Its Demonstration on OpenSSL,” in *The Cryptopgraphers’ Track at the RSA conference on Topics in cryptology (CT-RSA'08)*, pp. 256–273, 2008.

[17] T. Kim, M. Peinado, and G. Mainar-Ruiz, “Stealthmem: system-level protection against cache-based side channel attacks in the cloud,” in *The 21st USENIX conference on Security symposium (Security’12)*, p. 16, 2012.

[18] M. Godfrey and M. Zulkernine, “A Server-Side Solution to Cache-Based Side-Channel Attacks in the Cloud,” in *The IEEE Sixth International Conference on Cloud Computing*, pp. 163–170, 2013.

[19] Y. Zhang and M. Reiter, “Düppel: retrofitting commodity operating systems to mitigate cache side channels in the cloud,” in *The ACM SIGSAC conference on Computer & communications security (CCS ’13)*, p. 11, 2013.

[20] Y. Zhang, A. Juels, A. Oprea, and M. K. Reiter, “HomeAlone: Co-residency Detection in the Cloud via Side-Channel Analysis,” in *The IEEE Symposium on Security and Privacy*, pp. 313–328, 2011.

[21] S. Yu, X. Gui, and J. Lin, “An approach with two-stage mode to detect cache-based side channel attacks,” in *The International Conference on Information Networking (ICOIN)*, pp. 186–191, 2013.

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

[23] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: state-of-the-art and research challenges,” *J. Internet Serv. Appl.*, vol. 1, no. 1, pp. 7–18, Apr. 2010.

[24] S. Berman and L. Kesterson-Townes, “The power of cloud. Driving business model innovation,” *IBM Institute for Business Value*. [Online]. Available: http://public.dhe.ibm.com/common/ssi/ecm/en/gbe03470usen/GBE03470USEN.PDF. 2012.

[25] K. Weins, “Cloud Computing Trends: 2014 State of the Cloud Survey,”. [Online]. Available: http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2014-state-cloud-survey, 2014.

[26] D. du Preez, “IBM and Cable & Wireless to gather smart meter data in the cloud,” *Computing.Co.UK*. [Online]. Available: http://www.computing.co.uk/ctg/news/2035755/ibm-cable-wireless-gather-smart-meter-cloud. 2011.

[27] CBR Staff Writer, “BT adds new cloud-based solution to supply chain solution portfolio,” *Cloud Platform*,. [Online]. Available: http://cloudplatforms.cbronline.com/news/bt-adds-new-cloud-based-solution-to-supply-chain-solution-portfolio-241012. 2012.

[28] P. Danny, “Green light for National Grid’s cloud move,” *Computing.Co.UK*. [Online]. Available: http://www.computing.co.uk/ctg/analysis/2257295/green-light-for-national-grid-s-cloud-move. 2013.

[29] C. Wang, Q. Wang, K. Ren, and W. Lou, “Ensuring data storage security in Cloud Computing,” in *The17th International Workshop on Quality of Service*, pp. 1–9, 2009.

[30] R. Anderson, “Security Engineering: A guide to building dependable distributed systems,”. [Online]. Available: http://books.google.com/books?hl=en&lr=&id=eo4Otm_TcW8C&oi=fnd&pg=PT12&dq=Security+Engineering:+A+guide+to+building+dependable+distributed+systems&ots=gzIICLbB66&sig=1cwYaxE_fbgFQsQx6qpDdgD_o6c. 2010.

[31] Z. Wan, J. Liu, and R. Deng, “HASBE: A Hierarchical Attribute-Based Solution for Flexible and Scalable Access Control in Cloud Computing,” *IEEE Trans. Inf. Forensics Secur.*, vol. 7, no. 2, pp. 743–754, 2012.

[32] Z. Tianyi, L. Weidong, and S. Jiaxing, “An Efficient Role Based Access Control System

for Cloud Computing," in *The 11th IEEE International Conference on Computer and Information Technology*, pp. 97–102, 2011.

[33] V. C. Hu and K. Scarfone, "Guidelines for Access Control System Evaluation Metrics," *National Institute of Standards and Technology*. [Online]. Available: http://csrc.nist.gov/publications/nistir/ir7874/nistir7874.pdf. 2012.

[34] S. Harris, "Mike Meyers' CISSP(R) Certification Passport,". [Online]. Available: https://books.google.com/books/about/Mike_Meyers_CISSP_R_Certification_Passpo.html?id=Vp3MEDK0E7sC. 2002.

[35] R. Ausanka-Crues, "Methods for Access Control: Advances and Limitations," *Harvey Mudd College*. [Online]. Available: http://www.cs.hmc.edu/~mike/public_html/courses/security/s06/projects/ryan.pdf. 2004.

[36] D. Bell and L. LaPadula, "Secure computer systems: Mathematical foundations," *bedford, MA*. [Online]. Available: http://www.albany.edu/acc/courses/ia/classics/belllapadula1.pdf. 1973.

[37] K. Biba, "Integrity considerations for secure computer systems,". [Online]. Available: http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA039324. 1977.

[38] B. W. Lampson, "Protection," in *The Fifth Princeton Symposium on Information Sciences and Systems*, pp. 18–24, 1974.

[39] P. Samarati and S. de Vimercati, "Access control: Policies, models, and mechanisms," *Foundations of Security Analysis and Design*. [Online]. Available: http://www.springerlink.com/index/80wrewj7j1a716wb.pdf. 2001.

[40] B. Laurie, "Access Control (v0. 1),". [Online]. Available: http://www.links.org/files/capabilities.pdf. 2009.

[41] S. Oh and S. Park, "Task–role-based access control model," *Inf. Syst.*, vol. 28, no. 2002, pp. 533–562, 2003.

[42] V. Suhendra, "A Survey on Access Control Deployment," *Secur. Technol.*, vol. 259, pp. 11–20, 2011.

[43] Q. Munawer, "Administrative models for role-based access control,". [Online]. Available:

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.9150&rep=rep1&type=pdf. 2000.

[44] M. a. Al-Kahtani and R. Sandhu, "A model for attribute-based user-role assignment," in *The 18th Annual Computer Security Applications Conference*, pp. 353–362, 2002.

[45] A. Karp, H. Haury, and M. Davis, "From abac to zbac: The evolution of access control models," *HP Laboratories-2009-30*. [Online]. Available: http://www.hpl.hp.com/techreports/2009/HPL-2009-30.pdf?jumpid=reg_R1002_USEN. 2009.

[46] X. Jin, R. Krishnan, and R. Sandhu, "A unified attribute-based access control model covering DAC, MAC and RBAC," in *The 26th Annual IFIP WG 11.3 conference on Data and Applications Security and Privacy (DBSec'12 Proceedings)*, pp. 41–55, 2012.

[47] A. Brucker, L. Brügger, P. Kearney, and B. Wolff, "An approach to modular and testable security models of real-world health-care applications," in *The 16th ACM symposium on Access control models and technologies (SACMAT '11)*, pp. 133–142, 2011.

[48] P. Cheng and P. Rohatgi, "Fuzzy multi-level security: An experiment on quantified risk-adaptive access control," in *The IEEE Symposium on Security and Privacy (SP '07)*, pp. 222–230, 2007.

[49] L. Sun, H. Wang, J. Yong, and G. Wu, "Semantic access control for cloud computing based on e-Healthcare," in *The 16th IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 512–518, 2012.

[50] H. Andal Jayaprakash and M. Hadi Gunes, "Ensuring access control in cloud provisioned healthcare systems," in *The IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 247–251, 2011.

[51] W.-T. Tsai and Q. Shao, "Role-Based Access-Control Using Reference Ontology in Clouds," in *The Tenth International Symposium on Autonomous Decentralized Systems*, vol. 2, pp. 121–128, 2011.

[52] E. E. Mon and T. T. Naing, "The privacy-aware access control system using attribute- and role-based access control in private cloud," in *The 4th IEEE International Conference on Broadband Network and Multimedia Technology*, pp. 447–451, 2011.

[53] J. Hu, L. Chen, Y. Wang, and S.-H. Chen, "Data Security Access Control Model of Cloud Computing," in *The International Conference on Computer Sciences and Applications*, pp. 29–34, 2013.

[54] C. Zhou and B. Li, "iHAC : A Hybrid Access Control Framework for IaaS Clouds," in *The 7th IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, pp. 853 – 858, 2014.

[55] L. I. N. Guoyuan, W. Danrul, B. I. E. Yuyul, and L. E. I. Min, "MTBAC : A Mutual Trust Based Access Control Model in Cloud Computing," *China Commun.*, vol. 11, no. 4, pp. 154–162, 2014.

[56] D. Ricardo, C. M. Westphall, and C. B. Westphall, "A Dynamic Risk-based Access Control Architecture for Cloud Computing," in *The IEEE Network Operations and Management Symposium (NOMS)*, pp. 1–9, 2014.

[57] Z. Wu, Z. Xu, and H. Wang, "Whispers in the hyper-space: High-speed covert channel attacks in the cloud," in *The 21st USENIX conference on Security symposium (Security'12)*, pp. 1–9, 2012.

[58] I. Blake, G. Seroussi, and N. Smart, "Advances in elliptic curve cryptography,". [Online]. Available: http://books.google.com/books?hl=en&lr=&id=E3hVu5ZjbxQC&oi=fnd&pg=PR9&dq=Advances+in+elliptic+curve+cryptography&ots=GzQfOPw9AA&sig=h2kSit5rq0hzA1LHMTcN6WfIlcU. 2005.

[59] R. Anderson, M. Bond, J. Clulow, and S. Skorobogatov, "Cryptographic Processors-A Survey," *Proc. IEEE*, vol. 94, no. 2, pp. 357–369, Feb. 2006.

[60] M. Kowarschik and C. Weiß, "An overview of cache optimization techniques and cache-aware numerical algorithms," *Algorithms Mem. Hierarchies*, vol. 2625, pp. 213–232, 2003.

[61] Intel, "An Overview of Cache," *Intel*. [Online]. Available: http://download.intel.com/design/intarch/papers/cache6.pdf. 2007.

[62] O. Acıiçmez, B. Brumley, and P. Grabher, "New results on instruction cache attacks," in *The 12th international conference on Cryptographic hardware and embedded systems (CHES'10)*, vol. 216499, pp. 110–124, 2010.

[63] O. Acrimez and C. Koc, "Trace-driven cache attacks on AES," in *The 8th international conference on Information and Communications Security (ICICS'06)*, pp. 112–121, 2006.

[64] E. Tromer, D. A. Osvik, and A. Shamir, "Efficient Cache Attacks on AES, and Countermeasures," *J. Cryptol.*, vol. 23, no. 1, pp. 37–71, Jul. 2009.

[65] O. Aciiçmez, Ç. Koç, and J. Seifert, "On the power of simple branch prediction analysis," in *ASIACCS '07 Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pp. 312–320, 2007.

[66] O. Aciicmez and J.-P. Seifert, "Cheap Hardware Parallelism Implies Cheap Security," in *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2007)*, pp. 80–91, 2007.

[67] Y. Yarom and K. Falkner, "Flush + Reload : a High Resolution , Low Noise , L3 Cache Side-Channel Attack," *Cryptology ePrint Archive*. [Online]. Available: http://eprint.iacr.org/. 2013.

[68] D. Osvik, A. Shamir, and E. Tromer, "Cache attacks and countermeasures: the case of AES," in *The Cryptographers' Track at the RSA Conference*, pp. 1–25, 2006.

[69] F. Liu and R. B. Lee, "Security testing of a secure cache design," in *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy - HASP '13*, pp. 1–8, 2013.

[70] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *The Cryptographic Hardware and Embedded Systems (CHES 2001)*, vol. 2162, pp. 251–261, 2001.

[71] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-VM side channels and their use to extract private keys," in *The 2012 ACM conference on Computer and communications security (CCS '12)*, pp. 305–316, 2012.

[72] T. Ristenpart and E. Tromer, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *The 16th ACM conference on Computer and communications security (CCS '09)*, pp. 199–212, 2009.

[73] A. Aviram, S. Hu, B. Ford, and R. Gummadi, "Determinating timing channels in compute clouds," in *The ACM workshop on Cloud computing security workshop (CCSW '10)*, pp. 103-108, 2010.

[74] Y. Xu, M. Bailey, F. Jahanian, K. Joshi, M. Hiltunen, and R. Schlichting, "An exploration of L2 cache covert channels in virtualized environments," in *The 3rd ACM workshop on Cloud computing security workshop (CCSW '11)*, pp. 29-40, 2011.

[75] D. Page, "Partitioned Cache Architecture as a Side-Channel Defence Mechanism," *IACR Cryptology ePrint Archive*. [Online]. Available: https://eprint.iacr.org/2005/280.pdf. 2005.

[76] Z. Wang and R. B. Lee, “New cache designs for thwarting software cache-based side channel attacks,” in *The 34th annual international symposium on Computer architecture - ISCA ’07*, pp. 494-505, 2007.

[77] R. B. Lee, “A novel cache architecture with enhanced performance and security,” in *The 41st IEEE/ACM International Symposium on Microarchitecture*, pp. 83–93, 2008.

[78] L. Domnitser, A. Jaleel, J. Loew, N. Abu-Ghazaleh, and D. Ponomarev, “Non-monopolizable caches,” *ACM Trans. Archit. Code Optim.*, vol. 8, no. 4, pp. 1–21, Jan. 2012.

[79] J. Kong and O. Aciicmez, “Architecting against software cache-based side-channel attacks,” *IEEE Trans. Comput.*, vol. 62, no. 7, pp. 1276–1288, 2013.

[80] D. Bernstein, “OProfile overhead,”. [Online]. Available: http://oprofile.sourceforge.net/performance/. 2014.

[81] B. Stone and A. Vance, “Companies Slowly Join Cloud-Computing,” *New York Times*. [Online]. Available: http://www.nytimes.com/2010/04/19/technology/19cloud.html?_r=0. 18-Apr-2010.

[82] W. Group, “Guidelines for Smart Grid Cyber Security: Vol. 1, Smart Grid Cyber Security Strategy, Architecture, and High-Level Requirements,” *National Institute of Standards and Technology*. [Online]. Available: http://csrc.nist.gov/publications/nistir/ir7628/nistir-7628_vol1.pdf. 2010.

[83] S. Rani and A. Gangal, “Security Issues of Banking Adopting the Application of Cloud Computing,” *Int. J. Inf. Technol.*, vol. 5, no. 2, pp. 243–246, 2012.

[84] M. Mujinga and B. Chipangura, “Cloud computing concerns in developing economies,” in *The 9th Australian Information Security Management Conference*, pp. 47-55, 2011.

[85] E. Bezerra, “Critical infrastructure protection in Brazil: threat identification and analysis,” in *The 4th international conference on Critical information infrastructures security*, pp.14-21, 2009.

[86] A. Sharma, “Data Management and Deployment of Cloud Applications in Financial Institutions and its Adoption Challenges,” *Int. J. Sci. Technol. Res.*, vol. 1, no. 1, pp. 1–7, 2012.

[87] S. B. Andras Vajda, “Cloud Computing and Telecommunications: Business Opportunities, Technologies and Experimental Setup,” in *The World*

Telecommunications Congress (WTC), pp. 1–6, 2012.

[88] A. Almutairi, M. Sarfraz, and S. Basalamah, “A Distributed Access Control Architecture for Cloud Computing,” *IEEE Software*, vol. 29, no. 2, pp. 36–44, 2012.

[89] D. F. Ferraiolo, J. F. Barkley, and D. R. Kuhn, “A role-based access control model and reference implementation within a corporate intranet,” *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 1, pp. 34–64, Feb. 1999.

[90] K. Hasebe, M. Mabuchi, and A. Matsushita, “Capability-based delegation model in RBAC,” in *The 15th ACM symposium on Access control models and technologies (SACMAT ’10),* pp. 109–118, 2010.

[91] R. T. Simon and M. E. Zurko, “Separation of Duty in Role-Based Environments,” in *The 10th Computer Security Foundations Workshop,* pp. 183–194, 1997.

[92] A. Schaad, P. Spadone, and H. Weichsel, “A case study of separation of duty properties in the context of the Austrian eLaw process.,” in *the ACM symposium on Applied computing SAC ’05*, pp. 1328–1332, 2005.

[93] R. Sandhu, D. Ferraiolo, and R. Kuhn, “The NIST model for role-based access control: towards a unified standard,” in *The fifth ACM workshop on Role-based access control (RBAC ’00),* pp. 47–63, 2000.

[94] S. Hwang, Y. Chen, and Y. Tang, “Web Services and Role Selection in Support of Separation of Duties and Binding of Duties for Composable Process Execution,” *A J. Theory Ordered Sets Its Appl.*, vol. 5, no. c, 2009.

[95] S. Crago, K. Dunn, P. Eads, L. Hochstein, D.-I. Kang, M. Kang, D. Modium, K. Singh, J. Suh, and J. P. Walters, “Heterogeneous Cloud Computing,” in *The IEEE International Conference on Cluster Computing,* pp. 378–385, 2011.

[96] V. C. Hu, D. R. Kuhn, and D. F. Ferraiolo, “The Computational Complexity of Enforceability Validation for Generic Access Control Rules,” in *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing -Vol 1 (SUTC’06),* vol. 1, pp. 260–267, 2006.

[97] M. Dalton, C. Kozyrakis, and N. Zeldovich, “Nemesis: preventing authentication & access control vulnerabilities in web applications,” in *The 18th conference on USENIX security symposium (SSYM’09),* pp. 267–282, 2009.

[98] V. Patil, A. Mei, and L. Mancini, “Addressing interoperability issues in access control

models," in *The 2nd ACM symposium on Information, computer and communications security (ASIACCS '07)*, vol. 389–391, 2007.

[99] A. D. Keromytis and J. M. Smith, "Requirements for scalable access control and security management architectures," *ACM Trans. Internet Technol.*, vol. 7, no. 2, p. 22, May 2007.

[100] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman, "Role-based access control models," *Computer (Long. Beach. Calif).*, vol. 29, no. 2, pp. 38–47, 1996.

[101] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing," in *The IEEE INFOCOM*, pp. 1–9, 2010.

[102] Z. Iqbal and J. Noll, "Towards Semantic-Enhanced Attribute-Based Access Control for Cloud Services," in *The 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 1223–1230, 2012.

[103] W. Li and H. Wan, "A Refined RBAC Model for Cloud Computing," in *The 11th IEEE/ACIS International Conference on Computer and Information Science*, pp. 43–48, 2012.

[104] Z. Tan, Z. Tang, R. Li, A. Sallam, and L. Yang, "Research on trust-based access control model in cloud computing," in *The 6th IEEE Joint International Information Technology and Artificial Intelligence Conference*, pp. 339–344, 2011.

[105] D. Chisnall, "The definitive guide to the xen hypervisor,". [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:The+Definitive+Guide+to+the+Xen+Hypervisor#0. 2008.

[106] Amazon, "Amazon Elastic Compute Cloud,". [Online]. Available: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-ug.pdf. 2014.

[107] "Virtualization Spectrum,". [Online]. Available: http://wiki.xen.org/wiki/Virtualization_Spectrum#Xen_and_paravirtualization. 2014.

[108] J. Rutkowska and R. Wojtczuk, "Qubes OS architecture," *Invisible Things Lab Tech Rep*,. [Online]. Available: https://www.grounation.org/public/Security/Qubes/arch-spec-0.3.pdf. 2010.

[109] "Credit Scheduler,". [Online]. Available: http://wiki.xen.org/wiki/Credit_Scheduler. 2013.

[110] Intel, “Intel® 64 and IA-32 Architectures Software Developer’s Manual Volume 2 (2A, 2B & 2C): Instruction Set Reference, A-Z,”. [Online]. Available: http://www.intel.co.uk/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-instruction-set-reference-manual-325383.pdf. 2014.

[111] Xen, “Credit Scheduler,” *Xen.org*,. [Online]. Available: http://wiki.xen.org/wiki/Credit_Scheduler. 2015.

Appendix A: The source code of the novel lightweight solution for preventing cache side channel attacks in cloud computing

1. The trace function

```
static inline void vcpu_runstate_change(
  struct vcpu *v, int new_state, s_time_t new_entry_time)
{
  struct vcpu *v1=v;
  struct p2m_domain *p2m;
  struct page_info *page, *pg;
  void *p;
  int k,j;
  unsigned long  l;
  s_time_t delta;
  ASSERT(v->runstate.state != new_state);
  ASSERT(spin_is_locked(per_cpu(schedule_data,v->processor).schedule_lock));
  vcpu_urgent_count_update(v);
  trace_runstate_change(v, new_state);
  if  (v->domain->domain_id!=v1->domain->domain_id  ||  v->processor  !=  v1-
>processor)
      for (k = 0; k < MAX_NESTEDP2M; k++)
         {
               p2m = v1->domain->arch.nested_p2m[k];
               while ( (pg = page_list_read1(&p2m->pages)) )
               {
           asm volatile ( "clflush (%0)" : : "r" (&pg) );
               p = __map_domain_page(pg);
               for (l = 0; l < PAGE_SIZE; l ++)
               asm volatile("clflush (%0)" :: "r" (p + l));
               }
               for (j = 0; j < sizeof(p2m); j++)
               {
                     page=__virt_to_page(&p2m[j]);
                     //for (l = 0; l < PAGE_SIZE; l ++) /*4096*/
                     //asm volatile("clflush (%0)" :: "r" (page + l));
                     asm volatile ( "clflush (%0)" : : "r" (&page));
                     asm volatile ( "clflush (%0)" : : "r" (&p2m[j]));
               //asm volatile ( "clflush %0" : : "m" (*(const char *)&p2m[j]) );
          }
```

```
        }
    delta = new_entry_time - v->runstate.state_entry_time;
    if ( delta > 0 )
    {
        v->runstate.time[v->runstate.state] += delta;
        v->runstate.state_entry_time = new_entry_time;
    }
    v->runstate.state = new_state;
}
```

2. The erasing function

```
static inline unsigned long long rdtsc12(void)
{
  unsigned long hi, lo;
  asm volatile ("rdtsc" : "=a"(lo), "=d"(hi) : : "ebx", "ecx");
  return ((unsigned long long)lo) | (((unsigned long long)hi)<<32) ;
}

static inline void
page_read(struct page_info *page, struct page_list_head *head)
{
    struct page_info *next = pdx_to_page(page->list.next);
    struct page_info *prev = pdx_to_page(page->list.prev);

        next->list.prev = page->list.prev;
        prev->list.next = page->list.next;

}

static inline struct page_info *
page_list_read(struct page_list_head *head)
{
    struct page_info *page = head->next;

    if ( page )
        page_read(page, head);

    return page;
}

static inline void the_flush_function_with_D(struct domain *d)
{
    struct p2m_domain *p2m;
    struct page_info *pg;
    void *p;
    int k, j;
    unsigned long  l;

        for (k = 0; k < MAX_NESTEDP2M; k++)
             {
                       p2m = d->arch.nested_p2m[k];
                       while ( (pg = page_list_read(&p2m->pages)) )
                       {
                asm volatile ( "clflush (%0)" : : "r" (&pg) );
                       p = __map_domain_page(pg);
                       for (l = 0; l < PAGE_SIZE; l ++)
                       asm volatile("clflush (%0)" :: "r" (p + l));
                       }
                       for (j = 0; j < sizeof(p2m); j++)
                       {
                          asm volatile ( "clflush (%0)" : : "r" (&p2m[j]) );
                 //asm volatile ( "clflush %0" : : "m" (*(const char *)&p2m[j]) );

                       }
             }
}

static inline struct page_info *__virt_to_page(const void *v)
{
    unsigned long va = (unsigned long)v;

    ASSERT(va >= XEN_VIRT_START);
    ASSERT(va < DIRECTMAP_VIRT_END);
    if ( va < XEN_VIRT_END )
        va += DIRECTMAP_VIRT_START - XEN_VIRT_START + xen_phys_start;
    else
        ASSERT(va >= DIRECTMAP_VIRT_START);
    return frame_table + ((va - DIRECTMAP_VIRT_START) >> PAGE_SHIFT);
}


static inline void the_flush_function_with_V(struct vcpu *v)
{
    struct p2m_domain *p2m;
    struct page_info *page, *pg;
    int k, j;
    //unsigned long  count;
    //l1_pgentry_t *pl1e;

            //pl1e += l1_table_offset(vpt_start);
        for (k = 0; k < MAX_NESTEDP2M; k++)
             {
                       p2m = v->domain->arch.nested_p2m[k];
                       while ( (pg = page_list_read(&p2m->pages)) )
        asm volatile ( "clflush (%0)" : : "r" (&pg) );
              for (j = 0; j < sizeof(p2m); j++)
              {
                        page=__virt_to_page(&p2m[j]);
                        //for (l = 0; l < PAGE_SIZE; l ++) /*4096*/
                        //asm volatile("clflush (%0)" :: "r" (page + l));
                        asm volatile ( "clflush (%0)" : : "r" (&page));
                        asm volatile ( "clflush (%0)" : : "r" (&p2m[j]));
              //asm volatile ( "clflush %0" : : "m" (*(const char *)&p2m[j]) );
        }
     }
     /*for ( count = 0; count < v->domain->tot_pages >=; count++ )
     {
     page = mfn_to_page(l1e_get_pfn(*pl1e));
              asm volatile ( "clflush (%0)" : : "r" (&page));
     }*/

}
```

Appendix B: The source code of the novel detection infrastructure solution for detecting cache side channel attacks in cloud computing

1. **The VCPUs new data structure**

```
struct new_node
```

```
{
   int cache_miss_time;
   unsigned long start;
   unsigned long end;
   struct new_node *next=NULL;
   struct new_node *curr=NULL;
};
struct waitqueue_vcpu;
struct vcpu
{
   int           vcpu_id;
   struct new_node     *head, *curr;
   int           processor;
     /*node_younis *new_node,*head_node; the detection */
   //void *start_younis;
   vcpu_info_t     *vcpu_info;
   struct domain   *domain;
   struct vcpu     *next_in_list;
   s_time_t        periodic_period;
   s_time_t        periodic_last_event;
   struct timer     periodic_timer;
   struct timer     singleshot_timer;
   struct timer     poll_timer;    /* timeout for SCHEDOP_poll */
   void           *sched_priv;    /* scheduler-specific data */
   struct vcpu_runstate_info runstate;
#ifndef CONFIG_COMPAT
# define runstate_guest(v) ((v)->runstate_guest)
   XEN_GUEST_HANDLE(vcpu_runstate_info_t) runstate_guest; /* guest address */
#else
# define runstate_guest(v) ((v)->runstate_guest.native)
   union {
      XEN_GUEST_HANDLE(vcpu_runstate_info_t) native;
      XEN_GUEST_HANDLE(vcpu_runstate_info_compat_t) compat;
   } runstate_guest; /* guest address */
```

```
#endif
    /* last time when vCPU is scheduled out */
    uint64_t last_run_time;
    /* Has the FPU been initialised? */
    bool_t           fpu_initialised;
    /* Has the FPU been used since it was last saved? */
    bool_t           fpu_dirtied;
    /* Initialization completed for this VCPU? */
    bool_t           is_initialised;
    /* Currently running on a CPU? */
    bool_t           is_running;
    /* VCPU should wake fast (do not deep sleep the CPU). */
    bool_t           is_urgent;
#ifdef VCPU_TRAP_LAST
#define VCPU_TRAP_NONE    0
    struct {
        bool_t           pending;
        uint8_t          old_mask;
    }                async_exception_state[VCPU_TRAP_LAST];
#define async_exception_state(t) async_exception_state[(t)-1]
    uint8_t          async_exception_mask;
#endif
    /* Require shutdown to be deferred for some asynchronous operation? */
    bool_t           defer_shutdown;
    /* VCPU is paused following shutdown request (d->is_shutting_down)? */
    bool_t           paused_for_shutdown;
    /* VCPU need affinity restored */
    bool_t           affinity_broken;
    /*
     * > 0: a single port is being polled;
     * = 0: nothing is being polled (vcpu should be clear in d->poll_mask);
     * < 0: multiple ports may be being polled.
     */
    int              poll_evtchn;
```

```
 /* (over-)protected by ->domain->event_lock */
 int              pirq_evtchn_head;
 unsigned long    pause_flags;
 atomic_t         pause_count;
 /* VCPU paused for mem_event replies. */
 atomic_t         mem_event_pause_count;
 /* VCPU paused by system controller. */
 int              controller_pause_count;
 /* IRQ-safe virq_lock protects against delivering VIRQ to stale evtchn. */
 evtchn_port_t    virq_to_evtchn[NR_VIRQS];
 spinlock_t       virq_lock;
 /* Bitmask of CPUs on which this VCPU may run. */
 cpumask_var_t    cpu_hard_affinity;
 /* Used to change affinity temporarily. */
 cpumask_var_t    cpu_hard_affinity_tmp;
 /* Used to restore affinity across S3. */
 cpumask_var_t    cpu_hard_affinity_saved;
 /* Bitmask of CPUs on which this VCPU prefers to run. */
 cpumask_var_t    cpu_soft_affinity;
 /* Bitmask of CPUs which are holding onto this VCPU's state. */
 cpumask_var_t    vcpu_dirty_cpumask;
 /* Tasklet for continue_hypercall_on_cpu(). */
 struct tasklet   continue_hypercall_tasklet;
 /* Multicall information. */
 struct mc_state  mc_state;
 struct waitqueue_vcpu *waitqueue_vcpu;
 /* Guest-specified relocation of vcpu_info. */
 unsigned long vcpu_info_mfn;
 struct evtchn_fifo_vcpu *evtchn_fifo;
struct arch_vcpu arch;
};
```

2. The measuring and analysing functions

```
void the_measuring(struct vcpu *v)/*injected in the fetch step*/
{
    unsigned long str, r;
    str=rdtsc13();
    emulate_op(regs);
    r=rdtsc13();
    if ((r-str)>120) /* 120 is the maximum time to access data from CPU caches/*create
a linked list for the cache misses*/
    {
            if(v->head==NULL)
            {
            struct new_node *ptr = (struct new_node*)kmalloc(sizeof(struct new_node));
                        ptr->cache_miss_time=(r-str);
                        ptr->start=str;
                        ptr->end=r;
                        ptr->next=NULL;
                        v->head=v->curr=ptr;
            }
            else
            {
            struct new_node *ptr = (struct new_node*)kmalloc(sizeof(struct new_node));
                        ptr->cache_miss_time=(r-str);
                        ptr->start=str;
                        ptr->end=r;
                        ptr->next=NULL;
                        v->curr=ptr;
            }
}

void the_analysis(struct vcpu *v)/*injected where the VCPU relese the control of
PCPU*/
{
   int sum=0, s=0, readings=0;
   unsigned long r=0;
```

```
  double sum_devation=0.0, standard_deviation=0.0, percentage=0.0;
  struct new_node *ptr =v->head;
  struct new_node *head = NULL;
  struct new_node *curr = NULL;
  struct new_node *ptr1 = NULL;
  struct new_node *prev = NULL;
  struct new_node *del = NULL;
/************count cache misses***************************************/
  while(ptr != NULL)
  {
        if(ptr->cache_miss_time>120)
        {
                s++;
                if (r==0)
                r=ptr->end;
                else
                {
                        if(head==NULL)
                        {
                struct   new_node   *ptr1   =   (struct   new_node*)kmalloc(sizeof(struct
new_node));
                        ptr1->cache_miss_time=ptr->start-r;
                        ptr1->next=NULL;
                        head=curr=ptr1;
                        r=ptr->end;
                        }
                        else
                        {
                struct   new_node   *ptr1   =   (struct   new_node*)kmalloc(sizeof(struct
new_node));
                        ptr1->cache_miss_time=ptr->start-r;
                        ptr1->next=NULL;
                        curr>next=ptr1;
                        curr=ptr1;
```

```
                                r=ptr->end;
                                }
                        }
        }
        ptr = ptr->next;
           readings++;
   }
/************************************************************/
   percentage=(s/readings)*100;
   if (percentage>60.41) /* 60.41 is the average of CPU cache misses happened in a
normal execution/
    {
           ptr1=head;
        /*calculating the sumation of cache misses times*/
           while(ptr1 != NULL)
           {
                    sum=sum+ptr1->cache_miss_time;
                    ptr1 = ptr1->next;
           }
        mean=sum/s;
           ptr1=head;
        /*calculating the sum of deviation of cache misses times*/
           while(ptr1 != NULL)
           {
                    sum_devation=sum_devation+((ptr1->cache_miss_time-mean)*(ptr1-
>cache_miss_time-mean));
                    ptr1 = ptr1->next;
           }
        /*calculating the standard deviation of cache misses times*/
           standard_deviation=sqrt(sum_devation/s);
           if (standard_deviation<200.314)/*9,500 to 20,000 cycles*/
           printk(" The VCPU %d which uses by Domain %d lunched CSSA\n", v-
>vcpu_id, v->domain->domain_id);
    }
```

```
/*************empty the list after the VCPU release the control of PCPU*******/
   ptr1=v->head;
   while(ptr1 != NULL)
   {
      del = ptr1;
      if(prev != NULL)
         prev->next = del->next;
      if(del == v->curr)
      {
         v->curr = prev;
      }
      else if(del == v->head)
      {
         v->head = del->next;
      }
      free(del);
         del = NULL;
      ptr1 = ptr1->next;

   }
}
```

Appendix C: The new Prime and Probe cache side channel attack source code

1. **A function for getting virtual addresses**

```
static inline void VA_D(struct domain *d)
{
   struct p2m_domain *p2m;
   struct page_info *pg;
   void *p;
   int k, j;
   unsigned long  l;
```

```
        for (k = 0; k < MAX_NESTEDP2M; k++)
            {
                    p2m = d->arch.nested_p2m[k];
                    while ( (pg = page_list_read(&p2m->pages)) )
                    {
                       printk ( "%p”, (&pg) );
                       p = __map_domain_page(pg);
                      for (l = 0; l < PAGE_SIZE; l ++)
                       printk("%p", &(p + l));
                    }
                    for (j = 0; j < sizeof(p2m); j++)
                    {
                      printk("%p", (&p2m[j]) );
                    }
            }
    }
```

2. **The translation of physical addresses from virtual addresses function**

```
uint32_t trans_VA_to_PA(struct vcpu *v, struct p2m_domain *p2m,
                unsigned long va, walk_t *gw,
                uint32_t pfec, mfn_t top_mfn, void *top_map)
{
    struct domain *d = v->domain;
    p2m_type_t p2mt;
    guest_l1e_t *l1p = NULL;
    guest_l2e_t *l2p = NULL;
#if GUEST_PAGING_LEVELS >= 4 /* 64-bit only... */
    guest_l3e_t *l3p = NULL;
    guest_l4e_t *l4p;
#endif
    uint32_t gflags, mflags, iflags, rc = 0;
    bool_t smep = 0, smap = 0;
    bool_t pse1G = 0, pse2M = 0;
    p2m_query_t qt = P2M_ALLOC | P2M_UNSHARE;

    perfc_incr(guest_walk);
    memset(gw, 0, sizeof(*gw));
    gw->va = va;

    /* Mandatory bits that must be set in every entry.  We invert NX and
     * the invalid bits, to calculate as if there were an "X" bit that
     * allowed access.  We will accumulate, in rc, the set of flags that
     * are missing/unwanted. */
    mflags = mandatory_flags(v, pfec);
    iflags = (_PAGE_NX_BIT | _PAGE_INVALID_BITS);

    if ( is_hvm_vcpu(v) && !(pfec & PFEC_user_mode) )
    {
        struct segment_register seg;
        const struct cpu_user_regs *regs = guest_cpu_user_regs();

        /* SMEP: kernel-mode instruction fetches from user-mode mappings
         * should fault.  Unlike NX or invalid bits, we're looking for _all_
         * entries in the walk to have _PAGE_USER set, so we need to do the
         * whole walk as if it were a user-mode one and then invert the answer. */
        smep =  hvm_smep_enabled(v) && (pfec & PFEC_insn_fetch);

        switch ( v->arch.smap_check_policy )
        {
        case SMAP_CHECK_HONOR_CPL_AC:
            hvm_get_segment_register(v, x86_seg_ss, &seg);

            /*
             * SMAP: kernel-mode data accesses from user-mode mappings
             * should fault.
             * A fault is considered as a SMAP violation if the following
             * conditions come true:
             *   - X86_CR4_SMAP is set in CR4
             *   - A user page is accessed
             *   - CPL = 3 or X86_EFLAGS_AC is clear
             *   - Page fault in kernel mode
             */
            smap = hvm_smap_enabled(v) &&
                   ((seg.attr.fields.dpl == 3) ||
                    !(regs->eflags & X86_EFLAGS_AC));
            break;
        case SMAP_CHECK_ENABLED:
            smap = hvm_smap_enabled(v);
            break;
        default:
            ASSERT(v->arch.smap_check_policy == SMAP_CHECK_DISABLED);
            break;
        }
    }

    if ( smep || smap )
        mflags |= _PAGE_USER;

#if GUEST_PAGING_LEVELS >= 3 /* PAE or 64... */
#if GUEST_PAGING_LEVELS >= 4 /* 64-bit only... */

    /* Get the l4e from the top level table and check its flags*/
    gw->l4mfn = top_mfn;
    l4p = (guest_l4e_t *) top_map;
    gw->l4e = l4p[guest_l4_table_offset(va)];
    gflags = guest_l4e_get_flags(gw->l4e) ^ iflags;
    if ( !(gflags & _PAGE_PRESENT) ) {
        rc |= _PAGE_PRESENT;
        goto out;
```

```
}
rc |= ((gflags & mflags) ^ mflags);

/* Map the l3 table */
l3p = map_domain_gfn(p2m,
                     guest_l4e_get_gfn(gw->l4e),
                     &gw->l3mfn,
                     &p2mt,
                     qt,
                     &rc);
if(l3p == NULL)
    goto out;
/* Get the l3e and check its flags*/
gw->l3e = l3p[guest_l3_table_offset(va)];
gflags = guest_l3e_get_flags(gw->l3e) ^ iflags;
if ( !(gflags & _PAGE_PRESENT) ) {
    rc |= _PAGE_PRESENT;
    goto out;
}
rc |= ((gflags & mflags) ^ mflags);

pse1G = (gflags & _PAGE_PSE) && guest_supports_1G_superpages(v);

if ( pse1G )
{
    /* Generate a fake l1 table entry so callers don't all
     * have to understand superpages. */
    gfn_t start = guest_l3e_get_gfn(gw->l3e);
    /* Grant full access in the l1e, since all the guest entry's
     * access controls are enforced in the l3e. */
    int flags = (_PAGE_PRESENT|_PAGE_USER|_PAGE_RW|
                 _PAGE_ACCESSED|_PAGE_DIRTY);
    /* Import cache-control bits. Note that _PAGE_PAT is actually
     * _PAGE_PSE, and it is always set. We will clear it in case
     * _PAGE_PSE_PAT (bit 12, i.e. first bit of gfn) is clear. */
    flags |= (guest_l3e_get_flags(gw->l3e)
              & (_PAGE_PAT|_PAGE_PWT|_PAGE_PCD));
    if ( !(gfn_x(start) & 1) )
        /* _PAGE_PSE_PAT not set: remove _PAGE_PAT from flags. */
        flags &= ~_PAGE_PAT;

    if ( gfn_x(start) & GUEST_L3_GFN_MASK & ~0x1 )
        rc |= _PAGE_INVALID_BITS;

    /* Increment the pfn by the right number of 4k pages. */
    start = _gfn((gfn_x(start) & ~GUEST_L3_GFN_MASK) +
                 ((va >> PAGE_SHIFT) & GUEST_L3_GFN_MASK));
    gw->l1e = guest_l1e_from_gfn(start, flags);
    gw->l2mfn = gw->l1mfn = _mfn(INVALID_MFN);
    goto set_ad;
    }

#else /* PAE only... */

    /* Get the l3e and check its flag */
    gw->l3e = ((guest_l3e_t *) top_map)[guest_l3_table_offset(va)];
    if ( !(guest_l3e_get_flags(gw->l3e) & _PAGE_PRESENT) )
    {
        rc |= _PAGE_PRESENT;
        goto out;
    }

#endif /* PAE or 64... */

    /* Map the l2 table */
    l2p = map_domain_gfn(p2m,
                         guest_l3e_get_gfn(gw->l3e),
                         &gw->l2mfn,
                         &p2mt,
                         qt,
                         &rc);
    if(l2p == NULL)
        goto out;
    /* Get the l2e */
    gw->l2e = l2p[guest_l2_table_offset(va)];

#else /* 32-bit only... */

    /* Get l2e from the top level table */
    gw->l2mfn = top_mfn;
    l2p = (guest_l2e_t *) top_map;
    gw->l2e = l2p[guest_l2_table_offset(va)];

#endif /* All levels... */

    gflags = guest_l2e_get_flags(gw->l2e) ^ iflags;
    if ( !(gflags & _PAGE_PRESENT) ) {
        rc |= _PAGE_PRESENT;
        goto out;
    }
    rc |= ((gflags & mflags) ^ mflags);

    pse2M = (gflags & _PAGE_PSE) && guest_supports_superpages(v);

    if ( pse2M )
    {
        /* Special case: this guest VA is in a PSE superpage, so there's
         * no guest l1e.  We make one up so that the propagation code
         * can generate a shadow l1 table.  Start with the gfn of the
         * first 4k-page of the superpage. */
        gfn_t start = guest_l2e_get_gfn(gw->l2e);
        /* Grant full access in the l1e, since all the guest entry's
         * access controls are enforced in the shadow l2e. */
        int flags = (_PAGE_PRESENT|_PAGE_USER|_PAGE_RW|
                     _PAGE_ACCESSED|_PAGE_DIRTY);
        /* Import cache-control bits. Note that _PAGE_PAT is actually
         * _PAGE_PSE, and it is always set. We will clear it in case
         * _PAGE_PSE_PAT (bit 12, i.e. first bit of gfn) is clear. */
        flags |= (guest_l2e_get_flags(gw->l2e)
                  & (_PAGE_PAT|_PAGE_PWT|_PAGE_PCD));
        if ( !(gfn_x(start) & 1) )
            /* _PAGE_PSE_PAT not set: remove _PAGE_PAT from flags. */
            flags &= ~_PAGE_PAT;

        if ( gfn_x(start) & GUEST_L2_GFN_MASK & ~0x1 )
        {
#if GUEST_PAGING_LEVELS == 2
            /*
             * Note that _PAGE_INVALID_BITS is zero in this case, yielding a
             * no-op here.
             *
             * Architecturally, the walk should fail if bit 21 is set (others
             * aren't being checked at least in PSE36 mode), but we'll ignore
             * this here in order to avoid specifying a non-natural, non-zero
             * _PAGE_INVALID_BITS value just for that case.
             */
#endif
            rc |= _PAGE_INVALID_BITS;
        }
        /* Increment the pfn by the right number of 4k pages.
         * Mask out PAT and invalid bits. */
        start = _gfn((gfn_x(start) & ~GUEST_L2_GFN_MASK) +
                     guest_l1_table_offset(va));
        gw->l1e = guest_l1e_from_gfn(start, flags);
        gw->l1mfn = _mfn(INVALID_MFN);
    }
    else
    {
        /* Not a superpage: carry on and find the l1e. */
        l1p = map_domain_gfn(p2m,
                             guest_l2e_get_gfn(gw->l2e),
                             &gw->l1mfn,
                             &p2mt,
                             qt,
                             &rc);
        if(l1p == NULL)
            goto out;
        gw->l1e = l1p[guest_l1_table_offset(va)];
        gflags = guest_l1e_get_flags(gw->l1e) ^ iflags;
        if ( !(gflags & _PAGE_PRESENT) ) {
            rc |= _PAGE_PRESENT;
            goto out;
        }
        rc |= ((gflags & mflags) ^ mflags);
    }

#if GUEST_PAGING_LEVELS >= 4 /* 64-bit only... */
set_ad:
#endif
    /* Now re-invert the user-mode requirement for SMEP and SMAP */
    if ( smep || smap )
        rc ^= _PAGE_USER;

    /* Go back and set accessed and dirty bits only if the walk was a
     * success.  Although the PRMs say higher-level _PAGE_ACCESSED bits
     * get set whenever a lower-level PT is used, at least some hardware
     * walkers behave this way. */
    if ( rc == 0 )
    {
#if GUEST_PAGING_LEVELS == 4 /* 64-bit only... */
        if ( set_ad_bits(l4p + guest_l4_table_offset(va), &gw->l4e, 0) )
            paging_mark_dirty(d, mfn_x(gw->l4mfn));
        if ( set_ad_bits(l3p + guest_l3_table_offset(va), &gw->l3e,
                         (pse1G && (pfec & PFEC_write_access))) )
            paging_mark_dirty(d, mfn_x(gw->l3mfn));
#endif
        if ( !pse1G )
        {
            if ( set_ad_bits(l2p + guest_l2_table_offset(va), &gw->l2e,
                             (pse2M && (pfec & PFEC_write_access))) )
                paging_mark_dirty(d, mfn_x(gw->l2mfn));
            if ( !pse2M )
            {
                if ( set_ad_bits(l1p + guest_l1_table_offset(va), &gw->l1e,
                                 (pfec & PFEC_write_access)) )
                    paging_mark_dirty(d, mfn_x(gw->l1mfn));
            }
        }
    }

 out:
#if GUEST_PAGING_LEVELS == 4
    if ( l3p )
    {
        unmap_domain_page(l3p);
        put_page(mfn_to_page(mfn_x(gw->l3mfn)));
    }
#endif
#if GUEST_PAGING_LEVELS >= 3
    if ( l2p )
    {
        unmap_domain_page(l2p);
        put_page(mfn_to_page(mfn_x(gw->l2mfn)));
    }
#endif
    if ( l1p )
    {
        unmap_domain_page(l1p);
        put_page(mfn_to_page(mfn_x(gw->l1mfn)));
    }

    return rc;
}
```