PUCRS

ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

MAURO STRELOW STORCH

FULL-STACK CONFIDENTIALITY COST MODELING FOR CLOUD COMPUTING

Porto Alegre
2017

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
FACULTY OF INFORMATICS
COMPUTER SCIENCE GRADUATE PROGRAM

FULL-STACK CONFIDENTIALITY COST MODELING FOR CLOUD COMPUTING

MAURO STRELOW STORCH

Thesis submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fulfillment of the requirements for the degree of Ph. D. in Computer Science.

Advisor: Prof. César Augusto Fonticielha De Rose
Co-Advisor: Prof. Avelino Francisco Zorzo

Porto Alegre
2017

Ficha Catalográfica

S884f Storch, Mauro Strelow

Full-Stack Confidentiality Cost Modeling for Cloud Computing / Mauro Strelow Storch . – 2017.
109 p.
Tese (Doutorado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. César Augusto Fonticielha De Rose.
Co-orientador: Prof. Dr. Avelino Francisco Zorzo.

1. Security. 2. Cryptography. 3. Cost Modeling. 4. Cloud Computing. I. De Rose, César Augusto Fonticielha. II. Zorzo, Avelino Francisco. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecária responsável: Salete Maria Sartori CRB-10/1363

Mauro Strelow Storch

Full-Stack Confidentiality Cost Modeling for Cloud Computing

This Thesis has been submitted in partial fulfillment of the requirements for the degree of Doctor/Master of Computer Science, of the Graduate Program in Computer Science, School of Technology of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on August 10th, 2017.

COMMITTEE MEMBERS:

Prof. Dr. Antônio Tadeu Azevedo Gomes (LNCC)

Prof. Dr. Marinho Pilla Barcellos (PPGC/UFRGS)

Prof Dr. Tiago Coelho Ferreto (PPGCC/PUCRS)

Prof. Dr. Avelino Francisco Zorzo (PPGCC/PUCRS – Co-Advisor)

Prof. Dr. César Augusto Fonticielha De Rose (PPGCC/PUCRS - ADVISOR)

To my wife and my family.

“The present is theirs; the future, for which I really worked, is mine.”
(Nikola Tesla)

ACKNOWLEDGMENTS

This Ph.D. thesis is a result of the work developed in the last four years, and it was concluded with the support of mentors, colleagues, friends, and family. Many thanks for all the people who in some manner were part of the development of this work, especially to my advisor Professor César A. F. De Rose, for the opportunity to develop my Ph.D. and for sharing his priceless knowledge and guidance. I would also like to thank my co-advisor Professor Avelino F. Zorzo, who encouraged me and supported my work in new topics and perspectives, which added robust and valuable aspects to my research. A special thanks, too, to co-authors and colleagues Alex Orozco and Régio Michelin, who helped me improve important topics of research and also produce scientific content.

Thanks in advance to all Ph.D. committee members that who accepted being part of the thesis evaluation - Prof. Antônio Tadeu Azevedo Gomes, Prof. Marinho Barcellos, and Prof. Tiago Ferreto.

Many thanks to Ph.D. colleagues and LAD/PUC-RS members Endrigo D'Agostini Conte, Fábio Rossi, Kassiano José Matteussi, Miguel Xavier, Marcelo Neves, Rafael Lorenzo Belle, and Uillian Ludwig for sharing their experiences and thoughts about the subject and the processes to build this thesis. Thanks to all employees and professors of the Post-Graduate Program in Computer Science at PUC-RS, as well as HP Labs, CAPES, and CNPQ for the financial support during my studies. I have to mention and thank UCS and IFSul for the teaching opportunities as a lecturer where I reviewed and developed important concepts of computer science.

Finally, thanks to my family, in particular to my wife Grasi, for their patience and acceptance of my many unavailable moments, and for their inspiration to always do good. Also, many thanks to friends who shared encouragement throughout this work, and to God for the conscious freedom that encourages people to achieve a better life in this world.

MODELAGEM DO CUSTO DE CONFIDENCIALIDADE FULL-STACK PARA COMPUTAÇÃO EM NUVEM

RESUMO

A adoção de princípios de segurança em sistemas computacionais de nuvem é uma demanda crescente para diversas instituições, incluindo empresas e mesmo agências governamentais. A confidencialidade é o princípio de segurança que visa garantir acesso restrito à informações sensíveis. Esse princípio utiliza protocolos, métodos de autenticação e algoritmos de criptografia que demandam recursos computacionais, impactando tanto de usuários quanto de provedores. Pela natureza de precificação sob demanda dos provedores de nuvem, adicionar mecanismos de segurança para aplicações altera os custos totais dos serviços podendo até inviabilizar a adoção de computação em nuvem por algumas aplicações. Na intenção de mapear a adoção de confidencialidade em ambientes computacionais de nuvem, deve-se considerar a utilização de criptografia nos seus três principais eixos: (a) na comunicação em redes públicas; (b) no armazenamento de dados em serviços terceirizados; e (c) no processamento de dados em ambientes virtualizados e compartilhados. Quando aplicado nesses três eixos, o princípio de confidencialidade promove um ambiente com um maior nível de segurança permitindo que usuários usufruam dos benefícios da computação em nuvem, mesmo que possuam estritos requisitos de confidencialidade. No entanto, o custo de se adicionar confidencialidade nos serviços dos usuários em um ambiente de computação em nuvem deve ser estimado para dar suporte aos administradores tomarem decisões sobre suas aplicações. Considerando esse cenário, este trabalho propõe (i) um projeto de níveis de confidencialidade para computação em nuvem; e (ii) uma modelagem do custo do princípio de confidencialidade para os eixos comunicação, armazenamento e processamento. Considera-se também que esses eixos podem ser combinados para aumentar os níveis de segurança da aplicação do usuário. Os resultados preditos pela modelagem podem ser utilizados para redimensionar os recursos na nuvem, recalcular os custos da aplicação ou mesmo ajudar

na tomada de decisão na escolha de provedores. Na avaliação do modelo feita neste trabalho, utilizou-se um *benchmark* para simulação de um ambiente de *e-commerce* onde foi possível predizer a sobrecarga dos mecanismos de segurança, por exemplo criptografia AES e busca em banco de dados encriptados, com uma precisão próxima a 95%.

Palavras-Chave: Segurança da Informação, Criptografia, Modelagem de Custos, Computação em Nuvem.

FULL-STACK CONFIDENTIALITY COST MODELING FOR CLOUD COMPUTING

ABSTRACT

Institutes, companies, and governments have increased the adoption of security principles when using cloud computing environments. From protocols and authentication methods to cryptography algorithms, confidentiality has gained attention from both cloud users and cloud providers for, on one hand, preventing data leakage, but, on the other hand, demanding extra computational resources. Due to the nature of the on-demand billing process applied by public cloud providers, considering the pay-as-you-go model, adding security mechanisms may impact the rented resources, increasing the overall costs and minimizing the feasibility for some applications. To better understand the adoption of confidentiality in a cloud environment, users and providers have to consider applying cryptography algorithms in its three main axes: (a) communication on public networks; (b) data storage on third-party services; and (c) data processing in shared virtual environment. A full-stack confidentiality solution, considering these three axes, allows users to have the benefits of cloud computing even if they have strict confidentiality concerns. However, the costs of adding such privacy for assets in a cloud environment should be estimated, giving support to the manager making decisions about the application's availability and performance. This Ph.D. research presents (i) an architecture of full-stack confidentiality for cloud computing; and (ii) a model to estimate cryptography costs for communicating, storing, and processing in cloud computing environments. The axes can be combined to estimate users' overheads according to their security needs. The predicted values can be used for resizing cloud resources or even recalculating rental costs of cloud services. The model's evaluation presented an accuracy close to 95%. In the evaluation, we used a database-based benchmark in a cloud environment including standard cryptography algorithms, such as AES, and Querying over Encrypted Databases.

Keywords: Security, Cryptography, Cost Modeling, Cloud Computing.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

CRM – Customer Relationship Management

DBMS – Database Management Systems

EBS – Elastic Block Store

ECC – Elliptic Curve Cryptography

FHE – Full Homomorphic Encryption

HE – Homomorphic Encryption

IAAS – Infrastructure-as-a-Service

IDS – Intrusion Detection Systems

LAN – Local Area Network

LVM – Logical Volume Management

MCC – Mobile Cloud Computing

PAAS – Platform-as-a-Service

PLA – Privacy Level Agreement

SAAS – Software-as-a-Service

SGX – Software Guard Extension

SSE – Searchable Symmetric Encryption Schemes

URL – Uniform Resource Location

VM – Virtual Machine

CONTENTS

1. INTRODUCTION

Cloud computing security demands are increasing due to users moving their sensitive digital assets to public cloud providers. In order to take advantage of the vast number of benefits of Cloud Computing [Raj11], both users and providers have been motivated to add security strategies which guarantee confidentiality, integrity, and availability principles [Sta11]. Those principles, when adopted, are part of a software solution stack, and the techniques to yield these principles consist of strategies such as cryptography algorithms, secure protocols, and advanced computational solutions that combine several concepts.

The computational stack supporting Cloud Computing can be read as a complex combination of abstraction layers with the main objectives based on server consolidation, scalability, availability, easy management, and many others. Another perspective to consider is Cloud Computing as a "*hardware-based service offering computing, network, and storage capacity where: Hardware management is highly abstracted from the buyers, buyers incur infrastructure costs as variable OPEX, and infrastructure capacity is highly elastic*" [McK09].

Computational environments intending to achieve levels of protection from external risks and threats adopt security principles. The confidentiality principle [GR95] aims to guarantee data access and disclosure only by authorized entities and with strict modes. If a computational system lacks confidentiality, it may cause unauthorized data access of private information. This principle has also gained attention because its software solutions include complex password generation, authentication tokens, n-factor authentication, digital signatures, and cryptography algorithms. Solutions for confidentiality have been introduced in the software development process because Internet and cloud computing are important players for many applications, and they have a single characteristic allowing risks covered by the confidentiality principle: they are public environments. Despite its guarantees, users and providers should consider understanding the impact of confidentiality adoption in the overall system to understand the allocation costs of the user's applications, especially in public cloud providers where applications are billed by resource usage.

Cryptography algorithms are important confidentiality mechanisms present in authentication protocols, online purchases, and digital currencies. Their implementations include symmetric algorithms, where a single key is used for ciphering and deciphering data, and asymmetric algorithms, where a key-pair is defined considering one to be used for ciphering and the second for deciphering data; it is also called the *public-key schema*. Some advances in cryptography have been made to manipulate ciphered data through Homomorphic Encryption (HE) operations, where data do not need to be deciphered even during their processing phase [Gen09]. Most of the public cloud computing providers have

started to support security agreements through the adoption of these algorithms, since the architecture of a cloud environment introduced the sharing of network, storage, and processing that change security adoption, especially when the confidentiality principle is evaluated. However, although the benefits of cryptography algorithms are consolidated, one may consider calculating the costs of using such intermediate mechanisms layers to support security principles. This consideration has also challenged researchers to improve performance of new hardware and software solutions, from embedded cryptography instructions [Int06] and dedicated cryptography chips [Tru], to mathematics-based cryptography algorithms, including the so-called Elliptic Curve Cryptography (ECC) [Mil86].

There is a common sense in security adoption that makes managers follow standard market solutions. There is also a misconception about the costs of adding security principles, not only in the software development phase but, in the case of this work, in the execution environment, especially when hosted in public cloud environments where software usage is priced on-demand. In doing so, this thesis presents research focused on the confidentiality principle adoption in cloud computing environments with the aim of producing both a full-stack secure architecture for cloud environments and a modeling to estimate the costs of this security.

1.1 Scenario and Motivation

Security components composing standard solutions for cloud computing environments include the Virtual Private Networks [Amac, Gooc] and Web-based secure communication using HTTPS [FR14, NFL+14]. Besides communication, cloud providers have started to offer confidentiality for the user's data at rest through Disk Encryption [Amab, Gooa] and Object Storage Encryption [Amad] services. In addition, research on security processing has been developed using standard hash cryptography functions, homomorphic encryption techniques [DGBL+16, LWW+10, PZB11], and embedded processor instructions [Int06, Int14]. Besides these solutions deployed in the cloud computing environment, a few architectures explore assembling these security components, as well as, measuring their impact in cloud's resources.

Regarding the resulting overhead, adding security layers in a software stack impacts both resource allocation and software performance, especially for the demands of the confidentiality principle. If users aim to store data ciphered in the cloud, it is necessary to adopt cryptography in the persistence flow, adding overhead for encrypting and decrypting when accessing them. Similarly, overhead is observed when adding confidentiality in data transmission over shared networks. In this case, all transmitted data need to be encrypted before being sent and decrypted after being received at the destination host. Even if data is confidential during communication and persistence, computing over

the cloud's data needs also to be confidential in order to increase privacy levels avoiding data leakage, as in Cross-VM attacks that exploit in-memory data leakage in shared tenancies [RGVM13]. In this sense, techniques such as cryptographic-based confidential processing, querying over encrypted databases, and homomorphic encryption have been developed in order to process data without disclosing them. Such techniques also add overhead in overall cloud allocation.

Based on these three main axes and their security solution implementations, one may consider creating a full-stack confidential environment where data is protected during their entire life-cycle. To adhere to the usage of any combination of those axes, it is important to understand the security component placement and the different security levels impacting system performance in diverse ways, including data volumes, cryptography key sizes, and application behaviors. In this context, estimating overheads can help users calculate extra computational resources when adopting confidentiality in a computational environment, especially when they are using public cloud providers, where virtual resources are placed in shared tenancies.

The main motivations of this research are to design this architecture and to produce a modeling on which it is possible to predict the security overhead. The study and development of formulations considering the interference variables aim to help both users and providers to better allocate computational resources following the security establishment of markets, from the standard privacy recommendations of institutes to the complex security demands of governments and businesses.

1.2 Hypotheses and Research Questions

This Ph.D. research aims to investigate the following hypotheses (i) *it is possible to support confidentiality in all data's life-cycle in cloud computing environments* and (ii) *confidentiality costs can be modeled and used in the sizing of cloud computing environment*. The following are four research questions to be answered throughout this document and used to support the hypotheses:

1. *What are the confidentiality risks and solutions in cloud computing environments, and how is a confidential cloud created?* This question aims to leverage the study of both security risks and components present in cloud computing environments. This study will drive the placement of the confidentiality components in the cloud software stack for the creation of a full-stack confidentiality architecture for cloud computing.

2. *What is the overhead for adding confidentiality mechanisms in the cloud computing stack?* Answering this question will help to understand the overhead added by

confidentiality in the software stack when its mechanisms are placed in a cloud computing environment. It is important to evaluate the overhead and produce a detailed modeling, including the communication, storage, and processing axes.

3. *How can the overhead of combined security mechanisms for communicating, storing, and processing data in a cloud environment be estimated?* Once the security overhead is identified and measured, the answer to this question should produce a cost model to estimate the additional resources needed to apply security in cloud computing instances.

4. *How can security overhead modeling help users and providers to create confidential cloud environments?* Finally, the answer to this question will leverage the creation of full-stack cloud computing environments with the support of a cost model, which should help managers decide better allocation of sensitive applications considering cloud resource costs and performance.

1.3 Thesis Contributions

After introducing the research questions, this work will present a literature review of the security risks and solutions in cloud computing environments. In addition to that, the thesis will describe a cloud architecture, used in this work, to present the security mechanisms placement for confidentiality support in communication, storage, and processing axes. Furthermore, keeping focus on performance and security, the third contribution of the thesis is the modeling of confidentiality overhead variables that impact system performance. To the best of our knowledge there is no cost model for cloud computing environments that helps users and providers estimate confidentiality overhead. Based on this modeling and the architecture design, both the validation of formulas and their application for security overhead prediction support the hypotheses of this Ph.D. work.

The development process of this work has also produced scientific papers, some of them already published:

- *Multi-channel Secure Interconnection Design for Hybrid Clouds* [SDRZM15] is an initial instrumentation of the overhead evaluation, considering the CPU load impact added by security in the communication among cloud nodes. The contributions of this paper are a demonstration of CPU utilization by the VPN's cryptography mechanisms and an architecture for communication performance improvement.

- *Cloud Storage Cost Modeling for Cryptographic File Systems* [SDR17] presents a cost modeling for Cryptography File Systems used as a confidential persistent layer in

cloud-based environments. The modeling was validated using a Map-Reduce framework, where the confidentiality costs could be predicted using the data volumes and the persistence capacity overheads.

- *Full-Stack Confidentiality Design and Modeling for Cloud Computing*[1] presents the main work developed in this Ph.D. research, where the confidentiality principle is designed and modeled to produce a Full-Stack confidential architecture for cloud computing environments. The mathematical modeling for confidentiality overheads presented an accuracy close to 95% for the test cases.

1.4 Thesis Structure

This thesis document is organized as follow:

- *Chapter 2* presents a review of the main areas of this research. It considers the identification of security principles and its implications on cloud computing environments. It also presents a taxonomy of the security threats and solutions of the cloud. It ends with the state of the art in both the secure architectures of cloud computing and the previous work in cloud security costs estimation and modeling.

- *Chapter 3* presents the Full-Stack confidentiality architecture for cloud computing, where cryptographic components placement is discussed, considering all three phases of data life-cycle in the communication, storage, and processing axes.

- *Chapter 4* is about the first axis considered in this research: the communication among cloud nodes and its security implications. The confidentiality costs of this axis are modeled and validated, considering different cryptography algorithms adopted in tunneled communication through VPNs.

- *Chapter 5* presents the aspects of confidentiality in the data persistence flow for cloud environments. It is the second axis considered in the research, where the modeling of security overhead is designed and validated based on cryptography algorithms, data volumes, and the IO sub-system impact.

- *Chapter 6* is dedicated to describing an overview of the novel methodologies of processing data under the confidentiality premises for public clouds. By considering both Homomorphic Encryption and Querying over Encrypted Databases techniques, this chapter proposes a high-level cost modeling of confidentiality, applied in the processing axis.

[1] Submitted to a Journal and currently under review

- *Chapter 7* will then present a use case of the cost modeling proposed in this work. The use case considers an estimation of the extra computational resources demanded by the addition of the confidentiality principle, and the result will be used for the Full-Stack confidentiality evaluation.
- *Chapter 8* finally presents the main conclusions and an overall summary of the research, which includes the validation of the hypotheses based on research contributions, the answers to the research questions, and, as future works, the open issues to be explored within new scenarios and use cases.

2. STATE OF THE ART AND BACKGROUND

The increased demand for security in computer environments resulted in many new research topics over the years, especially in the last decades, considering the increased Internet adoption in many sectors, from financial institutions to personal health data processing. Computer security has been studied since the beginning of the digital age, and the security principles considered for present-day solutions are still based on consolidated themes of studies in Information Security, Communication Security, Cryptography, and many other fields. Historically speaking, from *Alan Turing's*[1] discoveries in breaking Enigma codes to recent information leakage and espionage facts revealed by Eduard Snowden [Tox14], information protection has been a de facto differential in organizations, and the security concepts are a relevant topic in almost all institutions' decisions. In some cases, the information is so important that security features are a basic requirement of software development, like in banks and health information systems [McG06].

To guide this thesis, this chapter presents a literature review of the concepts, tools, architectures, and models related to confidentiality in cloud computing. The chapter starts with a review of Security Principles, especially related to Confidentiality. Next, Cloud Computing Security aspects are described, and a taxonomy of security risks and issues is presented. This chapter ends with the state of the art of both the cloud security architectures proposed in the literature and the cost models used in estimations of security impacts in computational environments.

2.1 Background

2.1.1 Security Principles

Computer Security, according to the NIST definition, is "*the protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources*" [GR95]. Those three principles of computer security drive the development of solutions in the computing field within several perspectives, which also consider the already-consolidated cloud computing environment.

The integrity principle regards the avoidance of improper data modification, destruction, or fabrication, which includes intentional or unintentional data modification, as well as logical and physical data corruption [Sta10]. This principle aims to ensure information non-repudiation and authenticity at any time owners request them. By proper

[1] http://www.turing.org.uk/publications/dnb.html

authorization of data access, and with the support of mechanisms for data verification, such as Hash Algorithms, it is possible to offer this principle in most computing systems.

Besides integrity, whenever data are requested, they should be delivered properly without any interruption at any time; otherwise, this would lead to a loss of availability. In the cloud computing context, this principle is related to data, software, and hardware being available to authorized users anytime these resources are demanded, even if a security breach is identified or an authority misbehaves. Once the user or service is acknowledged and authorized by the system, they should be able to access the resources they demanded, or else the availability principle would fail.

These two principles are complemented by the confidentiality principle, which gained particular attention due to both its support of hiding sensitive information and the popularity of its solutions, which include the cryptography algorithms [Sta01, RSA78]. This principle aims to guarantee data access and disclosure under an authorized and restricted mode. An unauthorized information disclosure may lead to a loss of confidentiality [Sta11].

The confidentiality principle is also part of the privacy concept. Data privacy should be present during sensitive data manipulation, including its creation, transformation, storage, and deletion. The loss of confidentiality allows unauthorized people to possibly read or modify private information. This principle has also gained attention due to the development of modern mechanisms, which include biometric instruments, n-factor passwords, authentication tokens, digital signatures, and many others. Concomitantly to integrity and availability, this principle also supports security adoption in a computational system, since there is no significant success of confidentiality adoption if data are not available or integral for authorized users, who should decipher them using proper keys and algorithms.

Computer security is also based on a combination of the presented principles, according to the environment and requirements. In the cloud computing stack, these principles are also present, and they support the Privacy Level Agreement (PLA) demands of users and companies of any size [Clo]. Such agreement documents, as described by the Cloud Security Alliance, intend to support cloud providers and companies in defining both the baseline of mandatory data protection achievements and the structure of data protection levels. Besides the application of those three security principles, PLAs also define the Transparency aspect, which describes how cloud providers can review security structure. The mechanisms adopted for supporting the security principles may vary, but in some situations, their effectiveness is questioned, especially in public cloud environments. The Section 2.1.2 will explore such security details for the clouds.

2.1.2 Cloud Computing Security

Cloud Computing

Cloud computing has been a business model for many companies, from startups to global corporations. The root technologies involved in the cloud computing instantiation are mainly based on hardware virtualization, distributed systems (Cluster and Grid Computing), web-based technologies, and system management [Raj11]. The instances of a cloud can be exposed as public clouds, private clouds, community clouds, or hybrid clouds.

A public cloud consists of instances in which the computational infrastructure is provisioned by a public cloud provider [MG11]. The main advantage is their lower cost since the hardware can be shared by different users. Public cloud providers play an important role in business, information systems, social communication, scientific research, and many others fields. Some players such as *Amazon AWS* [Amaa], *RackSpace* [Rac], and *Google Cloud Platform* [Goob] follow common public cloud patterns to offer Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS) for customers, all of them using virtualization concepts [BDF+03]. This technology supports hardware consolidation to reduce power consumption and also provides network optimization, including low impact management and high availability [BBE+13]. However, one of its disadvantages is related to the infrastructure sharing that could raise some security concerns in organizations. This will be discussed later in this chapter.

In contrast, a private cloud environment consists of an instance in which the cloud infrastructure is provisioned for an exclusive use of a single organization [MG11]. It is a controlled environment under companies' security rules using internal networks and well-known features such as in standard data centers. Cloud concepts are here related to the way resources are managed. As in a public cloud scenario, hardware and software resources are available as a service model. In most cases, the company's private cloud is offered as an IaaS. It is possible to identify the advantages of applying cloud technologies (such as virtualization) even into small environments [MW11]. However, the private cloud have a disadvantage, which is related to instantiation costs, since the company has an exclusive cloud.

Since both cloud instances types bring some advantages and disadvantages, a company could have a solution that would use a hybrid cloud instance, which would integrate public and private clouds in a single point of view in order to get the best from each of them. For instance, in a hybrid scenario, the company can take advantage of processing its sensitive data in a private cloud and use the public cloud scalability and cost-effectiveness to process other data according to its demand. Figure 2.1 presents an overview of a hybrid cloud instantiation. According to the NIST definition [MG11], a hy-

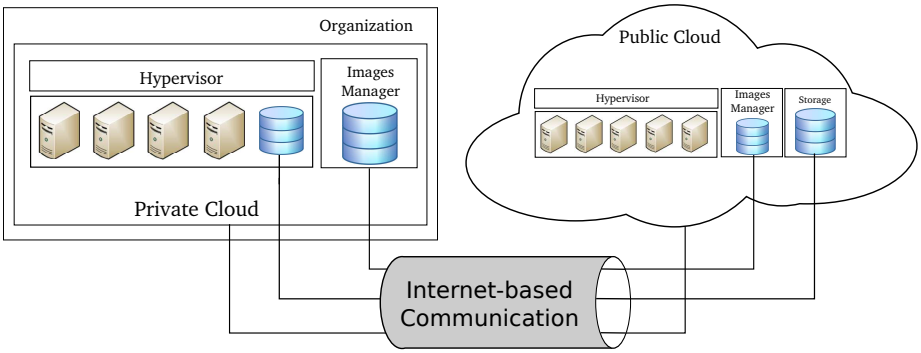

Figure 2.1 – A Hybrid Cloud instance is a combination of a Private Cloud, where computational resources are under companies' rules, and a Public Cloud through the Internet.

brid cloud is a composition of two or more instance types (Public, Community[2], Private) that remain unique entities but are bound together by technology that enables data and application portability.

In most hybrid cloud instances, external communication is commonly made through the Internet, since it is the cheaper platform for long-distance communication. Nowadays, fewer alternatives for communication solutions, such as direct links, are economically feasible. Although some companies have invested in undersea Internet cables [Bur], it is not a standard solution for most companies or business models. Commonly, the cloud communication follows the Internet-standard model.All features of a cloud instance are named "as-a-Service" and the access to each service, including their creation and management, is fully through communication over the Internet.

On one hand, the connection among cloud types is commonly offered based on either a Virtual Private Network (VPN) through a gateway [JZ11] or IPSec-based communication [KA98]. These features are used to build an interconnection in the IaaS design, running all connections of an application in the same manner as in a LAN network, which is supported by the tunnel-mode of VPNs. This model easily adheres to software deployment for its transparency, avoiding software modification. In the other hand, the interconnection between private and public scenarios is made through direct application communication or over WebServices [ACKM04]. This model does also support security through HTTPS implementations [FR14, NFL+14]. In this scenario, private instances of an application know the URL of the public ones, and the communication is made in the SaaS design. Although this avoids a low-level network instantiation, since no extra services are needed for the interconnection, developers need to open applications' ports for the public Internet, which could turn the application vulnerable and require application-level security implementations.

Once the application is deployed in the public cloud, files, data, and other assets should be persisted in its storage system. The storage system used by cloud computing

[2]The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns.

is often based on virtualization storing technologies [Cla05]. Although there is overhead added by this kind of storage system in cloud environments, currently many studies have been made to improve data access performance [FK09, GFC14]. Following the sharing ideas of cloud computing, virtualized storage systems are also deployed in shared central units with a management abstraction layer, such as Logical Volume Management (LVM). In order to keep the persisted data secret, some cryptography techniques are also applied to the data, using mechanisms such as the Cryptography File Systems [WDZ03] or a cloud storage tool that supports confidentiality [KL10, Amab, Gooa].

In addition to communication and storage systems, the processing axis gained attention from cloud users due to its high fragmentation capacity, which also draws a rupture in cloud pricing. The pay-as-you-go fashion, often adopted by public cloud providers, sells CPU time according to its model (i.e., considering chip frequency or graphic instructions in GPUs) and allocation, often based on time periods such as per hour. The term “sharing”, in this context, has a deeper abstraction compared with traditional Operating Systems. In this new layer, supported by hardware virtualization, the CPU is shared with multiple virtual machines at the same time, following rules of the virtualization scheduler [BDF+03]. Although most virtualization systems offer isolation of virtual machines [WZL15, GA03], some security issues have already been explored by attackers. This topic drives our research in the next section.

Security Taxonomy for Clouds

Once information is digital, some security mechanisms are needed to keep it safe. For offline systems, such as in private clouds, a few techniques, such as keys and passwords, could make them secure. For online systems, as in public clouds, it is reasonable to adopt cryptography, digital certificates, or even complex authentication mechanisms. Considering software deployment on cloud environments, security techniques have been revisited in the past few years to support companies’ requirements.

The principles of cloud computing move security studies to consider multi-tenancy scenarios, where different users could share the same virtualization stack to run their VMs. This scenario could enable inter-VM attacks or even information leakage [IHG10]. In addition to this, many other threats make managers consider cloud environments an unsafe solution, demanding the adoption of extra components.

Gonzales *et.al.* [GMR+12] present a taxonomy where security is the key and should be implemented for a safer environment. The Network Security category considers that all security principles applied into a company’s network should be extended to the cloud and be constantly synchronized to cover all local or remote resources or processes. The Data Security category considers the protection of data in terms of confidentiality, availability, and integrity, using cryptography and redundancy techniques during

the data's entire life cycle. Subsequently, the Virtualization category explores the vulnerabilities of virtual machines, where companies' software processes are run. The main threats to be considered are related to isolation of virtual machines, in which malicious entities could exploit data leakage and cross-VM attacks. Although those threats could be avoided by security updates of the virtualization platform, issues related to VM identification or VM access could also open backdoors, if they were not properly configured.

Regarding network communication, most threats have already been covered, by both the solutions of a non-virtualized environment (i.e. network firewall and file system encryption) and the specialized security solutions for environments based on virtualization technology. In [RCM09], the authors present the requirements to adopt Intrusion Detection Systems (IDS) in a cloud environment. IDSs normally use common network filters, such as firewall and access control systems applied in standard data centers, to detect malicious behavior; these filters also have been tested and validated in virtualization-based scenarios. In this case, no extra features are needed for a cloud environment. Regarding the adoption of confidentiality, IPSec and other VPN implementations (such as OpenVPN [Yon]) apply cryptography algorithms for ciphering the payload of the protocol stack [HKH+10, LS11]. The cryptography used for this process is based on symmetric algorithms, which use a single key for encrypting and decrypting data when in transit.

Similarly, the security solution for data storage in cloud environments could also follow standard patterns, such as cryptography. The solution should provide confidentiality (where the provider does not learn any information about users' data), integrity (any unauthorized modification of stored data should be detected by the customer), availability (customers' data should be available from anywhere at any time), reliability (customers' data should be backed up), efficient retrieval (data retrieval should be comparable to that of a public cloud), and data sharing (customers could share their data with trusted parties) [KL10]. Some cryptography techniques also provide auditing support for a cloud environment using a public key design for third-party auditing with no local data damage and adding minimal overhead [WWRL10].

These privacy-preserving techniques may be applied in different layers of the storage stack, and each of them has their own specific pros and cons [DW10]. In the Application layer, although it could be easy to deploy and adopt several kinds of encryption algorithms, performance is clearly affected compared to lower layers (which is critical in storage systems). Meanwhile, if privacy is applied in the File System layer, there are still several algorithms, and it is possible to offer privacy to the upper layer in a transparent manner; however, once the file structure is created, it becomes difficult to modify or revoke keys since it would be necessary to modify the entire list of files and directories. One layer bellow, Block structure is responsible for storing raw data, and it is possible to apply several different algorithms with higher performance compared to upper layers; however, nothing, not even the metadata, is kept in plain text in the File System, which

requires longer operations for key revocation or modification. The lower layer in this stack is the hardware-embedded cryptography. Although this layer offers higher performance than above layers, the cryptography characteristics, such as keys and algorithms, are predefined, reducing the security and privacy aspects. As a main conclusion, presented by the survey of Diesburg [DW10], the File System layer offers transparency and flexibility for the application layer; however, performance should be carefully analyzed in order to reduce overall impacts in the user's software execution [MJ15, RK14].

The most distinct aspect to be considered in a cloud concerns the execution of sensitive processes in a shared environment. For threats related to this execution, a specialized solution should fundamentally consider the virtualization layers. The Virtual Machine Introspection (VMI) technique considers monitoring a VM's memory and storage without intrusively injecting agents from outside [WTM+14]. With the monitor's information, it is possible to take some actions to protect the VM by migrating, pausing, or even shutting it down, mitigating a possible attack. A virtualization-aware solution should not consider only memory and storage but also the CPU allocation. Some vulnerabilities, such as Kernel rootkits are transparent in computational systems since they run in a privileged mode – the kernel mode. However, in a VMM-based environment, it is possible to intercept noticed rootkits, due to the running of virtualization in a lower level, and offer security in a transparent manner [RJX08]. Regarding isolation security, researchers tried to solve cross-VM attacks that exploit side-channel attacks with novel cache architectures, allowing per-VM cache allocation [WL08]. From a different perspective, the possibility of processing encrypted data, such as searching over encrypted data, comes as a solution discarding any other security mechanism. This is possible through systems such as the Fully Homomorphic Encryption [Gen09] and Querying over Encrypted Databases [KKTM11]. Such implementations are explained in detail in Chapter 6.

Based on the Security Principles, Cloud Computing definitions, taxonomies, techniques, and implementations presented in this chapter, Figure 2.2 shows a summary of the security concepts and their implementations, organized as a dependence from Cloud Security and classified according to the three computational system axes. It is possible to observe the presence of the confidentiality principle in all axes, each one with its own implementations. From this analysis, one can consider the creation of a cloud computing environment with confidentiality in those three axes by choosing the implementations properly.

Although functional, the solutions to build a secure cloud environment do add some overhead in computational systems or even extra costs on the public cloud, such as renting VPN gateways.

In order to support the execution of applications in a fully secure manner, either the rented public environment should be dedicated to companies' processes (adding extra costs), or it is necessary to add encrypted processing techniques in accomplishment with

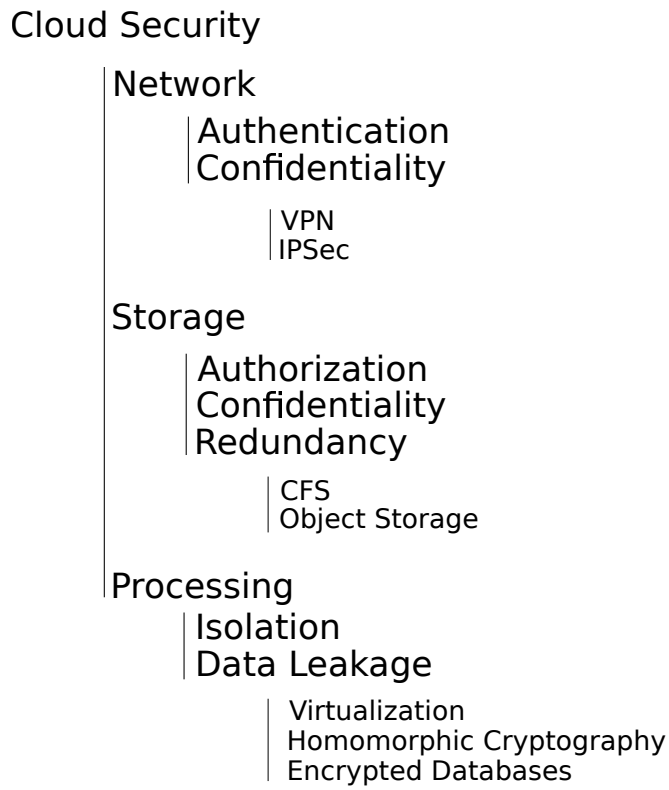

Figure 2.2 – Reviewed Taxonomy of Security applied to Cloud Computing, based on [HA12].

company's PLA. In addition to that, the data stored or transferred through the public cloud should be encrypted, giving support to privacy principles. From these requirements, one can consider building a cloud instance and applying the tools and techniques to support higher security levels. However, the impact of those privacy principles should be mapped in order to achieve a better estimation of resources allocation in the public cloud instance, from performance and services' response time to costs. These aspects drive the main line of this research, and the state of the art for both cloud security architectures and cloud security cost models are presented in next section.

2.2 State of the art

2.2.1 Security Cloud Architectures and Designs

The nature of the cloud computing environment incurs several benefits, from the infinite scalability of resources, to the efficient business model, through the pay-as-you-go fashion. These features are mainly supported by the resource-sharing approach that allows cloud providers to cut dedicated expenditures, improving the cloud feasibility for both users and providers [ZCB10]. Nevertheless, this computational model opens issues covered by researchers in several areas, including information security. By sharing

physical resources among different users, it is possible to explore confidential information leakage in several ways [RC11], as discussed in Section 2.1.2. The issues and challenges of cloud computing security [SC17, NSMZ16] have been mapped and constantly updated by researchers, considering security for communication, storage, and processing of users' data.

Regarding data confidentiality, solutions for achieving users' privacy-level agreements have been developed for different scenarios, considering mobile applications, in-hardware security, and authorization-based frameworks. Their deployment in the cloud environment has produced different architectures according to each software requirement.

Cloud mobile frameworks are solutions deployed in the cloud computing environment, used as processing and storage resources for mobile application. For instance, a mobile app used as an interface of an email service delivers the search operations to a cloud service that processes it over a mail account. The result is then returned to the mobile app, reducing the CPU usage in the mobile phone and, consequently, cutting battery consumption. Although efficient, most of the Mobile Cloud Computing frameworks (MCC) have security issues. This kind of framework has faced some barriers to its adoption due to such problems, and some surveys in the industry show that 74% of IT managers would avoid cloud environments to support their mobile solutions [KKKM13]. Following those demands, researchers have produced solutions for solving security and privacy issues in cloud environments.

The work produced by Huang D. *et.al.* [HZKL10] presents a solution where a mobile application can relay some processing phases to a cloud environment. The proposed framework covers the privacy aspects related to identity management, data access control, and risk management. Similarly, the work presented by Chadwick *et.al.* [CF12] aims to support the privacy principles by protecting users' information through an authorization mechanism in IaaS models, where every access would be controlled by polices defined by the user. Both solutions do not adopt any cryptography to protect data or processing handled in the cloud environment , which is a demand from the companies behind these applications.

To increase the privacy in a computational environment, including clouds, some solutions are considering hardware-based security. The authors Itani W. *et.al.* [IKC09] present a solution called *Privacy as a Service*, and they define it as a set of security protocols for ensuring the privacy and legal compliance of customer data in cloud computing architectures. The solution adopts cryptographic co-processors for ensuring tamper-proof capabilities where undefined accesses to sensitive data are alerted to users. By also modifying the hardware structure, the authors Seol J. *et.al.* [SJL+16] present a cloud architecture with a hardware security module that supports a Virtual Machine Monitor to verify the authenticity of sensitive data access only by trusted Virtual Machines. The solution considers

adding a trust platform module, which is a commercially available hardware-based root of trust that cannot be modified by software layers [Tru]. This hardware-based component is also presented by Santos N. *et.al* [SGR09], where a cloud environment also has a *trust coordinator*, which is responsible for managing the allocation of sensitive user demands. Although hardware-based solutions are stronger against software attacks and risks, it is not a standard solution offered by cloud providers, and it would increase the total cost of services.

To create solutions for clouds using only standard hardware (without any additional processors), some works consider applying security algorithms and protocols to avoid information leakage. The authors Waqar A. *et.al.* [WRAK13] present a framework to increase users' privacy by protecting the database structure using cryptography and privacy preservation operations. The framework modifies the database metadata before storing it in the cloud environment, and the reconstruction is made dynamically, only when data access is requested. Such implementation is deployed as a PaaS model, where the developer replaces regular databases with a solution with higher privacy levels. Similar to end users in the SaaS model, the authors Fahl S. *et.al.* [FHMS12] present a solution where a third party component is responsible for ensuring privacy mechanisms for cloud applications such as Facebook, Dropbox, and Googlemail. The authors named the solution *Confidentiality as a Service*, and its main focus is allowing final users to use cryptography in a usable fashion, integrating security in a familiar Internet context. The addition of a security mechanism always increases resource usage in the cloud, especially when data is modified, such as in the ciphering process. To reduce the overall impact of security, the authors Chuang I. *et.al.* [CLHK11] present the *Effective Privacy Protection Scheme*, where the security requirements are mapped, and a security solution is deployed to fit such requirements, trying to reduce the CPU overhead added by cryptography algorithms.

The solutions applied in an upper layer in the cloud software stack are often developed to cover some specific demands. Concerning the cloud deployment model IaaS, the solutions for security could cover a wide set of services, since they are applied to lower components such communication and storage.

The authors Puttaswamy K. *et.al.* [PKZ11] discuss the adoption of confidentiality for data stored in public cloud environments. The main features covered by the solution, named *Silverline*, consist of identifying the sensitive application's data that can be encrypted, managing the key distribution associating them to subsets of data, and allowing a transparent access to encrypted data but preventing inappropriate access by malicious parties. Although protecting the stored data with low overhead, this solution does not hide the database structure, and some attacks could exploit the data relationship learning where sensitive data are stored, allowing information leakage in some manner.

Seeking a solution to cover more features in terms of confidentiality, some works consider not only to protect the stored data but also protect the processing phase on them.

The authors Paladi N. *et.al.* [PGM17] present a solution where data are stored in a trusted manner, encrypting before persistence, and processing on virtual machines created with a *trusted launch* technique. This technique consists of applying a digital signature in the VM's images that guarantees their content is not modified, and the software will handle data appropriately and with confidence. This technique covers one of the attacks in the cloud environment that injects malicious software in VM's images, trying to leak data externally.

Besides the efforts in adding a security mechanism to support privacy principles in cloud environments, there are a few works presenting solutions supporting privacy in a fully trusted fashion, especially in a lower and wider level such as in IaaS cloud deployment. The authors Wei L. *et.al.* [WZC+14] present a solution for guaranteeing data privacy during the storing and processing phases. This solution presents an auditing protocol and a cheating-discouragement strategy where data access and processing are monitored, and operations performed in such an environment are verified. Still, data confidentiality could not be guaranteed if an attacker exploits the virtualization breaches, since in-memory data is in plain text during their processing. Recently, Haimbala J. [Hai16] published a thesis where data is stored and processed using a cryptographic mechanism for ensuring data confidentiality during the entire data's life cycle. Besides the adoption of symmetric cryptography algorithms in the storing module, the author uses Searchable Symmetric Encryption Schemes (SSE) for processing encrypted data without disclosing it. Moving toward the creation of a confidential cloud environment, the proposed approach follows the main concepts we expose in this thesis' motivation and objectives; however, by using and validating only SSE the author does not consider the possibility of using other secure processing mechanisms, such as the Homomorphic Encryption and solutions based on Intel's SGX [Int14] instructions like Haven [BPH14] and VC3 [SCF+15]. This mechanism is then applied by Makkaoui E. *et.al* [EMEBHM16, MEH15] in a similar perspective, where a layered confidentiality design is presented, considering hardware-based cryptography for encrypting data during persistence, and digital signature verification. However, the authors do not explore the alternatives for the processing and communication layers, and do not even consider the cost evaluation of the cryptography mechanisms.

2.2.2 Security Cost Modeling

The challenges to building a secure public cloud environment have demanded researchers' attention over the years. However, the impact of supporting companies' security requirements is rarely mentioned or even computed in providers' costs.

Earlier in the data center era, a security risk estimation was given by calculating the frequency of attacks versus the possible loss amount for each data file. Another model,

which follows the one previously mentioned, was called Cost-Benefit Analysis (CBA), which could be applied to any application, was based on identification and measurement of all related costs and benefits. It includes ”lost opportunity” costs, needed to account for shared costs and uses, and must consider and address risks, qualitative factors, and assumptions. Those variables, if not carefully taken into account, may produce grossly over- or underestimations. However, the result is very important since most managers and directors know little about computers and computer security, but they do understand risks and cost-benefit analysis [Mer03].

During the emerging of cloud computing some years ago, Grid Computing technology was mature and in use for some companies. However, in that time, companies hosted their own infrastructure, and the security of the system was supported only by firewalls since computers were not shared with third parties. In addition to this, costs were not measured in terms of CPU time, storage space, or even financial values. In *What Does Grid Computing Cost?* [OKS08], the authors make an evaluation of a real Grid instance and measure the financial costs of the environment. However, the study does not consider the impact of security once the infrastructure is under a company’s control.

Later, during the dissemination of SOA (Service Oriented Architecture), researchers and developers started to adopt security mechanisms to build SOA-based software. Since the cloud resources were still high priced at that time, it was important to measure the impact of adding those mechanisms. In [YYA09], the authors present a trade-off model which considers balancing the performance axis and the security axis. The performance metric was a function of traffic amount and communication delay, considering the security of a system (which is composed by the security functionality, the cryptography algorithm, the key length, and the protection percentage). The security metrics, although hard to be accurate, are defined as the product of the cryptography key’s size, the attacker’s capability of breaking the cipher, and the percentage of protection (i.e. the number of packets encrypted during the communication). Both the performance and security metrics were used to build a trade-off objective function on which it was possible to define weight factors to balance the preferences of the security system. Although the model could estimate the impact when applying security requirements, the work only considers the communication mechanism. Likewise, when applying the model in a cloud environment, the performance measurement should consider new variables, such as the virtualization overhead as well as a higher protection percentage.

On the other hand, from cloud providers’ side, it is important to manage the variability of resources usage over time to achieve better profit results. In [GGT12], the authors present a cost estimation model for federated clouds. The model, on one hand, considers allocating resources in external cloud instances, when local resources are fully loaded. On the other hand, when local resources are empty (or partially empty), the model

estimates the costs of keeping the infrastructure online, as well as the revenue of renting it to a third party. Both scenarios are modeled as a function of variables such as:

- *Price_VM_Hour*: price per hour of renting a VM;
- *VM_Node*: the number of VMs hosted in a single bare metal machine;
- *Nodes*: the amount of bare metal machines in the data center;
- *Cost_Node_Hour*: maintenance costs of a single bare metal machine;
- U_p: the utilization of resources in a provider p.

to achieve the results:

- $Costs_{\{p,o\}}$: costs of the resources of a provider p and its outsourced o resources;
- $Revenue_{\{p,o,i\}}$: revenue of the provider considering its own resources, the out-sourced, and the in-sourced;
- $Profit_p$: the provider's profit, given as the result of allocating resources locally, outsourcing part of the demand, and renting some extra sub-utilized resources.

The model was evaluated considering a real scenario of a federated cloud environment. The environment was managed by a resource scheduler which considers balancing the virtual machine allocation over the public cloud instances and the local-virtualized hosts. However, the model was applied only for CPU utilization, and neither the communication nor storage was considered. Security factors were not considered too.

Similarly, in [STT+12] the authors present a pricing model based on the BSM (Black Scholes model, used for option pricing in financial market) and Moore's Law (which considers the computational power each 18 months to be doubled). When these two formulas are combined, it is possible to calculate the depreciation of a computational resource as:

$$ResourceVal_{t=T} = ResourceVal_{t=0} * (1 + r)^{T/2} \quad (2.1)$$

where r is the rate of interest and T is the resource's life time. The result, in its turn, will be used as one of the five variables considered in the BSM model, which considers

- *Initial_Investment*: the amount of money a cloud provider will spend per year;
- *Contract_Time*: the time for which a user will lease the resources;
- *Rate_of_Depreciation*: the rate at which the hardware is expected to lose its financial value;
- *Quality_of_Service*: the quality assurance to the client;

- *Age_of_resources*: the age of a resource leased to the client.

The application of the BSM model has produced as results the effect of the following axis on the resource price: initial investment, contract period, resources' depreciation, and quality of service. Although the application of BSM in this research is related to the infrastructure of a cloud, the results have shown that combining financial models and computing theories promotes the creation of new formulas for cost models. Some recent papers [STTB15, TVRB15] improved the cloud financial value model and proposed tools to simulate the costs in a cloud environment [ABF+16].

Some feasibility models have also been developed to support cloud providers' decisions through scheduling of Virtual Machines according to economic features [KLM16, LLLZ16, MNGV16]. These models produce resource allocation answers for cloud managers with the aim of including more variables, such as pricing market and commodities allocation (energy, network). Recently, Jouini *et al.* [JR16] present a cost modeling driven by a multiple analysis of security breaches in cloud computing environments and using a quantitative approach to create an assessment model for security risks.

Regarding security, another approach that should be considered is identifying the sensitive part of applications to deploy the security mechanism properly. Watson P. *et.al.* [Wat12] present a multi-level security strategy, selecting sensitive applications' workflows and running them in a safe location. This safe location can be related to either a security-aware cloud environment, which would be penalized by cryptography processing, or a private cloud instance managed by a company's security rules, avoiding sharing of computational resources. In such scenarios, it is possible to statically invest in the security of some specific resources, since an IT manager knows the sensitive applications' placement [FW12]. Although a static placement of security mechanisms could be easily guessed, one may consider using a model to estimate the impact of these mechanisms for calculating the monthly, even static, investment on the security cloud resources.

2.3 Summary

This chapter presented both the background and the state of the art used in the discussion of the topics in this research. The research mainly includes the themes of cloud computing and security, especially related to confidentiality.

Initially, the chapter presents the principles of the computer security research field. We pointed out the confidentiality principle and its implementations to be evaluated in a cloud computing environment from the performance-impact perspective. The confidentiality risks, threats, and solutions for cloud environment were discussed in this chapter, where technical aspects and implementations were identified and classified in a

taxonomy considering the three axes present in a computational system: communication, storage, and processing.

The chapter also presented works in the state of the art, considering the application of mechanisms supporting a confidential cloud environment, with guarantees of confidentiality in all data's life-cycle. Although most of the works are partial solutions (most do not cover confidentiality in the processing phase), they use standard security components for supporting privacy principles. The components include cryptography algorithms to protect data and digital signatures for authentication and authorization processes. Some solutions also consider installing physical components (also called cryptographic coprocessors) into the servers to securely process data. In addition, there are some works that consider the creation of a fully secure cloud environment based on software-based solutions and commodity hardware. Such approach allows cloud user to choose regular IaaS cloud providers to deploy their sensitive components with confidentiality guarantees. Similarly, cloud provider can start to offer PaaS and SaaS with confidentiality support.

Regarding the security costs, there are several works in which performance and security are modeled in scenarios with and without cloud principles. However, to the best of our knowledge, those models do not put together security principles and the concepts of the cloud's components, such as the shared tenancies in the virtualization layer; neither evaluate the security principles individually, which is a contribution of this work. Although estimating the usage of resources on a cloud environment is complex due to its high abstraction of the physical layer, the state of the art in modeling computational systems supports this research in developing a cost modeling for the confidentiality principle and its implementation for cloud environments.

This chapter presented elements to answer the first part of the research question *1 - What are the confidentiality risks and solutions in cloud computing environments, and how is a confidential cloud created?*

The research on security risks and threats produced a taxonomy that points out the confidentiality principle as an important player in coping with the challenges of cloud computing adoption for users with strict privacy levels. By adding solutions such as cryptography in the cloud software stack, users can experience the benefits of cloud computing with higher guarantees of avoiding information leakage during communication, storage, and even during the processing phase.

3. FULL-STACK CONFIDENTIALITY FOR CLOUD COMPUTING

The creation of a cloud environment with support of the confidentiality principle should avoid information leakage, especially when hosting sensitive applications in public cloud environments. This principle is broken when an unauthorized party has access to an inappropriate information disclosure. In the last chapter, we identified solutions to create cloud environments with the support of security in communication, storage, and also during sensitive data processing. However, those solutions do not achieve confidentiality guarantees when data are *in-transit*, *at-rest*, and *on-processing*. The Full-Stack Confidentiality Architecture presented in this chapter aims to identify the placement of confidentiality components in the cloud computing software stack. This placement should consider the nature of cloud computing environments that often share their resources among different users, from market competitors to malicious entities. The components include the solutions for transferring, storing, and processing data with privacy, using techniques such as cryptography and homomorphic encryption. This architecture design is a proposition based in both the architectures presented in Section 2.2.1 and their components for providing data confidentiality in a cloud environment. It is the starting point of this research, which aims to demonstrate the impact of the confidentiality principle when hosting users' sensitive data.

3.1 Confidentiality and Cloud Computing

The Security of a Cloud environment is managed in different perspectives according to the chosen deployment model: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), or Software-as-a-Service (SaaS). For the IaaS model, the security should be managed and deployed by the cloud users. They are responsible for cryptography and authentication mechanisms consuming resources, *i.e.* CPU cycles, in their virtual machines. For PaaS and SaaS, the cloud provider will deliver security as part of services such as databases, e-mail services, and e-commerce systems. In both cases, measuring the confidentiality overhead impact can help both cloud users and cloud providers to estimate resource allocation, availability, and costs. However, it is necessary to understand the placement of the security mechanism in the software stack.

On one hand, cloud providers currently offer security mechanisms like file encryption and private networks. Those services are also known as Security-as-a-Service (SECaaS) [VT14, FGT14], where security is relayed to the cloud provider. However, the cloud user has a lower control of the chosen algorithms and the key management. On the other hand, in the IaaS model, the cloud user is responsible for deploying the security mechanisms for communicating, storing, and also processing data. In such scenario,

security levels could be established following the company's security requirements. Both scenarios apply similar techniques for protecting data, especially when supporting confidentiality using cryptography algorithms.

In order to provide confidentiality in the cloud, it is necessary to see it as a computational system that is composed of data processing, storage, and communication axes. Each of these axes have their own confidentiality solutions, and they can be combined to cover the privacy level demanded by a given software or application.

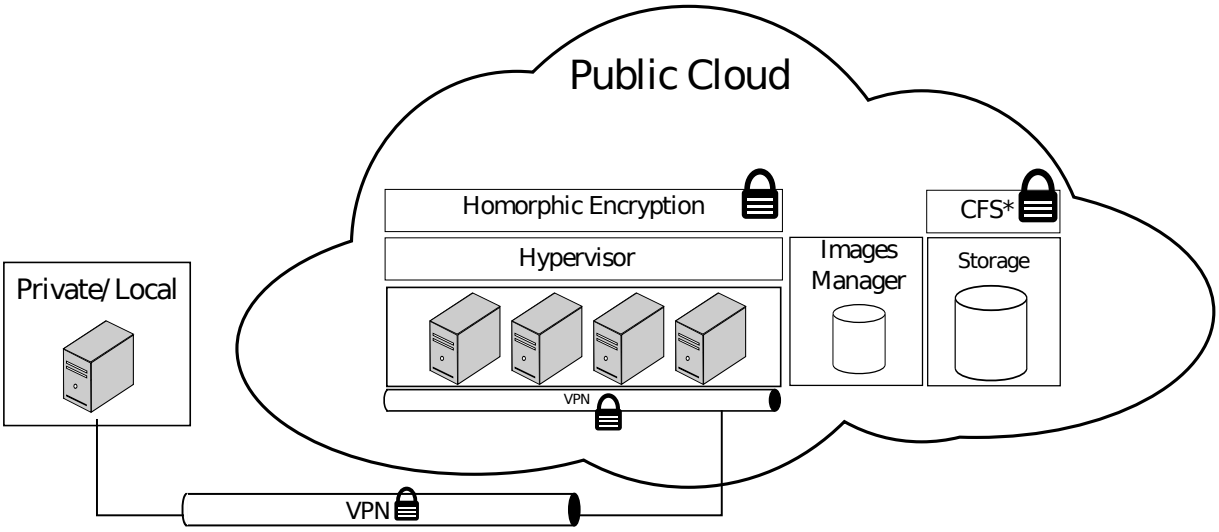

Figure 3.1 – Confidentiality for Clouds is often applied to three axes of the computational architecture: Communication (i.e. using VPNs), Storage (i.e. using Cryptographic File Systems), and Processing (i.e. using Homomorphic Encryption). Some of the mechanisms override each other in terms of data confidentiality, but can coexist depending on user's demands.

Figure 3.1 shows a cloud composed of services from a provider, which are accessed by a remote client under a private and controlled environment. The padlocks in the figure represent the adoption of confidentiality in different parts of the architecture. The instance identified as private is a controlled environment under the company's security rules, and no extra security mechanisms are applied for privacy guarantees. This client is connected through the Internet to a public cloud, where the padlocks represent where security is applied. The key management should be always handled by a trusted entity, which could be a third party or the company itself. The coexistence of confidentiality components can happen due to users' demands in guaranteeing data are never disclosed, neither during communication nor persistence nor processing.

Confidentiality could be extended to a hybrid cloud environment, as depicted in Figure 3.2. In such scenario, each cloud player should support the confidentiality requirements, and its management should be also handled by a trusted entity. The cloud services should adopt a security component in this figure represented by the padlocks. The hybrid cloud confidentiality interconnection and management is a further topic not detailed in this work.

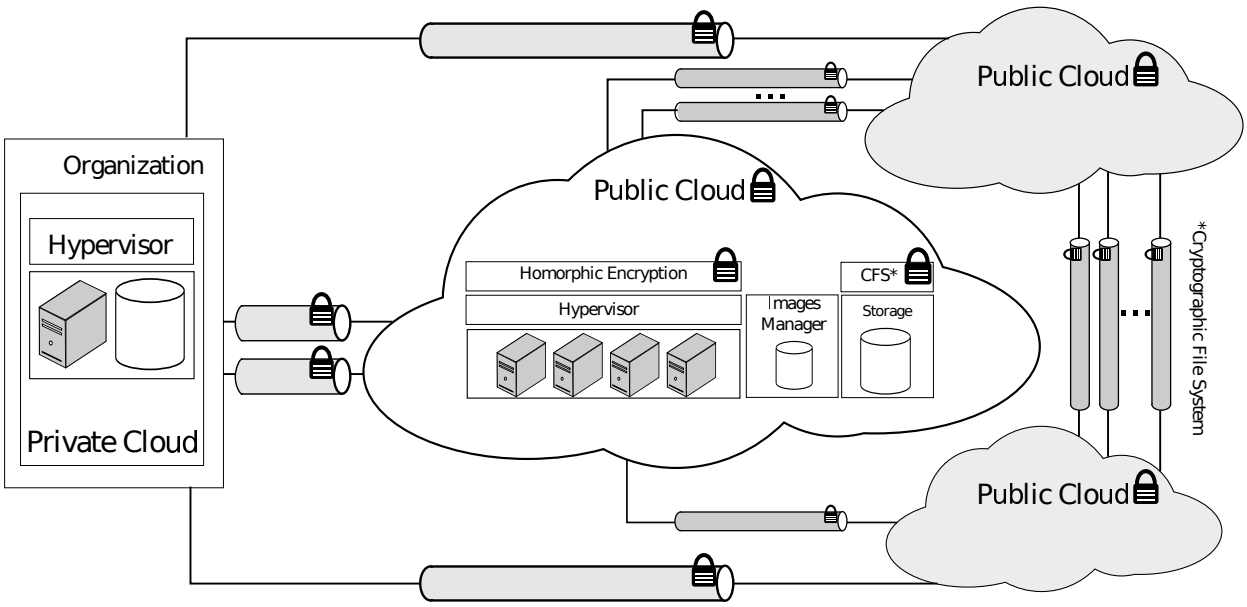

Figure 3.2 – Confidentiality applied for hybrid cloud context composed by multiples cloud instances.

3.2 Full-Stack Confidentiality Architecture

The composition of the Full-Stack confidential cloud instance considers the solution for the axes independently. Although some of the confidentiality solutions are based on similar technologies, the implementation of each component has its own characteristics. Figure 3.3 describes the cloud architecture, where the *Confidentiality* box, placed on top of the Infrastructure-as-a-Service layer, holds the confidentiality solutions that will be offered to upper layers, *i.e* for users' application and services. Each axis in the Confidentiality box can be applied independently, but can also co-exist to build an *On-Transit/At-Rest Confidentiality*, combining communication and storage axes, or a *Full-Stack Confidentiality*, combining security solutions of all three axes.

3.2.1 Network Confidentiality

The first axis to be considered is the communication among cloud nodes, internally and externally. Confidentiality is necessary here since communication on the Internet is not safe. Moreover, attackers' hosts could be placed inside the cloud, sharing the local network or even the same physical network interface, allowing internal attacks.

The connections will not only carry sensitive data. There are several different types of communication belonging to a variety of services in each instance. Those communications are related to real-time applications' synchronization, storage replication, management, monitoring, billing, etc.

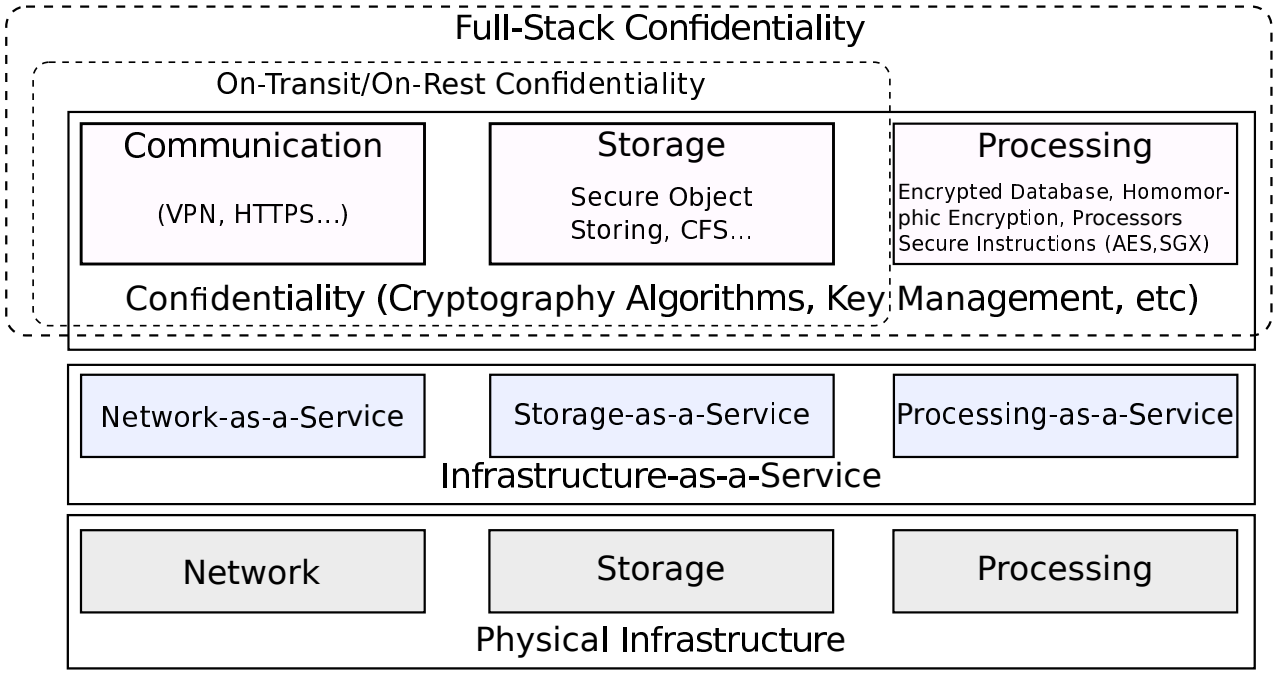

Figure 3.3 – Full-stack security architecture for cloud computing environments.

To build a confidential communication in the clouds, managers commonly create channels based on Virtual Private Networks [HKH+10, LS11]. The channels apply authentication protocols (using password or digital certificates) and cryptographic algorithms. Alternatively, managers can create several channels according to each user's security requirement. A multi-channel design, presented by Storch *et.al.* [SDRZM15], presents the results in terms of network bandwidth compared to a single channel. The bandwidth increases up to 5 times; however, the CPU time used for data communication does not exceed 5% using a single processor. Although the solution is feasible, it is not common to split the communication, since it demands extra management and software modification. Besides, IT managers often penalize security to acquire better performance during communication. This trade-off relates to the fact that some cryptography algorithms both increase the data size and take some extra time in the processing phase (queuing applications' communication).

As a standard communication channel in the cloud, webservices with HTTP SSL-/TLS support handle problems such as certificate authentication and network firewall bypass, due to its large adoption. In terms of performance, the authentication time is spread across the total communication time since it occurs only once at the beginning. Otherwise, secure communication follows the standard process, using cryptography as in VPNs and producing a similar overhead.

3.2.2 Storage Confidentiality

The second axis considered in a cloud environment is the persistence layer used by data and software storage. This mechanism is commonly managed by the virtualization technology that hosts the user's virtual machines.

To allow data confidentiality at-rest, the persistence layer should also support this security principle. Recently, public cloud providers have started to offer encrypted discs for confidentiality support for users' data [Gooa, Amab]. On one hand, from providers' perspectives, such functionality demands computational resources and may impact feasibility of cloud services. On the other hand, users would not be able to deploy their own security rules and mechanisms, relaying storage security responsibility to the cloud provider.

In order to support confidentiality in a public cloud environment, some works [KL10, PLM+11] present solutions for safely persisting data in public cloud environments. Those solutions adopt data integrity and availability principles in terms of keeping data enciphered and non-touchable based on theoretical proofs. Besides the novel techniques proposed in the literature, the Cryptographic File Systems (CFS) can be considered for data storage in public clouds. This technique, initially proposed by Blaze [Bla93] in the 90's, consists of encrypting and decrypting data during the persistence flow that supports the confidentiality principle. CFSs use regular cryptography algorithms available in well-established operating systems' images provided by cloud providers.

The impact of adding security to the persistence layer should be considered by the user since it requires additional computational resources, and users would be billed by the cloud provider. However, leaving data in plain text, stored in a cloud environment, would increase the risk of information leakage.

3.2.3 Processing Confidentiality.

The third axis is related to tasks running in a cloud environment, specifically the run-time phase. In this phase an application consists of transferring data and instruction in/out of the processor as plain text. Even if both communication and storage mechanisms add some security, this should be considered in order to keep the data's secure during its entire life cycle in a shared and public computational environment. As mentioned in Section 2.1.2, there are some vulnerabilities in virtualized environments exploited by cross-VM attacks. It will actually not be a threat if the physical host (memory and the processors) in a cloud environment is not shared. However, this scenario would increase

the costs to deploy an application in the cloud. It is also not common in public cloud providers, due to financial concerns.

In the last few years, researchers have been investigating solutions to add confidentiality during the run-time phase. The embedded cryptography instructions of Intel processors [Int06] allow (de)ciphering data using hardware instructions. Those instructions handle only the registers' data, keeping data stored in the main memory encrypted. Although it could prevent data leakage for cross-VM attacks exploiting the shared memory, the data still be in plain text in some manner inside the chip. Recently Intel presented an extension, called Intel SGX [Int14] (*Software Guard Extension*), to promote secure software execution with trust verification. Such an approach leverages confidentiality in a cloud environment, but since the users have to change parts of their application to use it, this may also impact the performance.

In order to support confidentiality for the processing phase, it is necessary to consider algorithms to handle encrypted data, such as operations over encrypted databases and the homomorphic encryption. Such solutions ensure the computing of data without decrypting it entirely, therefore avoiding information leakage.

The encrypted database querying [HILM02, BW07, AEKR14] technique considers handling enciphered data with an interface like in Database Management Systems. The technique handles queries by comparing enciphered parameters within the enciphered data stored in the database management system. In doing so, there is no exposure of sensitive data even during the querying phase. However, having been studied in last ten years, the encrypted database querying technique still is a basic database system with poor support for rich SQL instructions, such as the store-procedures [AEKR14].

At same time, research about homomorphic encryption has attempted to develop a rich mechanism for confidential data processing. The homomorphic multiplication operation provided by the RSA cryptography [RSA78], supports multiplying two encrypted values, resulting in an enciphered value which is the product of the original two values. For example, for the values p and q, the encryption function enc, and the decryption function dec, $p * q = dec(enc(p) * enc(q))$ [RAD78]. The only way to acquire the result is by deciphering the computed value with the private key, which is a pair of the public key used over the two original values. This feature enables handling values without disclosing them during the processing phase, and it is called a homomorphic operation. The work called Full Homomorphic Encryption (FHE) [Gen09] proposes a complete model for computing encrypted values. This theory is considered a definitive solution for handling enciphered data in a strong, secure manner, providing additive and multiplicative operations [vDGHV10] that could support a rich set of functions for computational systems. However, it is still a theory, and no feasible solutions based on cryptography have achieved the desired results, even at a high overhead.

3.3 Real Scenarios Implications

The security principles applied to the entire stack of a cloud environment can support higher privacy levels in accomplishment with users' requirements. Currently, the privacy policies of public cloud services neither describes details about the back-end mechanisms (such as key management or data placement) nor offers tools for controlling confidential data's life cycle. To provide confidentiality in a full-stack manner, it is necessary adopting techniques such as the ones presented earlier for the axes: communication, storage, and processing. However, adopting security mechanisms in those three axes will change aspects of application execution, platform management, data placement, costs estimation, etc.

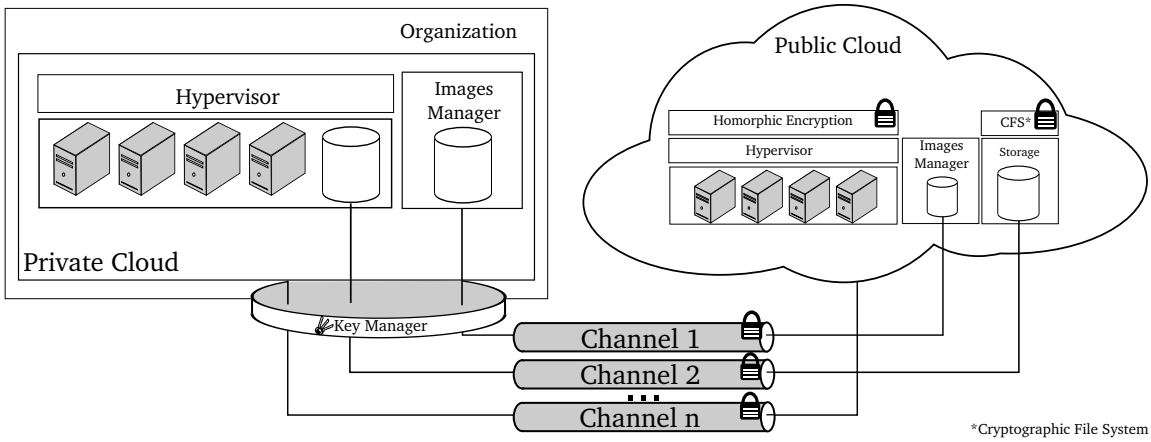

Figure 3.4 – Secure interconnection of two cloud instances in a Full-stack design.

For instance, one may consider providing a set of desired security requirements in a distributed cloud environment, composed by services of two different cloud providers, as depicted in Figure 3.4. The padlocks in the figure point out the components where confidentiality can be applied. By adopting confidentiality techniques such as cryptography algorithms, it is also necessary to add a key manager in the architecture, responsible for keeping cryptography keys safe and available for authorized users, in this case placed on the organization's side.

Both internal and external networks should provide an encrypted layer to connect cloud services. This connection could be based on VPNs (Virtual Private Network) where a new private LAN will be instantiated. From this point, the application peers need to share the network addresses of this new private network to contact each other. In such cases, the VPN channel should be previously instantiated on virtual machines, which could increase the deployment phase time. However, some benefits could be identified from multi-channel instantiation, such as load balance and bandwidth aggregation [SDRZM15].

Alternatively, applying standard communication of the SaaS (Software-as-a-Service), by using a webservice design, comes a solution to provide private connections

through HTTPS for applications in a more transparent mode and avoid the VPN management. However, the application will need either a built-in webservice or an API based on this technology.

Besides transmission setup, data should be stored in a secure manner, using the Cryptographic File Systems or an Encrypted Object Storage system. Considering data storage, symmetric cryptography is commonly adopted due to its low overhead compared to asymmetric cryptography. In such cases, there are two possibilities: sending the already encrypted data and sharing the cryptography key; or encrypting data on remote site with a remote key, adding some overhead if an encrypted communication is also adopted and creating a key management problem since the Organization would need to control the cryptography keys [FN94].

The first option requires an application's feature to be specially designed for master-key exchange since it will be necessary on the remote site for data access. However, there are risks in exposing the entire data on both sides if the master-key is stolen either during its transmission or on the remote environment. The second option could be implemented by platforms' services such as cryptographic files systems. After transferring the data through a secure channel (such as a VPN), it will be automatically encrypted and then persisted on the remote storage system. Although this scenario could be transparent for an application developer, it would add extra processing time to both sides, considering the channel must read-and-decrypt data from source storage, encrypt-and-decrypt it for transmission, and finally encrypt data for persistence on the destination side.

Even if both communication and storage layers are encrypted, to provide confidentiality in those mechanisms, the data will only be confidential if the plain text is never handled, even on memory or in registers inside the processors. If the cryptographic algorithm opens the plain text at any point, there are no security guarantees, since the environment is not entirely under users' control. For example, the cryptographic file system only ensure confidentiality for persisted data and all cached in-memory bytes are in plain text. The same happens with Virtual Private Networks that support confidentiality only for in-transit bytes.

The mechanisms mentioned in Section 3.2 can hide the plain text even during the processing phase. However, those techniques, such as homomorphic encryption, impact some aspects of the application. Initially, there are some issues to be verified in the software architecture since the application hosted in a public environment will handle only ciphered data. Every time an authorized user wants to access the plain data, it will be necessary to call the key management software (see Figure 3.4) to acquire the cryptographic key. It is important to implement this process on the application and also rewrite parts or even the entire system since it will handle sensitive users' data. The main problem is related to the key management placement security, but the literature [BKL+09, KD12, YWRL10] has already discussed solutions to enable trusted platform integration.

Nevertheless, performance becomes an issue since it is strongly affected by the cryptographic processing techniques. Although the encrypted databases have a similar performance when compared to the non-encrypted systems, the operations made over data are restricted to comparing bytes – in this case, encrypted bytes. The plain text would never be exposed, but some operations in relational databases, such as *triggers* and *procedures*, could not be implemented. To provide a full set of encrypted operations, the encrypted database needs to rewrite standard database functions making the operations confidential [FW12], bringing to the cloud only the storage role and consequently increasing the processing time.

The homomorphic encryption, which aims to provide computation over encrypted data, still has not been implemented in production environments. Although it solves most of the problems related to privacy in public cloud environments, there are some issues related to both performance and storage. The techniques of current implementation increase both the processing phase and the data size since they are commonly based on asymmetric cryptography. However, the performance issue could be solved in the near future with specialized processors, as well as processors with embedded instructions [Int06, Int14], which have been used for symmetric encryption.

3.4 Summary

There are several solutions for supporting confidentiality for users' digital assets that can be deployed in a cloud computing environment. Moreover, recently, cloud providers have been supporting security by offering services with features for encrypting data during communication and storage. In order to understand the security component placement in a cloud environment, this chapter presented an architecture design used to identify the confidentiality component placement in cloud computing environments to support data-leakage avoidance in the data's life-cycle, which consists of applying the confidentiality principle during data communication, storage, and processing.

The architecture design puts confidentially components over the Infrastructure-as-a-Service model to support application execution on top of them. The components, as mentioned, can co-exist to improve the confidentiality level, but they can also be deployed independently. The chapter ended with an overview of the implications of adding such components in a cloud environment, which include the increased deployment time due to the setup of confidentiality mechanisms and, in some cases, as when using homomorphic encryption, the modification of the users' application. Besides these implications, there is additional overhead introduced by those components, and users may need to calculate the extra costs as more confidentiality is added to the software stack. The next three chapters aim to extend these topics, focusing on the performance overhead, through creating a

prediction model. The architecture presented in this chapter is used as a reference to identify the components and their roles in the cloud software stack.

This chapter contributes to the validation of the hypothesis *(i) it is possible to support confidentiality in all data's life-cycle in cloud computing environments* by answering the research question *1 - What are the confidentiality risks and solutions in cloud computing environments, and how is a confidential cloud created?*

By properly choosing the confidentiality components matched with the security demands and the cloud computing software stack, it is possible to improve confidentiality guarantees such as information leakage prevention along with the services implemented by a cloud computing provider. These services can support the confidentiality principle through handling users' data using cryptographic tools and protocols during the three main phases of users' data: communication, persistence, and processing.

4. NETWORK CONFIDENTIALITY MODEL

The Full-Stack confidential architecture, designed in the previous chapter, presented the confidentiality components placement following the computational aspects of cloud computing. The modeling of security impact in the proposed architecture should observe mechanisms and techniques used for supporting security principles, in this case the confidentiality. Each axis in the cloud architecture has its implementation, and they can be modeled separately. In this way, this chapter presents the modeling for the network axis, which includes the interconnection among cloud endpoints considering the public, private, and hybrid cloud, internally and externally.

Based on the security taxonomy of cloud computing, presented in Chapter 2, it is possible to identify at least two main principles that guarantee privacy for networks. Firstly, an authentication mechanism is responsible for establishing the communication within authentic peers. One can identify two main events implemented by mechanisms that support this principle, one at the beginning (exchanging identities' information), and another at the ending (closing the communication). Secondly,concerning the confidentiality principle, it has to guarantee aspects for avoiding disclosure of in-transit data. This principle is present in the entire data flow from the connection's beginning to its ends. The most common method for hiding data during their transference is ciphering them using cryptography algorithms, which demands extra resources in both the sender and the receiver. This overhead is the main focus of the cloud network cost modeling.

Following in this chapter, the network security aspects for cloud computing are presented in Section 4.1. Then, in Section 4.2, the mathematical modeling for the network axis is described and discussed with the aim of predicting the demanded extra CPU load needed for confidentiality guarantees in the cloud. Section 4.3 presents both the demonstration and the evaluation of the proposed modeling, and, in Section 4.4, a summary is presented considering this chapter's remarks.

4.1 Network Security in the Cloud

The network security in cloud computing environments considers the communication among services, placed either inside or outside the cloud, for the modes IaaS, PaaS, and SaaS. These communications are commonly established through the Internet, and the security requirements should be considered in agreement with a company's data exposure rules. Confidentiality is necessary here since Internet-based communication is not safe. This caution is also applied inside the cloud, where attackers' hosts could be placed either physically or logically in the same link on the cloud, allowing internal attacks.

The network architecture build in the IaaS mode concerns the communication in the Network Layer of the TCP/IP protocol stack. The security in such a layer is often provided by an IP-based solution where the IP address is the identifier of the end-to-end connection. Regarding establishing a connection, some security mechanisms such as firewalls and authentication tools grant the authorization aspects by controlling and identifying peers. After a connection has been established, the security concern which demands attention is the confidentiality of in-transit data.

The IPSec protocol was designed to be backward compatible with IPv4 and IPv6. This allows the creation of a encrypted network without modification of any intermediate components in the protocol stack. Since it works in the network layer after two nodes setup an IPSec communication (applying keys and certificates appropriately), all communication made through either TCP or UDP enjoy the security service provided by IPSec [Sta10]. In summary, the IPSec protocol encapsulates and encrypts the entire payload of the IP protocol but reuses the same header's information such as the IP address (which is identified by intermediate components such as the routers).

In its turn, the Virtual Private Network (VPN) also aims to guarantee privacy for the communicating process. This concept creates a new private network among the peers by encapsulating not only the application's data but part of the network stack. Although overlapping the network architecture, it is possible to identify some advantages for this model, such as creating an extension for the company's local network (using private addresses) including the public cloud nodes. The IPSec protocol can act as a VPN in the Tunnel mode, where even the IP header is encapsulated within the payload and rewritten. Figures 4.1 and 4.2 depict the protocol stack for both scenarios.

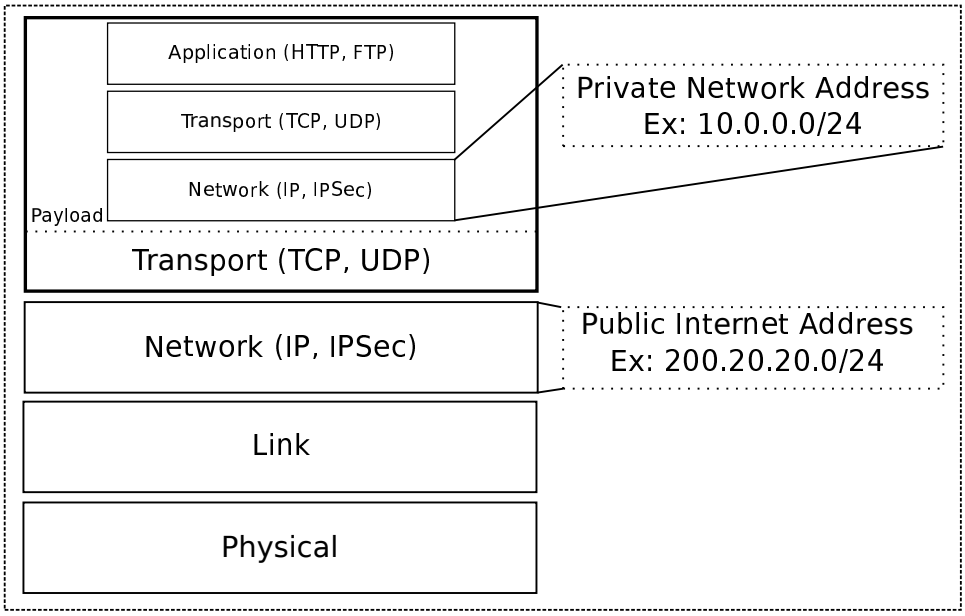

Figure 4.1 – VPN stack layers.

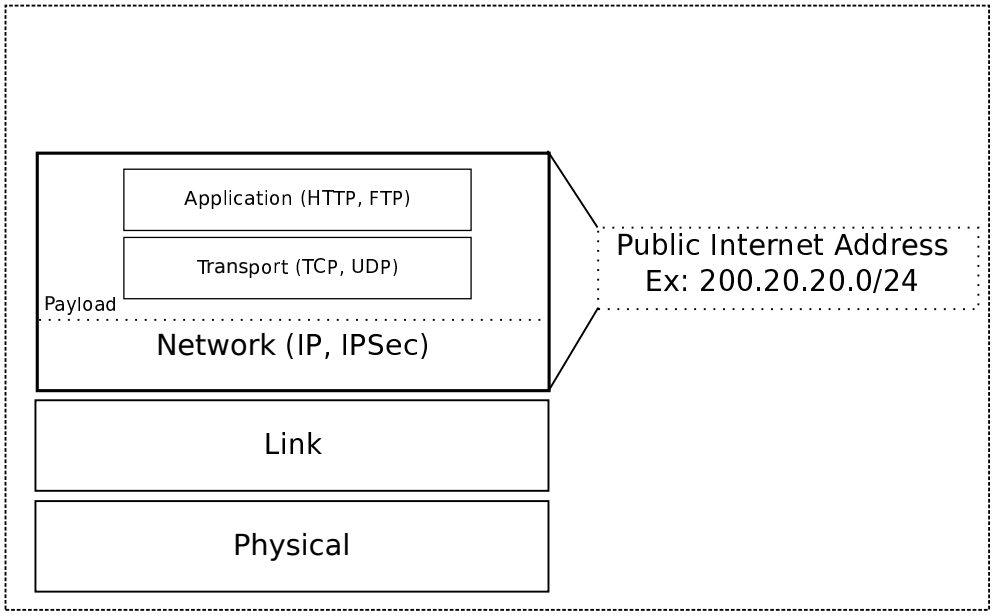

Figure 4.2 – IPSec stack layers.

Regarding the adoption of confidentiality, IPSec and other VPN implementations (such as OpenVPN [Yon]) apply cryptography algorithms for ciphering the payload of a protocol stack [HKH+10, LS11]. The cryptography used for this process is based on symmetric algorithms, which use a single key for encrypting and decrypting the data. This single key is exchanged once, at the beginning of the communication during the authentication process. Some implementation also considers regenerating the communication key using time intervals or communication events, increasing security by forcing an attacker to recalculate the target key. Alternatively, managers can create several channels according to each user's security requirement. A multi-channel design is detailed in [SDRZM15], where the authors present the results in terms of network bandwidth compared to a single channel. The bandwidth increases up to 5 times; even so, the CPU time used for data communication does not exceed 5% using a single processor.

However, IT managers often penalize either the security or the system's performance due to a lack of knowledge about the VPN's cryptography algorithm behavior. The trade-off between network performance and the demanded security level is related to the fact that some cryptography algorithms both increase the data size and take some extra time in the processing phase (queuing the application's communication).

The most common symmetric cryptography algorithms used in VPNs are AES, Blowfish, Camellia, and 3DES. Although they are being used for many implementations, IT managers may consider figuring out the overhead of each one as a model for predicting performance issues and, in consequence, make decisions about the architecture design, considering not only the security level but also the performance impact.

4.2 Cost Modeling

When evaluating the CPU costs of a secure communication, it is necessary to measure each side and to isolate the total sent and received data and measure the encryption and decryption costs, respectively. These costs will be impacted by the chosen cryptography algorithm, and, in order to generalize it, the total CPU time needs to be defined by a metric related to each algorithm.

In doing so, it is possible to write the total CPU time's definition as a function of the cryptography algorithm in encrypting the total sent data and decrypting the total received data, in a certain network structure. The cryptographic CPU time is spread over the whole transmission, since data is ciphered on-demand, as the network packages are dispatched online.

This theoretical model aims to produce a mathematical formula that could be used in simulations to define the total cost of mechanisms supporting secure communications. The target of the model is a perceptual CPU consumption for each node in a communication. At this point, it is possible first define the formula as a CPU time allocation over the total transmission time consumption as:

$$N_{(d)} = \frac{C_{time}(d)}{T(d)} \tag{4.1}$$

where C is the ciphering time and T is the total time in transmitting a certain amount d of data.

Next, it is important to consider handling encryption and decryption functions independently, since different algorithms execute those processes using more or fewer CPU cycles as noticed in the experiments mentioned in Section 4.3. In doing so, if communication between two nodes A and B occurs in one way (A to B), A would consider only the encryption CPU time, and B would consider only the decryption CPU time, disregarding the communication control packages. Therefore, the total data amount d needs to be represented as two variables s (for sent amount) and r (for received amount). In this case, one can rewrite the formula as:

$$N_{(s,r)} = \frac{C_{time}(s+r)}{T(s+r)} \tag{4.2}$$

where s is related to the total sent data and r is the total received data. Now, first decompose the function T and get the time from the partition of the data amount and the network capacity as:

$$N_{(s,r,b)} = \frac{C_{time}(s+r)}{(s+r)/b} \tag{4.3}$$

where b is the bandwidth of the communication. As mentioned earlier, the CPU cycles for each ciphering process need to be considered in different perspectives for the specific

amount of data sent or received. So the formula could be rewritten as:

$$N_{(s,r,b)} = \frac{E \times s + D \times r}{(s+r)/b} \tag{4.4}$$

where E and D give the weight of CPU time for each portion of data, encrypting and decrypting data respectively. Although E is the inverse operation of D, there is a significant difference in terms of CPU time impacting the final results. By adding those values, the formula gives the total CPU time in a network node using a certain cryptography algorithm for sent and received data.

In order to consider a generic formulation of CPU allocation for cryptography, it is essential to extend the formula to different algorithms. Therefore, it is necessary to insert a factor that represents the encryption/decryption cost for each algorithm. Hence, a generic formula could now be rewritten as:

$$N_{(s,r,b)} = \frac{F \times (E \times s + D \times r)}{(s+r)/b} \tag{4.5}$$

where F is the cryptography factor of a given algorithm, E is the function of Encrypting sent data s amount and D is the function of Decrypting the received data r amount. Those functions should be designed to define the CPU time amount on each operation. The function F will produce the CPU time that an algorithm used to cipher data, and this value is divided by the total transmission time in order to achieve the CPU perceptual allocation for safe network operations.

4.3 Evaluation and Validation

The evaluation of the security overhead modeling for the network axis consists of creating both an isolated environment for controlled measurement and a cloud-based scenario for demonstrating the application of the formulas.

4.3.1 Security Overhead Measuring

To evaluate the communication on a Cloud scenario, first of all, it is necessary to develop a controlled environment to isolate uncontrolled variables. For this experiment, two virtual machines were created on a testbed composed of two Intel Xeon E5530 2.4 GHz (4 cores each) and 16GB RAM. Virtualization was provided by the Xen hypervisor v6.2 [BDF+03]. Each virtual machine ran OS Debian 8.2 with kernel 3.16.0-4-amd64, 512MB RAM and 1 VCPU.

To provide secure communication in this environment, OpenVPN [Yon] was used for encrypted connections setup. OpenVPN provides a secure network communication using OpenSSL [VMC08]. OpenSSL offers several symmetric cryptographic algorithms. The set of algorithms, provided by OpenSSL, allows a custom-ciphered communication, since in some situations, even though performance could be compromised, it is important to increase the security level, or vice-versa. The operational concerns about the adoption of OpenVPN to set different levels of security have already been studied and presented in the literature [KK04]. In this work, researchers assessed several tools to provide VPN and concluded that OpenVPN would offer the best solution. Therefore, it would be possible to consider all security possibilities offered by OpenSSL to protect the communication through an acceptable security level.

Hence, in the controlled environment, virtual machines were connected through two OpenVPN tunnels, with bandwidth limited to 1 Gbps. Both tunnels were authenticated with TLSv1.2, TLS_DHE_RSA_WITH_AES_256_GCM_SHA384, 2048 bit RSA. The first tunnel was configured to transfer data without any kind of cryptography. The second one was configured to use different types of symmetric cryptographic algorithms (each experiment used a different algorithm). In each tunnel, 1GB of data was sent, on a client-server application through TCP sockets. The client side captured total time from when it starts to when it receives a final ACK after sending all data, both with and without cryptography. Furthermore, processor usage was also measured in terms of time allocation.

The experiment was conducted for each symmetric encryption algorithm organized by key length: 128 bits (RC2, DES-EDE, BF, CAST5, AES and CAMEL-LIA), 192 bits (DES-EDE3, DESX, AES, CAMELLIA) and 256 bits (AES and CAMELLIA). The encryption modes were CBC, OFB, CFB, CFB1 and CFB8 [AGP+, KRRR98, EKK13].

The results for the first group were obtained when communication occurred, encrypting data using a 128-bit key. In Figure 4.3, one can notice that AES in CBC mode produced the least average overhead impact because bandwidth was the highest (210 Mbps). Nonetheless, the CPU load of AES-CBC was not much different (56%) when comparing with the best CPU load, CAMELLIA in OFB mode (49%). This will give a balance between bandwidth and CPU usage when using AES-CBC. Furthermore, if comparing the different operation modes for AES, *i.e.*, CBC, OFB, CFB, CFB1 and CBF8, it is possible also, for example, to say that, in terms of bandwidth, AES-CBC is 23% more efficient than AES-OFB and AES-CFB modes, and 500% more efficient than AES-CFB1 and AES-CFB8. The cloud costs related to bandwidth are not considered in this work since they are not part of pricing for all cloud providers.

The results for the second group of experiments used a 192-bit key. It is interesting to notice that the CPU load of AES and CAMELLIA was almost constant (less than 7% variation) in CBC, OFB and CFB modes. However, AES in CFB mode presented the best CPU load (52%) and the worst performance in CFB1 and CFB8 modes. That situation indi-

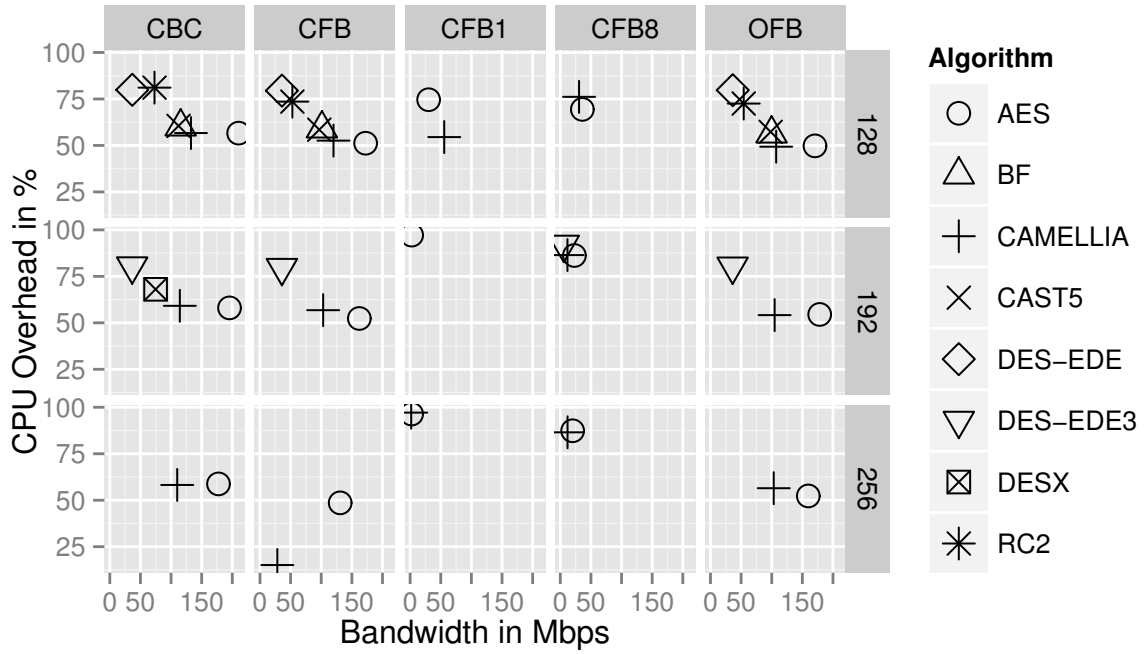

Figure 4.3 – Bandwidth and CPU overhead of different symmetric cryptography algorithms

cated the huge performance impact of shifting mechanisms presented on CFB1 and CFB8 encryption modes. Furthermore, AES-CBC had the best bandwidth per CPU load ratio, and AES-CBF1 mode had the worst ratio. Those results show that the operation mode has to be taken into consideration when using a secure communication.

The last set of experiments used a 256-bit key. The results indicated that using this key size, AES-CBC continued to be the best option, followed by AES-OFB and AES-CFB modes. CAMELLIA presented equivalent values in CBC and OFB modes. However, CAMELIA-CFB produced a very low (the worst) bandwidth usage efficiency, but with the best CPU load rate. Such a mode may be an option for application with low bandwidth requirements. As it was already shown with the other key sizes, CFB1 and CFB8 presented the worst range between bandwidth and CPU load rate.

It is possible to notice that the only situation when AES is not the best option was when using a 128-bit key, since CAMELLIA-CFB1 was better than AES-CFB1 (55Mbps vs. 30Mbps and 54% CPU vs. 74% CPU). The best performance (CPU vs bandwidth) was obtained with AES in CBC mode with a 128-bit key. The worst performance was CAMELLIA in CFB1 mode with 256-bit key. Replacing a 128-bit key with a 192-bit key made AES-CBC decrease to 7% of bandwidth and 3.5% of CPU load. Replacing a 128-bit key to a 256-bit key made AES-CBC decrease to 15% of bandwidth and the same 3.5% of CPU load. This shows that there is a linear relation between cryptography of VPNs and the data volume. Naturally, cloud users and providers should further investigate this relation using larger keys and datasets according their requirements.

The observed values when comparing the cryptography algorithm, the operation mode, and the key size have produced a comparison table which fits the function F of Equation 3.5. Table 4.1 presents the weight of varying the key size and the operation Mode, and each algorithm have a baseline. In a real scenario, one can consider measuring the baseline CPU times and then predict the application of alternative key length and operation mode setups.

Algorithm	Enc	Dec
AES-128-CBC	1	0,8742851376
AES-128-CFB	1,1523485587	1,0626404709
AES-128-OFB	1,070218983	1,0157329126
AES-192-CBC	1,1554807151	0,9829813061
AES-192-CFB	1,2409618295	1,036069918
AES-192-OFB	1,1233924342	0,9106074808
AES-256-CBC	1,2810885594	1,0526752633
AES-256-CFB	1,2554311118	0,9466890275
AES-256-OFB	1,2096260789	0,8878171437
BF-CBC	1	0,9336018075
BF-CFB	0,9388494781	0,7381345507
BF-OFB	0,927076264	0,7567569253
CAMELLIA-128-CBC	1	0,7099748031
CAMELLIA-128-CFB	1,0552358024	0,7872815992
CAMELLIA-128-OFB	1,0250045954	0,7697152656
CAMELLIA-192-CBC	1,2009548507	0,846741928
CAMELLIA-192-CFB	1,189711938	0,8129598678
CAMELLIA-192-OFB	1,1656006894	0,7866807282
CAMELLIA-256-CBC	1,256340577	0,8978393011
CAMELLIA-256-CFB	1,1803523326	0,7983552169
CAMELLIA-256-OFB	1,160345349	0,7910437887
CAST5-CBC	1	0,8938781089
CAST5-CFB	1,0321314016	0,8735323703
CAST5-OFB	0,9623151557	0,8098246713
DES-EDE3-CBC	1	1,0233901873
DES-EDE3-CFB	0,9891461626	1,0038822639
DES-EDE3-OFB	0,9885653418	1,0126936967
DES-EDE-CBC	0,9924265018	1,0220268135
DES-EDE-CFB	0,9841869277	1,0071327864
DES-EDE-OFB	0,9912080889	1,0083969209
DESX	1	0,9687318414
RC2-CBC	1	1,1845378842
RC2-CFB	0,9662279262	0,9820333425
RC2-OFB	0,9614032211	0,97981095

Table 4.1 – Formula weights algorithms baselines.

Furthermore, it is important to notice that, when considering the use of cryptography to send secure data among cloud nodes, not only the algorithm that is chosen has to be considered, but also the operation mode that it is using.

4.3.2 Model Validation

In order to verify whether our model would be applied in real environments, we implemented a scenario using two virtual machines in the Amazon EC2 Cloud infrastructure [Amaa]. Each virtual machine was defined by the *Amazon T3.medium* instance, composed by Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz, 1 vCPU, 3.75GB RAM and Ubuntu Server 14.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. The connection between those virtual machines was performed through two VPN channels. The first one was an insecure channel and the second one was a secure channel using the OpenVPN/OpenSSL tool [Yon]. To simulate the data transference from executions of a standard Cloud application, the Netperf benchmark tool, version 2.7.0, [AMN09] was used and installed on both virtual machines, using a client-server mode.

During the experiment, 500 MB of data was transferred from the client to the server and 500 MB from the server to the client, concurrently. At the end of the communication, the Netperf tool recorded the percentage of CPU usage of each node, for sending and receiving data. The experiment was executed through an insecure channel, without cryptography, and then executed again through a secure channel. All measurements from the secured channel were compared with the measurements from the insecure channel, and only the overhead was considered in this evaluation.

A set of experiments per algorithm was conducted. In the experiments, we varied the operation mode (CBC,CFB,OFB) and the key size (128, 192, and 256-bit), when available. The experiments were conducted for all algorithms and modes in Table 4.1.

The "Local CPU" information provided by Netperf, which is related to the entire host's CPU load, was considered from an average from the two used channels. The Netperf tool also provides the total elapsed time for the transmission; this value was used to compute the total CPU time demanded by each channel. From the execution, all baseline values were used as the F value in Formula 4.5. Using these baseline values, the CPU load for other modes of the same algorithm was predicted.

Since Netperf provides the percentage of hosts' CPU load, and, because it is necessary to have the total CPU time used by each channel to apply Formula 4.5, the baseline CPU time (in seconds) was computed as the product of the CPU load (%) of the sending channel and the total elapsed time. In the results from the experiments for AES-128-CBC, the mean value of total CPU time was 3.438305 seconds. From this value it was be possible to apply Formula 4.5 as:

$$N_{(0.5,0.5,0.01253)} = \frac{F \times (3.438305 + 0.8742851376 * 3.438305)}{(0.5 + 0.5)/0.0233904} =$$
$$N_{(0.5,0.5,0.01253)} = \frac{F \times (6.4443638)}{42.7525} =$$
$$N_{(0.5,0.5,0.01253)} = \frac{1 \times 6.4443638}{42.7525} = 15.074\%$$

where 3.438305 is the model baseline, 0.8742851376 is the factor for deciphering based on values from Table 4.1, and value 1 indicates the baseline factor. The same principle was used to apply Formula 4.5 to the modes in Table 4.2, which are predicted values. Table 4.2 also presents, in column "Real", the measured values in the real scenario.

The comparison of the measured values and the modeled values are presented in Figure 4.4 and Figure 4.5. These values were calculated to validate our model, and, as can be seen in Table 4.2, the predicted results have an accuracy superior than 90% in most cases.

Figure 4.4 shows that the prediction model accuracy is superior than 91% for most operation modes except for 256-CFB and 256-OFB, which show also higher error rate for real measurements. On the other hand, in Figure 4.5, the prediction model applied for the Camellia algorithm presented an accuracy of 91% for CAMELLIA-256-CFB, other modes were predicted with accuracies bellow this metric.

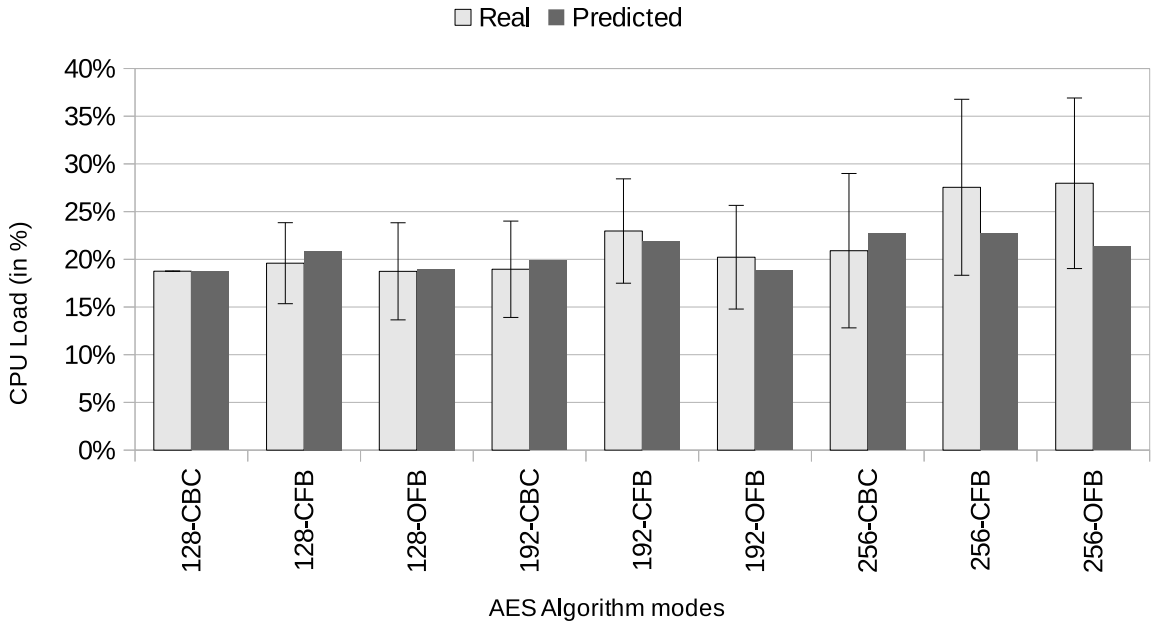

Figure 4.4 – Model Validation for AES Algorithm: comparison between real and modeled overhead by using cryptography in data communication among Cloud instances.

As was presented in this section, our model can represent real scenarios' behaviors for most algorithms and operation modes. Nonetheless, we could also notice that

Algorithm/Mode	Predicted	Real	Error
AES-128-CBC*	18,75%	18,75%	0,00%
AES-128-CFB	20,86%	19,60%	6,46%
AES-128-OFB	19,00%	18,74%	1,35%
AES-192-CBC	19,85%	18,96%	4,72%
AES-192-CFB	21,86%	22,97%	4,81%
AES-192-OFB	18,84%	20,22%	6,83%
AES-256-CBC	22,69%	20,90%	8,55%
AES-256-CFB	22,73%	27,55%	17,49%
AES-256-OFB	21,31%	27,97%	23,80%
BF-CBC*	38,11%	38,11%	0,00%
BF-CFB	30,64%	31,81%	3,66%
BF-OFB	30,66%	32,15%	4,64%
CAMELLIA128-CBC*	30,58%	30,58%	0,00%
CAMELLIA-128-CFB	35,00%	30,20%	15,91%
CAMELLIA-128-OFB	32,96%	29,72%	10,92%
CAMELLIA-192-CBC	39,27%	34,24%	14,71%
CAMELLIA-192-CFB	38,30%	31,41%	21,94%
CAMELLIA-192-OFB	36,71%	30,63%	19,85%
CAMELLIA-256-CBC	42,43%	34,02%	24,73%
CAMELLIA-256-CFB	38,28%	35,10%	9,07%
CAMELLIA-256-OFB	37,38%	33,58%	11,31%
CAST5-CBC*	31,48%	31,48%	0,00%
CAST5-CFB	32,34%	33,74%	4,13%
CAST5-OFB	29,32%	36,36%	19,37%
DES-EDE3-CBC*	56,82%	56,82%	0,00%
DES-EDE3-CFB	56,17%	63,13%	11,03%
DES-EDE3-OFB	57,17%	63,20%	9,54%
DES-EDE-CBC	56,42%	55,49%	1,69%
DES-EDE-CFB	55,84%	58,67%	4,83%
DES-EDE-OFB	56,94%	59,40%	4,15%
DESX-CBC*	48,76%	48,76%	0,00%
RC2-CBC*	59,52%	59,52%	0,00%
RC2-CFB	59,45%	51,19%	16,12%
RC2-OFB	61,96%	58,19%	6,48%

Table 4.2 – Algorithm CPU load prediction comparison. (* baseline values are not predicted.)

extra variables can be included for accuracy improvements for specific algorithms, for example, for the Camellia algorithm. However, since AES is the most-used algorithm in HTTPS or SSH communications [LMN07], and for the proposed model it had achieved high accuracy, this implies that our model can be used for SLA estimation in real Cloud environments adopting VPNs.

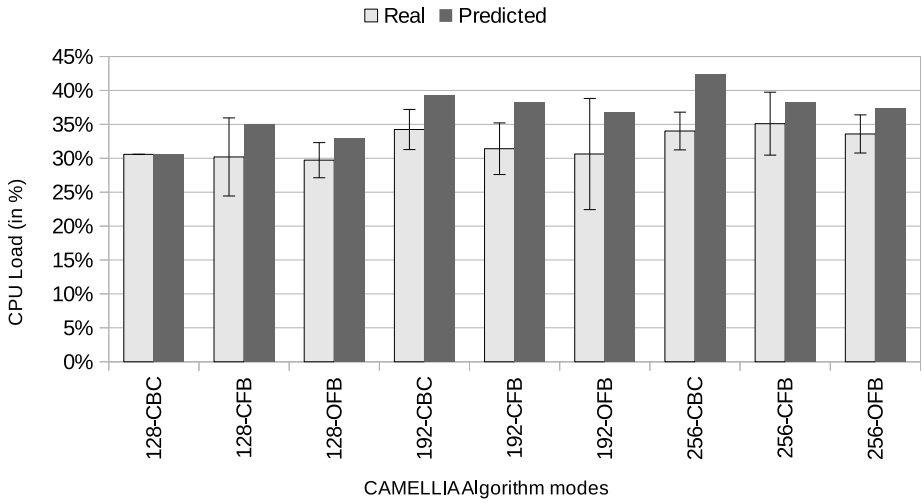

Figure 4.5 – Model Validation for CAMELLIA Algorithm: comparison between real and modeled overhead by using cryptography in data communication among Cloud instances.

4.3.3 Mathematical Validation

The model proposed in this chapter mainly supports the relation between the CPU load and the data volume to predict overhead allocation in cloud computing environments. The previous validation consists of defining a constant F which represents the difference among the algorithms. In this Mathematical Validation, a multi-linear regression[1] over the measurements will be used for fitting the Encryption and Decryption variables. These variables are represented by the β values of a regular multi-linear formula, which aim to find the CPU load as the result of the data volumes (sent and received) within their respective weights. A general formula can be written as:

$$CPU = \beta + \beta_1 \times s + \beta_2 \times r \tag{4.6}$$

where β_1 represents the weight for encryption, and β_2 is for the weight of decryption. The variables s and r are, respectively, the data amount sent and received.

Based on this observation and following a similar evaluation, a new experiment was conducted with the objective of (1) executing an OLTP benchmark and measuring the CPU overhead, and (2) verifying the relation of CPU and the data volume as a linear regression for predicting different scenarios, applying the results to fit the formula variables.

[1]Multiple linear regression attempts to model the relationship between two or more explanatory variables and a response variable by fitting a linear equation to observed data. http://www.stat.yale.edu/Courses/1997-98/101/linmult.htm.

This validation considers a benchmark execution over a cloud environment composed of two virtual machines (VM) which communicate through a local network, and their discs are provided by the virtualization layer, in this case using Xen Server 6.2.

The benchmark is a TPC-C [LD93] implementation, producing a workload from one VM over a standard MySQL instance hosted into the second VM. For each execution, the database is re-populated, and both the communication and storage data volumes are profiled, as well as the CPU load percentages of the Database process and the entire node. These metrics are produced in two execution scenarios: (1) without any cryptography, (2) with a Virtual Private Network (VPN). These two scenarios are also stressed for the benchmark, where the executions are from 1 to 10 simultaneous connections.

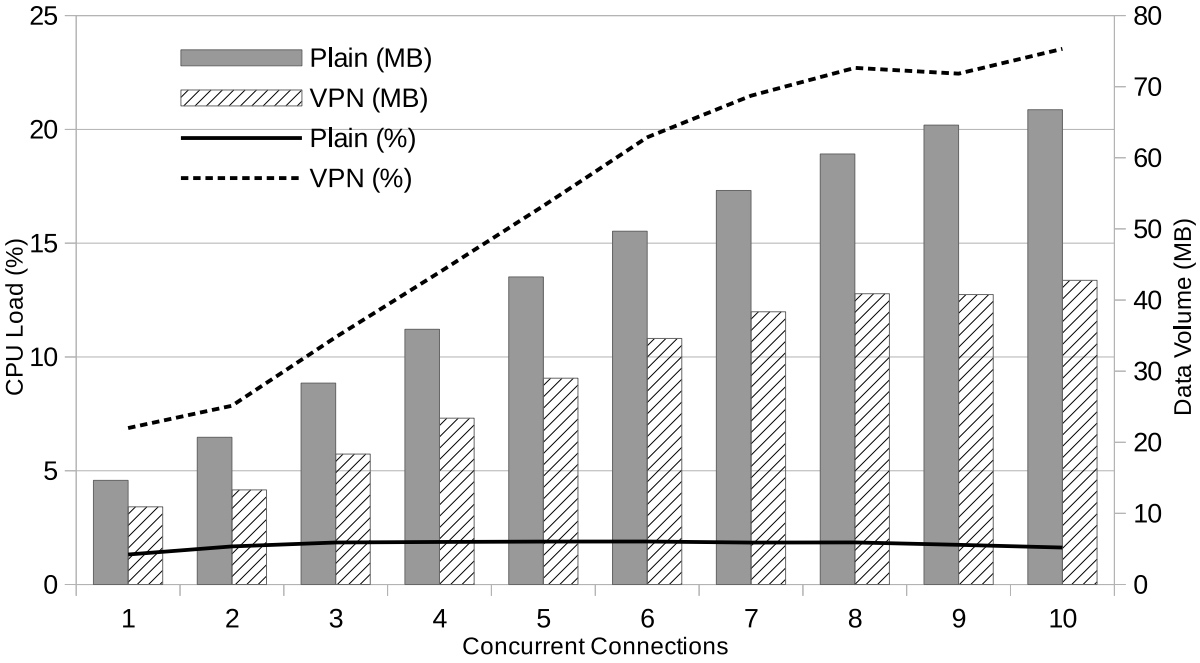

Figure 4.6 – CPU overhead demonstration for increasing demand and data volume.

Figure 4.6 shows an increased load of CPU as more concurrent connections are added to the VPN scenario. It is also possible to observe a decreased data volume transmitted, impacted by the VPN overhead. The VPN CPU load is related to the entire node, discarding the CPU load of the database process. By subtracting the plain CPU load from the VPN scenario, it is possible to acquire the measured CPU overhead of adding the VPN in the communication of the benchmark. The multi-linear regression considered the CPU overhead as a function of the sent and the received data volume, where the R-squared values were 0.9991.

Figure 4.7 presents the comparison of different VM's memory size, where the network data volume was modeled in function of the overhead of adding a VPN. It is possible to observe a difference in the model accuracy in execution with few concurrent connections. In these scenarios, the CPU was not stressed, allowing some variations in CPU

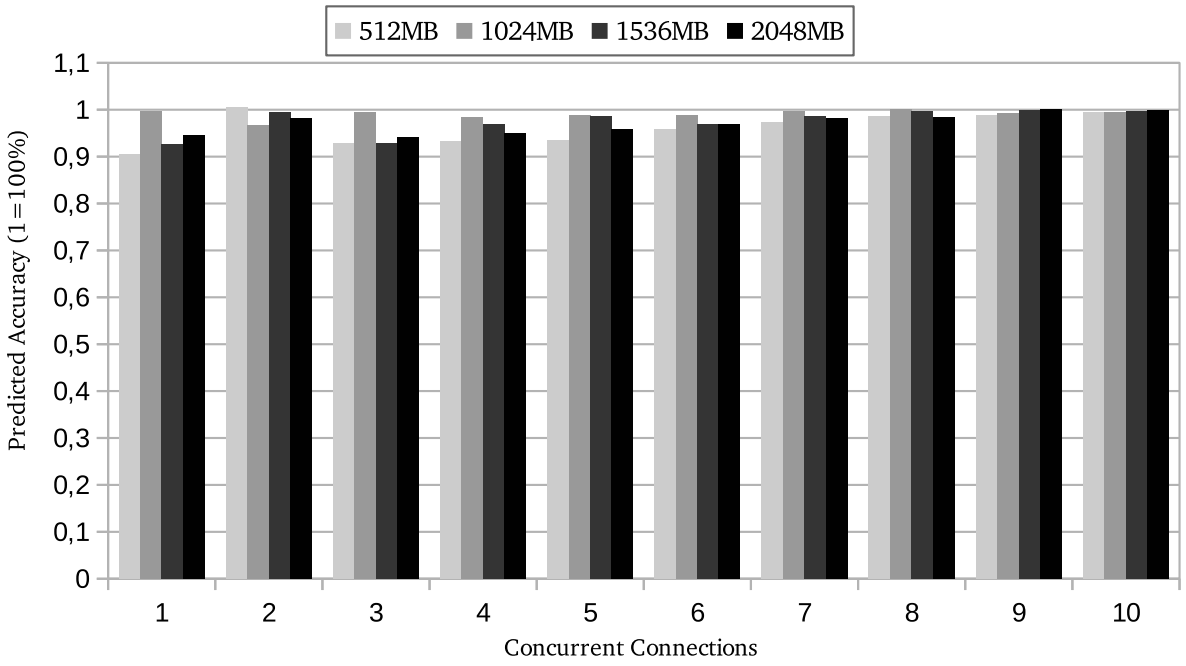

Figure 4.7 – Predicted values for different VM's memory size based on multi-linear regression.

utilization due to the OS scheduler policies. However, for the stressed scenarios, the average error was not superior to 5%, which validates the relation between data volume and CPU load when using security for networks based on cryptography algorithms.

4.4 Summary

Communication security for cloud computing scenarios is well-established for almost all solutions from government agencies to standard users. The variety of solutions and implementations often grants the confidentiality principle based on cryptography algorithms such as AES, Camellia, 3DES, and Blowfish. With the aim of choosing an efficient algorithm, cloud users or providers may yearn to predict the demanded CPU allocation for supporting cryptography during the communication flow for estimating costs, relocating parts of the application, or even choosing an alternative for cryptography within company's Privacy Level Agreements.

This chapter presented a study of the cloud communication architecture to detach security aspects and measure the impact of the confidentiality principle. Based on an evaluation of VPN overhead for different cryptography algorithms, the developed modeling considered the relation between data volume and CPU allocation. This modeling was validated by a set of experiments in a cloud environment where predicted values accuracy was from 91.45% to 98.65% for the AES algorithm. The modeling in this chapter is also validated for an OLTP benchmark execution, where the communication of the bench-

mark's loading process was measured and used for fitting a multi-linear regression. The multi-linear model considers the function of the network's sent and received volumes to explain the VPN's CPU overhead. The regression had an R-squared equal to 0.9991 and could predict scenarios with different VM's memory sizes with an accuracy close to 95%.

The network axis model is part of the full-stack confidentiality model we are developing in this research, and it is part of validation for the hypothesis *(ii) confidentiality costs can be modeled and used in the sizing of cloud computing environments*, by answering the research questions:

- *2 - What is the overhead for adding confidentiality mechanisms in the cloud computing stack?* Although cloud computing environments already have security for in-transit data, we have shown how the security is applied in this axis and what the impact of implementing cryptography algorithms in the communication flow is. After being identified, the cryptography algorithms were evaluated in terms of CPU allocation to communicate in a cloud environment, and the overhead was explained by the relation between data volume and CPU allocation.

- *3 - How can the overhead of combined security mechanisms for communicating, storing, and processing data in a cloud environment be estimated ?* The modeling presented in this chapter has as input the characteristics of the cryptography (including the algorithm and the key sizes) used in the communication as well as the data volume transferred by users' application. From these values, the modeling could be fitted to predict the extra CPU needed in the environment to support the service level of the application with confidentiality for data in-transit.

5. STORAGE CONFIDENTIALITY MODEL

The increased data volume in the BigData era, fed by the tremendous amount of connected devices and global Internet services adherence, have been motivating IT managers look for cheaper, heterogeneous, and distributed persistence services. In addition to their volume, data generated in areas such as business, health care, and social media have an augmented importance nowadays, from buyer tracking for sales suggestions to disease-symptom evaluation for public health predictions.

Regarding data confidentiality, cloud computing providers have been offering encryption of data at rest, where both the persistence flow between users' virtual machines and the storage unit and hard disks are ciphered using symmetric cryptography algorithms [Gooa]. This feature is identified in the full-stack confidentiality architecture as the second axis to be considered to promote the confidentiality principle for data stored in public cloud providers.

Although confidentiality is essential in many applications, and ciphered persistence have been improved in recent years, the impact of adding a security layer in the persistence flow should be accounted since data storage is a common bottleneck in computational systems. By identifying the common implementations of confidentiality used in cloud providers and understanding the overhead in the persistence flow, it is possible to use a model to predict the extra computational resources needed to efficiently provide this service, helping IT managers decide among alternatives to support institutions' Privacy Level Agreements.

In this chapter, in Section 5.1, the security aspects and implementations are presented and discussed. Next, in Section 5.2, a modeling for predicting the security impact in clouds' storage systems is proposed, especially considering the confidentiality principle's impact in terms of CPU allocation. The proposed modeling is evaluated in Section 5.3, where both a performance analysis and a mathematical evaluation are used. Finally, Section 5.4 summarizes this chapter's content as well as its findings and results.

5.1 Storage Security in the Cloud

Cloud storage services are used for persisting both users' data and applications' files. Cloud storage mechanisms are commonly managed by the virtualization technology that hosts the user's virtual machines and the cloud services.

To allow data confidentiality, the persistence layer should also support the usage of both authentication and cryptography tools. Considering the IaaS cloud service model, in most public clouds, VM's volumes are stored in shared spaces (*i.e.* managed by LVM)

and encrypting them is an extra feature supported by some providers [Gooa, Amab]. For PaaS and SaaS, providers have also started to support encryption mechanisms using standard cryptography algorithms like the AES, such as the Amazon's S3 object storage [Amad] and Azure's Blob Storage [Mic]. In order to support privacy in a public cloud environment, alternatively, some works [KL10, PLM+11] present solutions for safely persisting data on public cloud environments. Those solutions adopt data integrity and availability principles in terms of keeping data enciphered and non-touchable based on theoretical proofs.

The impact of adding security to the persistence layer should be considered by the user since it requires computational resources which would be billed by the cloud provider. Otherwise, leaving data in plain text stored in a cloud environment would leverage information leakage. The following sections present two confidentiality solutions for storage services, considering both SaaS/PaaS and IaaS.

5.1.1 Object Storage

Considering the storage services available in current cloud providers, the Object Storage (Amazon AWS S3, OpenStack Swift) is an alternative for the SaaS service mode. This service abstract from users some aspects such as the infrastructure scalability or availability, delivering a simplified interface for storing or retrieving byte-based objects (files, figures, videos, documents, etc) to the user. The interface is developed for delivering authentication and authorization features as well as enabling the confidentiality principle by adding cryptography algorithms.

The cloud software architecture should change its persistence flow and, instead of reading and writing files into a file system, it should send/receive data into/from a cloud service, normally through the Internet. For instance, a Photo Gallery web-application based on cloud services will allow authenticated users to upload files and, unlike hosting a user's file within the web server, the application can redefine the upload target into an Object Storage, relaying storage management to the cloud provider.

Regarding confidentiality, the cloud storage service commonly allow enabling data encryption either on the client-side or on the server-side [Amad]. In the client-side encryption mode, the application should manage the cryptography keys, matching it within the enciphered data. In the server-side encryption mode, the key management is also delivered as a service by providers, demanding no modification in the application. Although the server-side mode encryption is simple to implement, all users' keys are stored in the cloud, which might not be in accordance with a company's security rules.

The back-end architecture commonly uses regular storage systems, in some cases shared with IaaS services. Figure 5.1 depicts the elements for the OpenStack Swift Object Storage [Ope] (similar to the Amazon AWS S3), and it is possible to observe a

software stack upon the storage discs, where data are persisted. For instance, the cryptography application could be placed either in the client component or in a server-side component, for Swift in the *Proxy Server* layer.

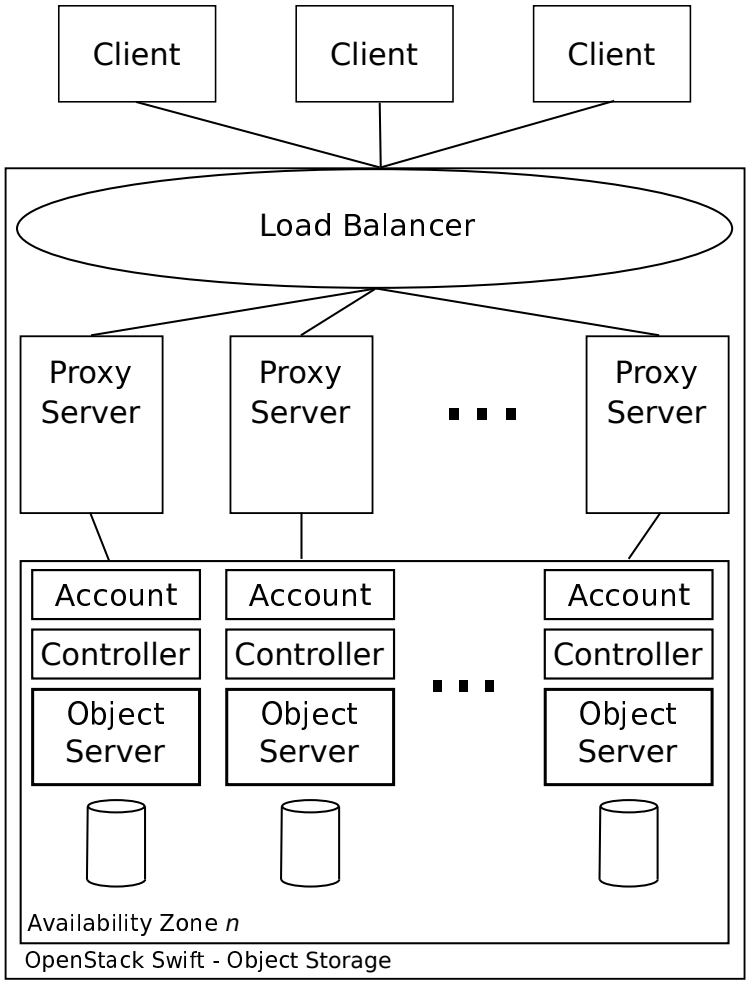

Figure 5.1 – OpenStack Swift architecture.

5.1.2 Cryptography File Systems

Cryptography File Systems (CFSs), earlier introduced by *The Design of a Cryptography Based Secure File System* [Gud80] and *A Cryptographic File System for Unix* [Bla93], have been used to provide persistent data with confidentiality. The main feature of this mechanism consists of providing encrypted stored data with no application modification. The application uses standard I/O instructions since the CFS works in lower systems' layers.

Most cloud providers build their service on top of regular data centers. The storage system is often provided by a centralized per-rack storage unit and on-host discs. The hardware storage is managed by the virtualization layer that places the virtual machine

(VM's) disc images according to rules of the virtualization player. These disc images are attached to VMs, and the images are accessed like regular discs by the guest operating system running inside that VM. Figure 5.2 shows a regular I/O stack for Unix-like systems, where in the Block Layer the Virtualization technology adds extra sub-layers in order to access shared physical resources managed by the virtualization software. In such an environment, the CFSs operates in layers between the application and the Block Layer without knowledge of the virtualization layer.

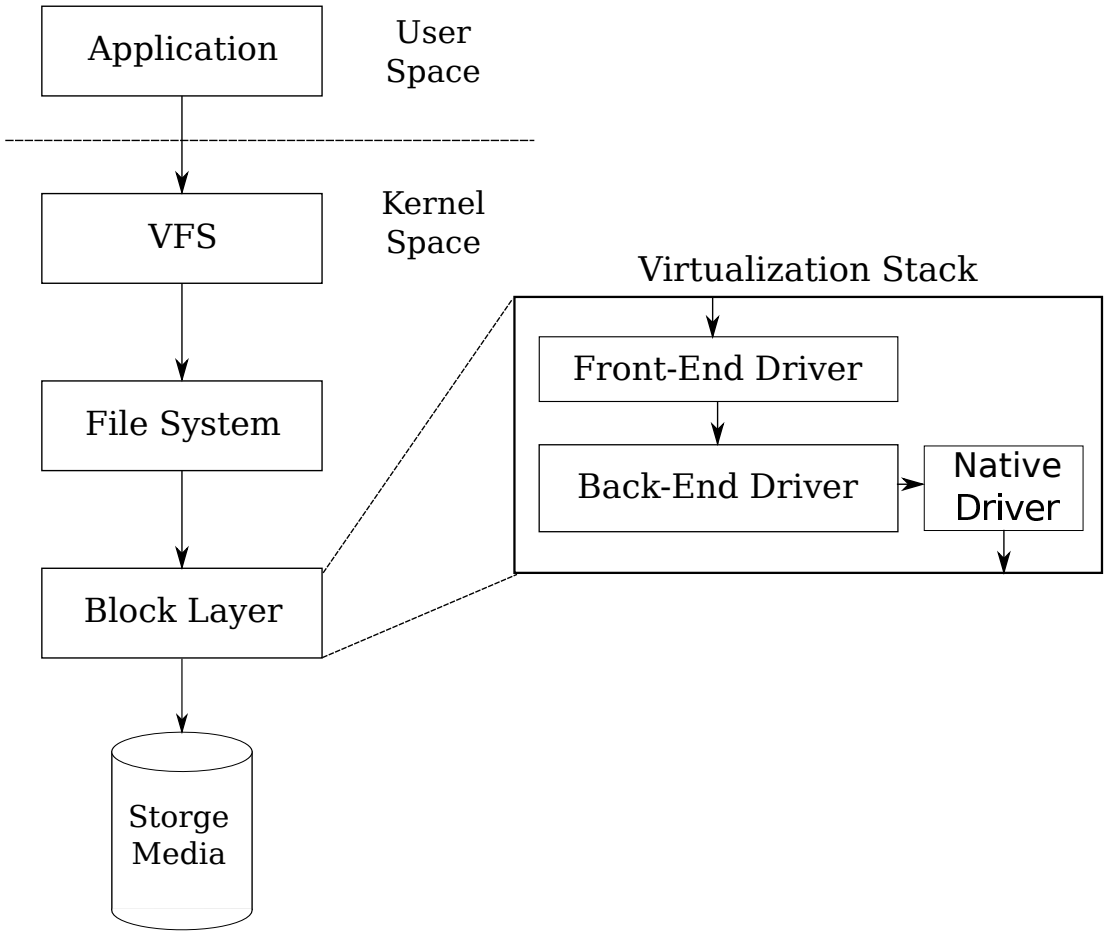

Figure 5.2 – I/O stack in Unix-like systems over a virtualized environment.

Similarly, cloud providers offer storage as a service through internet-based communication, *e.g.* HTTP and RESTful. Those systems are also called WebServices and are also handled by the virtualization layer (to support service elasticity). Some of these services support security principles using cryptography and secure protocols.

Cloud storage systems are composed of layers which are implemented according to the application, i.e. a guest operating system hosted into a virtual disc, which is managed by a virtualization layer over a storage service managed by a Network-Attached Storage. Confidentiality mechanisms can be applied to different levels of the storage stack, and doing it in at least one is enough to avoid data leakage.

When applying confidentiality for storing data inside the virtual machine, the user is responsible for choosing the cryptography algorithm and also for managing the keys, having total control of the security. The cryptography mechanism could be applied either internally in the user's software or using a persistence mechanism with cryptography support, *i.e.* the Cryptographic File Systems.

Most CFSs use symmetric cryptography due to performance issues and could be implemented with different levels of abstraction [WDZ03]:

Block-based System

The block-based storage system has cryptography support by handling one disk block at a time. There is no knowledge of upper layers, such as files or directories, and it could even be used by software that needs to access raw partitions (such as databases management systems). In other words, the persistence flow is intercepted by a cryptography phase which applies the ciphering to each block in the raw device, as shown in Figure 5.3. In the Linux systems, the Device-mapper crypt [Fru05] is a kernel module which provides block devices using the kernel's crypto API [Cor]. This API runs in kernel privileged memory area, where also all instructions and keys are maintained.

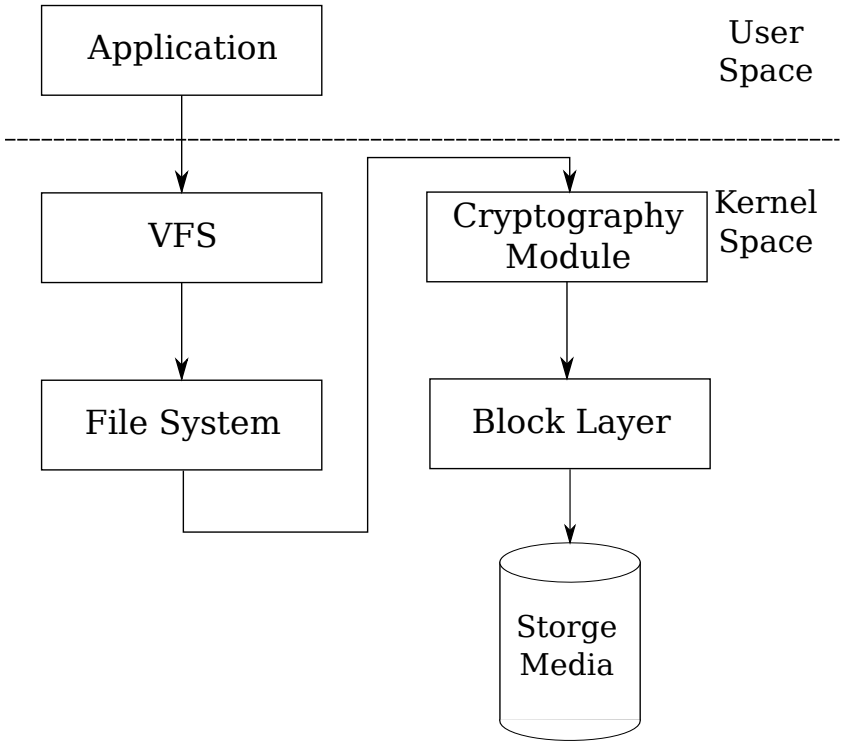

Figure 5.3 – I/O stack in Unix-like systems with Cryptography read/write applied directly the block storage unit.

Stackable File System

Stackable cryptography file systems (SFS) are placed on top of the regular file system, and usually support multiple operating systems since the software layer which intercepts the data flow for encryption is independent of the persisted data. The ciphering process uses a regular file system as the lower layer for hosting the directory-file structure and modifies only the files' content and optionally the filename (or directory name). Figure 5.4 shows a Cryptography Module running in a User Space layer that handles applications' files, storing them into another File System.

The EncFS [Gou] is a Linux-based and was created as a virtual encrypted view for a directory in a regular file system in the user-space. It runs without any special permission and uses the FUSE (File System in Userspace) library. The key management is made through users' configurations (but never stored), and it is prompted every time the encrypted directory is mounted. During the mounting phase, a new key is derived from the original within a certain number of rounds, which is described in the user's configuration file. The primarily goal of this file system is to protect data off-line and it supports strong cryptography algorithms, such as AES and 3DES, for standard OS's users. Similarly, the eCryptFS [Hal07] builds a cryptographic file system over a regular directory structure using algorithms such as the AES with different key sizes. In this case, the key is managed by a regular Linux key ring, and the ciphering processes are relayed to the kernel module Crypto API. All algorithms supported by this kernel module are also eligible to be used by eCryptFS.

When comparing Block and Stackable models, the former may achieve better performance due to fewer layers between application and the storage hardware. However, a performance comparison between those two models should be evaluated according to the above applications' behavior. Despite the model architecture impacting the overhead, the main impact on the performance is caused by the cryptography algorithm.

5.2 Cost Modeling

The application of confidentiality in storage systems, supported by the cryptography modules presented earlier, is often based on a regular symmetric algorithms such as AES and BlueFish. However, the overhead of storing data with privacy is not only based on the performance of the cryptography algorithm. It is necessary to consider the architecture of the persistence flow, the IO subsystem, as well as the encryption and decryption operations in such software stack. Additionally, the storage system often uses cache mechanisms for performance improvement that needs to be paired with the cryptography module.

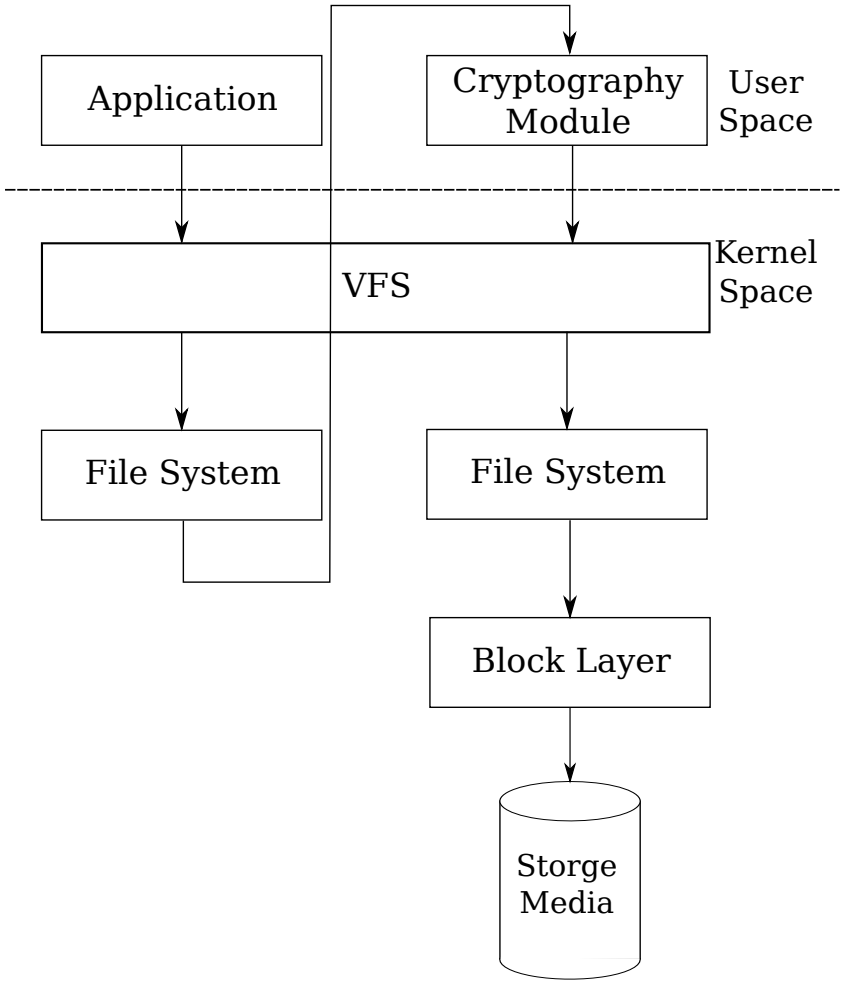

Figure 5.4 – I/O stack in Unix-like systems with Cryptography module read/write in a File System.

Currently, cryptography algorithms are achieving reasonable performance, especially using modern processors [Int06]. On the other hand, the IO subsystem's performance still depends on physical aspects for persisting data. These two characteristics make the security storage systems have a chained dependency, in which the persistence flow throughput is affected by the cryptography instructions, and, in turn, the cryptography performance is affected by the IO subsystem queuing, which still is a main bottleneck in computational systems.

But how much the cloud storage impacts the CPU usage, what the impact of cryptography on the persistence flow is, and how to predict the overhead of adding confidentiality in the storage layer are some of the questions that need answered for better allocation of cloud computing resources. For this reason, it is crucial to model the cost of using cryptography in the cloud storage environment. Specially the impact in CPU usage since cloud billing is commonly based on CPU units (such as cycles or hours).

As mentioned earlier, the CPU time used by cryptography algorithms is spread along the IO queue time. In doing so, one can first deduce an inverted relation of the total

CPU time consumed by cryptography for a certain data amount and the time to persist it. This relation can be written as:

$$S_{(d,t)} = \frac{C(d)}{t} \tag{5.1}$$

where d is a data amount, C is the function for expressing the CPU time for that given data amount d, and t is the total time of the persistence flow. The formula would be simpler if a single one-way direction (reading or writing) is considered. However, it is necessary to identify and split the data amount d in write and read operations to make it adherent to real scenarios. It is also necessary to consider the significant difference in reading and writing throughput to estimate the total time of the persistence process. In doing so, the formula could be rewritten as:

$$S_{(w,r,TP_r,TP_w)} = \frac{E \times w + D \times r}{(r/TP_r) + (w/TP_w)} \tag{5.2}$$

were w and r are total data written and read, respectively; E and D are the function weights for estimating the CPU time of encrypting and decrypting an amount of data. These values will produce the total CPU time for the ciphering process. Also, the TP variables are related to the throughput of the reading and writing operations in IO subsystems. Although it would be a reasonable abstraction to simply add the cryptography's CPU time to the IO time, the machine's memory size has also a significant impact on the application's total time since the file system's cache subsystem allows better performance of the cryptography algorithms' utilization. The memory size m impacts the total time of the IO subsystem, and it is represented as:

$$S_{(w,r,TP_r,TP_w,m)} = \frac{E \times w + D \times r}{\Delta(r/TP_r + w/TP_w)} \tag{5.3}$$

where the variable m represents the available memory for the IO subsystem and needs to be considered for two scenarios: the total dataset size being smaller than the available memory, or not. Δ is dependent of a condition within m as:

$$\Delta = \begin{cases} F(r/TP_r + w/TP_w, m), & r + w < m \\ 1, & r + w >= m \end{cases} \tag{5.4}$$

where the F function in the formula gives the index of the extra performance for in-memory operations. This behavior could be observed during the experiments (later presented) where a file with a size smaller than the host's available memory achieved at least 3 times higher performance.

As a summary, to apply the formula values for the following variables are needed: the amount of data read (r) and written (w), the function costs of encrypting (E) and decrypting (D) each king of persistence flow, and the throughput of the IO subsystem. Throughput needs to be measured for two resources: the read and write operations into

raw-external devices including virtual disc of virtualization layer, and the read and write operations into memory, which will fit the F function, as presented in the Formula 5.4.

The presented model estimated the CPU allocation of a host (or virtual cloud host) for a certain data amount. This value helps to predict the overhead in a host when confidentiality is added in the persistence flow for privacy guarantees. The system administrator could trace a company's application I/O data amount and use this formula to estimate the CPU allocation per node.

5.3 Evaluation and Validation

In order to evaluate the proposed model, initially, a set of experiments was conducted to fit the mathematical model, considering the impact of memory, the absolute CPU load and the relation between dataset's size and execution time. Next, the data volumes (read and write) of an OLTP benchmark was used to do a multi-linear regression fitting, still considering the relation to the CPU overhead.

5.3.1 Environment Description

The validation environment was composed by virtual machines running standard Linux using a single virtual processor and variable memory sizes (256MB, 512MB, 768MB, and 1024MB). The virtual machines were hosted in a physical server with an Intel Xeon X6550 2GHz 8-core processor, 96GB RAM, a local SATA disk with 128GB, and XenServer 6.2. For these experiments, the Cryptography File System approach is used because its adoption does not demand application modification, and it is considered a low level high-performance solution for the cloud software stack. The Cryptography File Systems used in the experiments are the dm-crypt [Fru05], placed in the kernel level and acting as a Block CFS; the EncFS [Gou], running in user-space with the FUSE library, acting as a Stackable CFS; and the algorithm was always set to AES with a 256-bits key. This algorithm was chosen due to its recommendation by security institutes [BHW06]. This configuration was evaluated by the IOZone [NC] I/O benchmark tool.

5.3.2 Validation

This evaluation defines the weights for the variables and functions from Equation 5.3. The variables E and D in the formula are the processing time for encrypting and decrypting an amount of data in bytes, respectively. Considering that the CFSs used in

this validation use a standard cryptography algorithm, these values could be captured from an isolated test case (*i.e.* measured with command line tools) and applied in a linear function. The measured values for E and D for a 500MB file were respectively $4558ms$ and $3956ms$. It is noticeable that the encryption operation consumes more time compared to decryption. This operation is also essential during the writing processes, which is far more expensive than reading in terms of time for the I/O subsystem. When combining both encrypting and writing operations, the worst case scenario for this experiment is presented. So, the next experiments would consider only writing operations in order to demonstrate the model application.

In Equation 5.3, the throughput variables are determinant in the overall formula application. In order to measure them, an experiment was conducted to write files in a CFS with sizes from 10MB to 1000MB, and the host CPU consumption is captured for four memory sizes: 256MB, 512MB, 768MB, and 1024MB. Figure 5.5 shows differences in CPU allocation according to the file size and the host memory size. In general, the CPU load was near 10% for files with size up to 68% of the available memory. This experiment shows memory consumption during the encryption phase. Otherwise, for bigger files, the I/O subsystem queues the work-flow, not allowing the progress of the encryption. In such a scenario, the formula would be impacted by the host's memory size, which is mapped in the mathematical model as the Δ factor. This factor is conditioned by the function in Equation 5.4, where the total execution time is affected by a factor F when persisted data is bigger than available memory size m. The reduced CPU load is also reflected in the total execution time, as it was observed during the test executions.

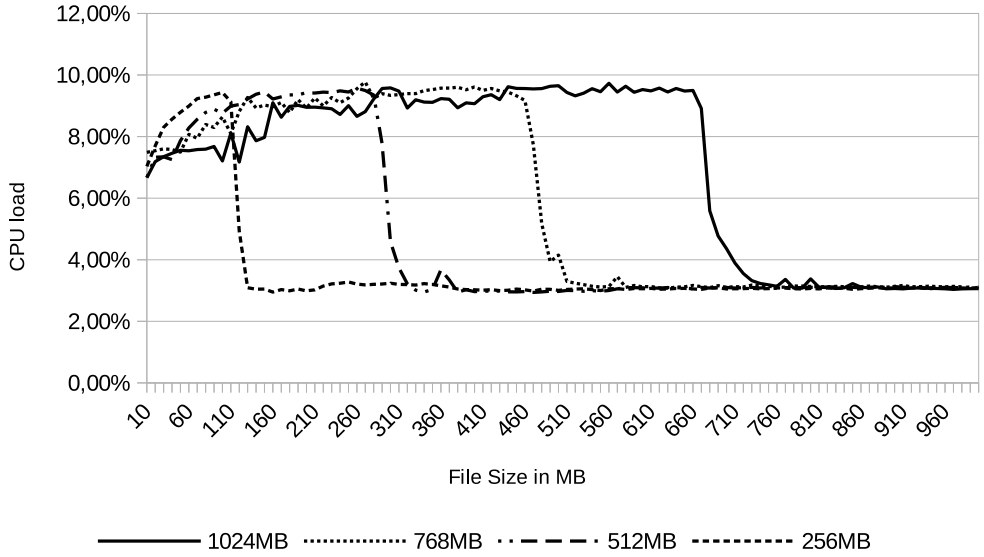

Figure 5.5 – Memory size effect in overall CPU load during CFS persistence

When isolating CPU overhead with or without memory as a support to improve performance, it is possible to demonstrate a stabilized load due to the IO subsystem's limitation. Figure 5.6 presents a comparison between two cryptography key sizes for writing

files from 10MB to 1GB in a VM with 512MB of memory. It is possible to see that, although the key length increases, the CPU allocation follows a standard behavior.

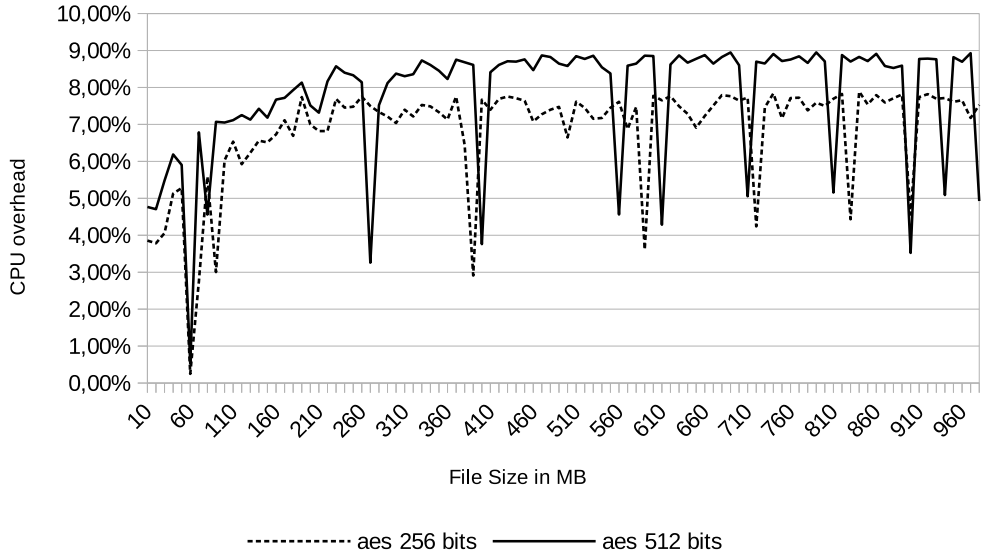

Figure 5.6 – CPU overhead comparison for AES with different key lengths

5.3.3 Mathematical Evaluation

Similarly to network modeling, the behavior of the cryptography persistence flow also considers the relation between CPU load and data volume. This relation is verified in the validation presented earlier for files sizes smaller than the available VM's memory.

Based on the previous experiments, a new set of experiments were conducted for an OLTP benchmark with and without a CFS. The goal of this experiments is to produce a multi-linear regression to explain the CFS CPU overhead. through the function of the amount of bytes read and written(REMOVED).

The used benchmark software is a TPC-C [LD93] implementation producing a workload over a standard relational database instance with data being persisted in two different directories. One of them is a raw file system and the second one is a dm-crypt [Hal07] file system (CFS). For each execution, the database is repopulated into the desired file system (with and without cryptography). These two scenarios are also stressed with concurrent connections from 1 to 10.

Figure 5.7 shows a CPU load comparison for scenarios with and without a CFS. In one hand, for the plain mode, without CFS, there is an increase in the data volume as more concurrent clients are added to the execution. However, there is no significant variation in CPU load. In the other hand, the CFS scenario increases the CPU utilization with the addition of concurrent connections since the data volume also increases. However, the

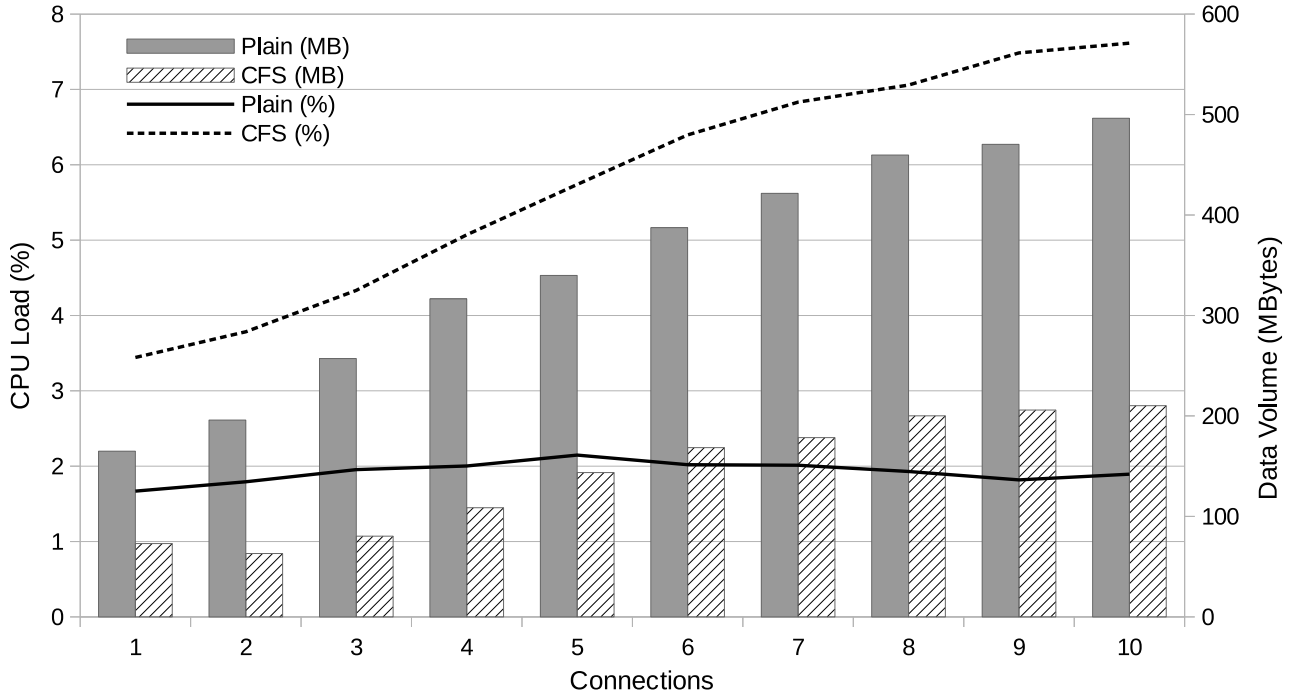

Figure 5.7 – CPU load variation for an OLTP benchmark with and without CFS

data volume is significantly inferior to the Plain scenario. The CPU load and the data volumes of the CFS scenario are used to fit the linear regression where R-squared was 0.9982. The linear regression considers the variables *w* (write) and *r* (read) representing the data volumes for the respective operations. The values are then guessed form $CPU = \beta + \beta_1 * w + \beta_2 * r$. This modeling was then used to predict the CPU overhead of CFS for four scenarios, where the VM's memory size was modified to 512MB, 1024MB, 1536MB, and 2048MB, always using a single VCPU. It was possible also to observe that the database management system does not access the entire file taking benefits from a cache subsystem, so the memory size effects observed in the previous validation were not present in this experiment.

Figure 5.8 depicts the relative difference between the measured CPU overhead and the prediction, based on the fitted values. It is possible to observe an error not superior to 10%, and in most cases, the accuracy was close to 95% in average.

5.4 Summary

Cloud storage confidentiality has been considered an essential feature for achieving companies' security requirements. The current implementation for persisting data on cloud environments commonly adopts cryptography algorithms to provide confidentiality for data at rest, including operating systems files, users documents, etc.

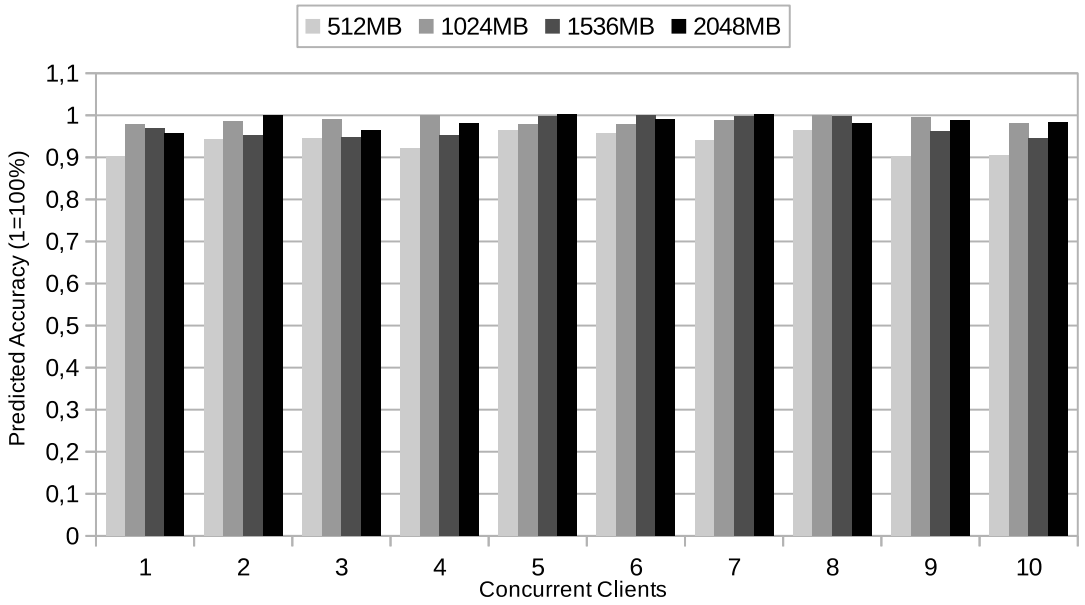

Figure 5.8 – Predicted CPU overhead for different memory sizes, based on multi-linear regression.

The modeling presented in this chapter considers the relation between persisted data volume and the CPU overhead introduced by the cryptography algorithms. In a different perspective of the network modeling presented in the previous chapter, there is a extra variable for the cloud storage system related to the IO subsystem, where the available memory changes CPU allocation due to the nature of file systems that use cache mechanism (*i.e.*paging) to improve performance. This variable was also modeled and validated in the experiments. The proposed modeling was evaluated in two scenarios, first considering a specialized IO benchmark where it was possible to predict the CPU overhead of the CFS within an accuracy close to 98%. Later, an OLTP benchmark was used to produce a workload for a database persisted in a CFS. The modeling variables, for this case, were fitted using a multi-linear regression where it was possible to predict different scenarios setups with an accuracy close to 95%.

The modeling of the cloud storage sub-system is also part of the full-stack confidentiality architecture we are developing in this research. This is the second axis considered in the validation of the hypothesis *(ii) confidentiality costs can be modeled and used in the sizing of cloud computing environments*, which is supported by answering the research questions:

- *2 - What is the overhead for adding confidentiality mechanisms in the cloud computing stack?* It is possible to identify several components for supporting data confidentiality at rest, and we have shown how confidentiality is applied in this axis and what the impact when cryptography algorithms are applied. After being identified, the storage confidentiality was evaluated in terms of CPU allocation for persistence

operations in a cloud environment, and the overhead was explained by the relation between data volume and CPU allocation, mapped in the formulations.

- *3 - How can the overhead of combined security mechanisms for communicating, storing, and processing data in a cloud environment be estimated?* The modeling presented in this chapter uses as input the stored volumes handled by the application hosted in a cloud node. These volumes, read and written in a storage unit, feed the prediction formulas that also need to consider the cryptography algorithm used in the storage setup.

6. PROCESSING CONFIDENTIALITY MODEL

Although most of the cross-VM attacks are theoretical demonstrations, and some successful intrusions in co-resident VMs could be made only in specific processors [IIES14, RTSS09], there still are alerts from security institutes to adopt a cloud computing environment for sensitive data processing. The run-time phase of an application in a computational system consists of transferring data and instruction in/out of the processor. In conventional computer architectures, this is made in plain text, and the literature presents several works which aim to exploit information leakage in vulnerabilities in both the virtualization layer and hardware architecture [WL08, RJX08, HCAL17, RGVM13, WZL15].

Based on such security risks, which can also be exploited in cloud computing environments, the full-stack confidential architecture, presented in Chapter 3, introduced the processing axis where the confidentiality principle should be supported for data inside the processor chip, specifically when they are stored in registers, caches, and memories.

In order to protect data, even during the processing phase, some techniques, such as Homomorphic Encryption [Gen09, RSA78], Querying over Encrypted Databases [PZB11, AEKR14, DGBL+16, HILM02], and embedded processor's instructions [Int14], have been developed to keep data ciphered for calculations and comparisons, operations which are made on the memory-cache-CPU flow. In fact, data are never disclosed in this kind of solution, which increases the overall confidentiality of the system. However, this security approach adds overhead for the user's application, and additional costs should be mapped in order to estimate its impact on the cloud environment.

This chapter is structured as the following. Section 6.1 presents the technologies and solutions for supporting secure processing in cloud environments. Section 6.2 presents a high-level performance modeling for confidentiality in the processing axis. Later, an evaluation of the proposed modeling is described in Section 6.3, considering solutions in the literature and their performance evaluations. This chapter ends with the conclusions and considerations in Section 6.4.

6.1 Secure Processing in the Cloud

One of the main characteristics of cloud computing is the hardware-sharing capability supported by the virtualization layer. This layer supports the creation of Virtual Machines, each of them with virtual discs for storage, virtual network interfaces for communication, and virtual CPUs for data processing. Regarding processing, the concept of a virtual CPU (vCPU) is inspired by the so-called time-sharing mechanisms of operating

systems, where instead of a single process, the processor is shared by all guests' virtual machines (each VM with an OS applying internal scheduling in its turn) [BDF+03].

The run-time phase of an application demands transference of data and instructions between the processor and memories (caches and RAM), normally as plain text. Based on this characteristic, exploiting a virtualization or hardware vulnerability for copying or even retaining neighborhoods' VM's data may have success. Even if both communication and storage mechanisms add some security level, this fact should be considered in order to keep the data's security during their entire life cycle in such shared and public computational environment. It is actually not a threat if the physical host (memory and the processors) in a cloud environment is not shared, as in private clouds or in Virtual Private Clouds [Amac]. However, this scenario would increase the costs to deploy an application in the cloud, and it is also not common among cloud users due to financial feasibility.

Another approach that can be considered is to identify the sensitive parts of applications to schedule them properly. Watson P. *et.al.* [Wat12] present a multi-level security strategy to take apart sensitive applications' workflows and run them in a safe location. This safe location could be related to either a security-aware cloud environment, which would be penalized by cryptography processing, or a private cloud instance managed by the company's security rules. In such scenarios, it is possible to statically invest in the security of some specific resources, since the IT manager knows the sensitive applications' placement [FW12]. Although this kind of solution can solve the leakage of sensitive data, the application has to be modified, and its execution should be tracked for delivering applications' routines which process sensitive data in, i.e., either a private cloud or a dedicated bare-metal server.

Trying to solve security issues in public cloud environments, researchers have been producing some solutions to add confidentiality during the run-time phase. The embedded cryptography instructions of Intel processors [Int06, Int14] allow (de)ciphering data using hardware instructions. Those instructions handle only the registers' data, keeping data stored in both the cache hierarchy and the main memory encrypted. Although it could prevent data leakage for some cross-VM attacks by exploring the shared memory [WL08], data would be in plain text inside the chip, theoretically allowing some leakage. Adding such a technique also demands software modification due to the necessity of using specialized processor instructions. Recently, some works using container-based solutions and Intel SGX instructions have solved the side channel attack by binding dedicated processors and cache memories [KGP+17]. This approach leads to under provisioning and impacts the feasibility of cloud solutions, also demanding a specialized processor not common in public cloud providers.

In order to improve confidentiality for the processing phase, it is necessary to consider algorithms to handle encrypted data such as in operations over encrypted databases

and homomorphic encryption. This would allow the data stay encrypted during processing and therefore avoid plain data leakage.

The encrypted database querying [DGBL+16, AEKR14, BW07, HILM02, PZB11] techniques consider handling enciphered data as in Database Management Systems (DBMS). The technique handles queries by comparing enciphered parameters within the enciphered data stored in the database. For instance, a *Select Equals* operation, *e.g.* querying a name in a table, calculates a hash for the searching value and compares it within all hashes of a certain DBMS table's column. In its turn, the *Select Sum* operation, *e.g.* sum a table column, applies a homomorphic sum operation for a given numerical column in a search. In doing so, there is no exposition of sensitive data even during the querying phase. However, having been studied in last ten years, the encrypted database querying technique still is a incomplete database system with few mechanisms for complex SQL instructions, such as the store-procedures [AEKR14], causing companies to avoid migrating their applications using such encrypted DBMS.

At same time, research about homomorphic encryption has tried to develop enhanced mechanisms for confidential data processing. The homomorphic multiplication operation provided by the RSA cryptography [RSA78] supports multiplying two encrypted values resulting in an encrypted value which, when decrypted, is the product of the plain two values: for example, for the values a and b, the encryption function enc, and the decryption function dec, $a * b = dec(enc(a) * enc(b))$ [RAD78]. The only way to acquire the result is by deciphering the computed value with the private key, which is a pair of the public keys used for the two original values. This feature enables handling values without disclosing them during the processing phase, and it is called a homomorphic operation. In the work called Full Homomorphic Encryption (FHE) [Gen09] Gentry C. *et.al.* propose a complete model for computing encrypted values. This theory is considered a full solution for handling enciphered data in a trustful and secure manner providing additive and multiplicative operations [vDGHV10] that could support a complete set of functions for computational systems. But it is still a theory, and no practical solutions based on cryptography have achieved the desired results, even at a high overhead. In this context, a high-level cost modeling is presented in the next sessions, presenting an estimation of the impact added by a secure processing axis in the Full-Stack confidentiality architecture.

6.2 Cost Model

Unlike the linearity observed in communicating and storing models (Chapters 4 and 5), the complexity of each operation for the processing axis needs to be considered separately, which demands software tracing in order to estimate the overall impact of confidentiality in the processing phase. For instance, in CryptDB, a *Select equality* oper-

ation is far cheaper than a *Select sum* operation since the former computes simple hash comparisons while the second requires homomorphic operations.

The difference among operations demands a formulation which considers each operation in isolation. The dataset is also observed per operation. Regarding the operation's complexity, this work proposes an abstract formulation for modeling the extra computational resources in the processing axis. The cost of processing data with confidentiality for this scenario is a function with operations and dataset as input. This function should produce a CPU overhead as the output. This can be written as:

$$f(op_1, op_2, ..., op_n, d) = C_1(op_1, d) + C_2(op_2, d) + ... + C_n(op_n, d) \quad (6.1)$$

where op is related to the operation weight, d is a given dataset, and C is a function producing a cost in terms of CPU overhead to compute the operation over the dataset. The final result f is given by the total cost of operations. In other words, a dataset associated with an operation has its own weight in the total overhead estimation. This function could be replaced by a regular mathematical calculation to achieve such results. From this high level modeling, one can apply a formulation based on a multi-linear regression[1], where each operation has its own weight. This can be written as:

$$f(op1, op2, ..., opn) = \beta_0 + op1\beta_1 + op2\beta_2 + ... + opn\beta_n \quad (6.2)$$

where β values are the weights to be fitted with a training dataset, which will produce the result based on data volumes of each operation.

6.3 Applying the Cost Model

This model is applied in two phases: mapping the application's operations, and applying per-operation overhead, which has been already measured by cryptography database authors, in this case, the CryptDB.

The workload used in this evaluation consists of an OLTP benchmark with a set of operations, common in an e-commerce application. The TPC-C benchmark [LD93], used in this thesis in previous evaluations, produces a workload based on the following operations:

- New Order: consists of creating an entire new order with a high frequency and heavy flow, which represents on-line operation for common production environments;
- Payment: updates customers' transactions, reflected in several databases' tables, and represents a light-weight operation with a low-frequency flow;

[1] http://www.stat.yale.edu/Courses/1997-98/101/linmult.htm

- Order Status: queries the status of customers' last order. It is a mid-weight low-frequency operation flow;
- Delivery: process a batch of 10 new (not delivered) orders, reviewing the entire operation, deleting and updating several tables;
- Stock Level: query recent sold items, looking for products with low stock level. It is a light-weight operation and not frequent.

Equation 6.1 will be used to quantify the overhead of the different complexities for encrypted operations. The application of this formula requires profiling the software taking into account the database queries. For these experiments, the set of operations executed by the TPC-C benchmark were mapped, and the overhead for each was calculated according to the CryptDB evaluation [PZB11, PRZB11]. Table 6.1 presents the accumulated overhead for each TPC-C operation following the specification of the benchmark[2]. It is possible to observe different operation sets being affected by the cryptography characteristics added by CryptDB. Particularly for the *Delivery* operation, although it is processed in batches, it has the highest overhead specially because the addition of the operation *Select Sum*, which demands a homomorphic sum. Such an operation was reported in the CryptDB author's paper as the most expensive operation, with an overhead up to 98%. The *Delivery* operation is composed by a set of *Select Equals*, *Update*, *Delete* operations, and one *SelectSum* operation, which impacted the overall overhead even though it was called once.

Benchmark Phase	New Order	Payment	Order Status	Delivery	Stock Level
Operations	Select Equals Update Insert	Update Inc Update Select Equals Select Range	Insert Select Equals Select Range	Select Equals Delete Update Select Sum Update Inc	Select Equals
Overall Overhead	8,73%	25,17%	10,04%	27,39%	7,6%

Table 6.1 – Calculated overhead based on SQL operations demanded by each TPC-C phase [PZB11].

Besides the overhead added by the techniques of Querying over Encrypted Databases, CryptDB also changes the data volume persisted in the server side, adding extra columns with metadata to tables. According to the authors, this increases the database volume by 4.5 times in terms of storage allocation. Figure 6.1 depicts the comparison between a base-line execution using a non-encrypted database, represented by the solid

[2] https://github.com/Percona-Lab/tpcc-mysql

line, and a encrypted database, represented by the dotted line. The values for the encrypted database were obtained though mapping the benchmark's operations and Table 6.1, producing an estimated overhead.

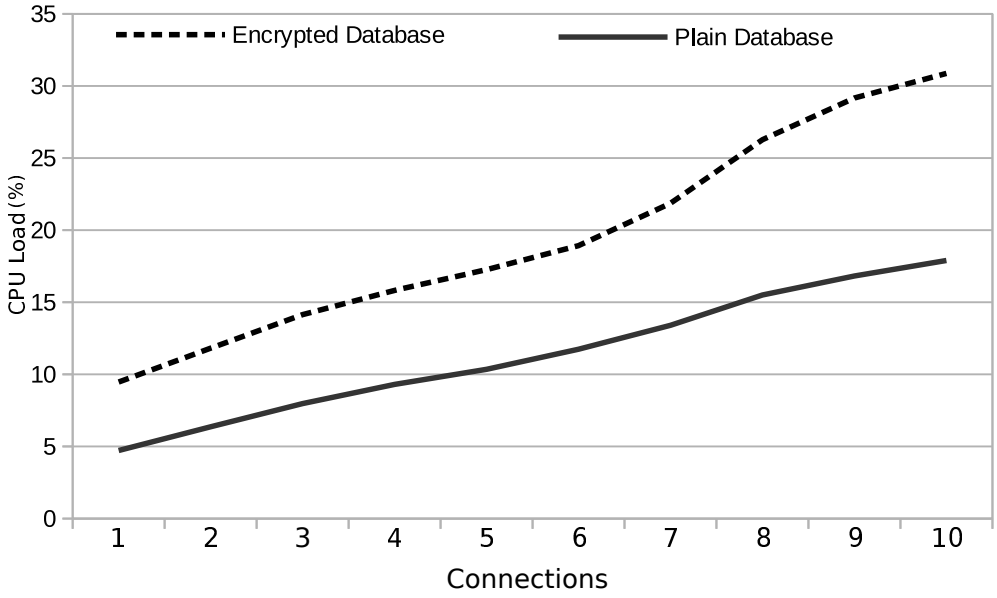

Figure 6.1 – CPU Load overhead recalculated in function of CryptDB overhead.

6.4 Summary

The solutions to support security in the cloud environments, in particular for the confidentiality principle, can be supported in the processing axis through mechanisms such as the Homomorphic Encryption and the Querying over Encrypted Databases. Although the virtualization technology, used for sharing server's processor among users, supports context isolation for users' VMs, the literature presents several works in which information leakage is exploited, *e.g.*, cross-VM attacks.

The solutions for supporting confidentiality of users' applications during their run-time phase are based on two primary requirements: data should be ciphered, and every processing over the data should not disclose them. These conditions demand a per-operation solution since the complexity of data manipulation is dependent on the application features development. A general solution, yet a theoretical concept, is proposed by Gentry as a Full-Homomorphic Encryption [Gen09]. Alternatively, the Querying over Encrypted Databases solutions proposes a set of operations to achieve security requirements, including the confidentiality principle. All solutions describe overhead in supporting encrypted data processing.

In this chapter, a high-level modeling is proposed with the aim of estimating the overhead of supporting homomorphic operations in cloud computing environments. Based

on overhead measurements of an Encrypted Database, we produced an evaluation where it was possible to identify the database operations of an OLTP benchmark and map them within an Encrypted Database solution.

This chapter complements the full-stack confidentiality model we developed in this research. This is the third axis supporting the validation of the hypothesis *(ii) confidentiality costs can be modeled and used in the sizing of cloud computing environments*, and complements the research questions as follows:

- *2 - What is the overhead for adding confidentiality mechanisms in the cloud computing stack?* New solutions for processing over encrypted data have been developed, and have opened new opportunities to move confidential data processing to cloud environment, which allow users to get its benefits. Most of the solutions for handling encrypted data add a high overhead to the computational systems, but some solutions for Encrypted Databases have low-cost operation such as the hash comparisons. Application's operation have different overheads, from 7% when comparing hash values, 20% when encrypting new data, and 98% when processing an homomorphic operation. Hardware-based confidentiality is an alternative with low performance impact, but is only possible with modifications in the user application.

- *3 - How can the overhead of combined security mechanisms for communicating, storing, and processing data in a cloud environment be estimated?* The difference in the CPU load of confidential solutions' operations demands an application trace to analyze their cost individually. Each application operation should be mapped as either a Homomorphic Encryption routine or a specialized query of the Encrypted Databases or a security hardware instruction. These confidential operations, when applied to a given dataset, will produce an overhead to be used in the formula fitting, which then finally can be used to estimate the overhead of supporting confidentiality for data during the processing phase. It is not a straightforward application since the solutions for this axis are still not mature, and many improvements need to be made if such a solution starts to be part of cloud computing environments. It is also necessary to consider the operations impact in the other axes such as in network and storage and then apply the formulas for predicting the total CPU load of a full-stack confidential architecture appropriately. This evaluation is presented in the next chapter of this work.

7. COST MODEL USE CASES

The cost model proposed in this work can be applied to estimate the impact of supporting the confidentiality in cloud environments. On one hand, for the cloud models SaaS and PaaS, cloud providers are responsible for security support, and should implement mechanisms for levering confidentiality, integrity, and availability of rented assets. In such a scenario, the provider needs to estimate the costs of supporting security principles for the services delivered to the customers so that this service is properly priced. For instance, a Database-as-a-Service product could achieve a higher security level by implementing confidentiality for both the communication with the customers application and the data of files stored in the public cloud, however, this increased security level may impact resources utilization and consequently the price of the services.

On the other hand, the IaaS cloud model lays out the security concerns to be managed by cloud users. Users also need to estimate the extra costs of adding privacy principles such as deploying a Cryptography File System (CFS) for protecting a VM's files or even implementing a VPN among public cloud nodes. This deployment also demands the knowledge of the impact of adding security mechanisms since the public cloud resources are billed on-demand, and any extra resource allocation will impact the final price of the services [MKL09]. Figure 7.1 depicts the security responsibility distribution between users and providers.

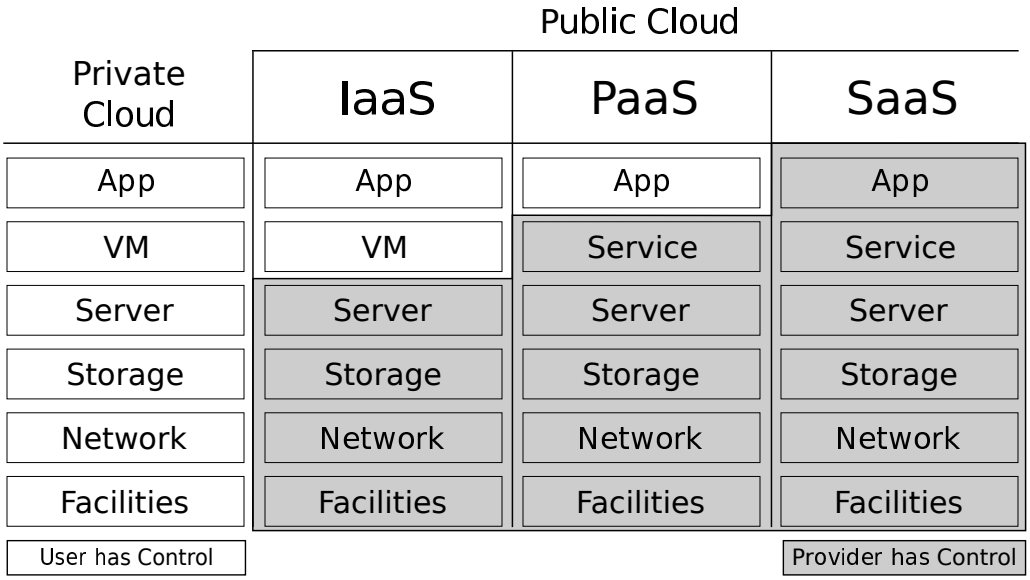

Figure 7.1 – Security responsibility in cloud computing deployment models.

With the aim of demonstrating the utilization of the cost model proposed in this work and discussing the overheads when adding security in cloud environments, the following scenario is described and the security costs are sized considering its adoption of confidentiality. The scenario consists of a Provider Cost Evaluation which considers demon-

strating the responsibility of cloud providers in delivering a safe solution to clients, considering cloud modes PaaS and SaaS.

7.1 Scenario Description and Measurements

Cloud computing providers have gained attention in recent years due to delivering a set of computational services demanded by companies of any size, from small start-ups to big and well-established corporations. The providers deliver services in three different modes. The IaaS mode provides computational infrastructure as services which include Virtual Machines, storage disks, and network. The PaaS mode delivers to customers essential services used by developers such as databases, message queues, and mailing services. Finally, the mode called SaaS delivers to end-users complete software solutions such as office suites, customer relationship management (CRM), and social media tools.

In order to understand the feasibility of delivering secure service in these three modes, it is necessary to first understand and then predict the overhead of the mechanisms that support the security principles.

In this context, a scenario of a database instance delivered to the customer as a service is considered (classified as PaaS). In practice, the provider deploys virtual machines with a desired software stack (*i.e.* a MySQL instance) and applies the concepts of elasticity (vertical or horizontal) to support the demanded user's workload. This deployment can be offered to users under different security levels, by adding confidentiality in both communication and storage axes.

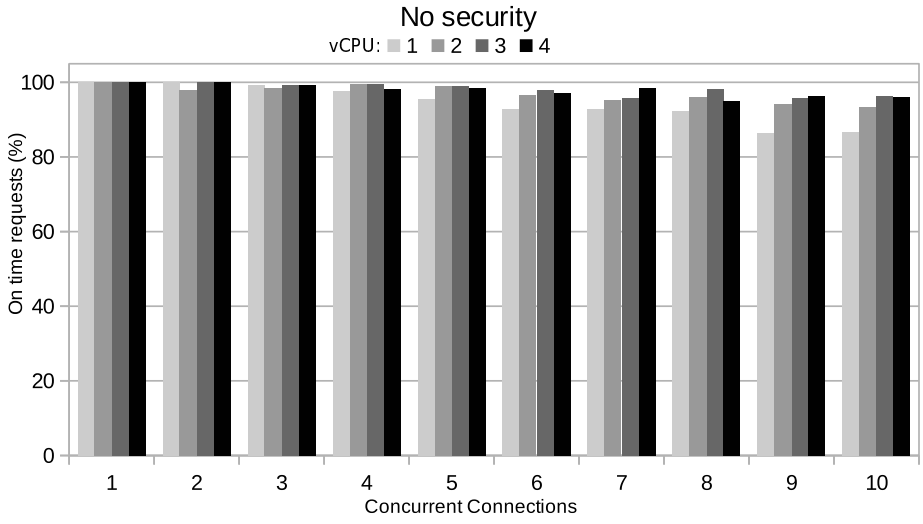

Figure 7.2 – Service level of requests, for a non-safe cloud environment.

To understand the impact of confidentiality mechanisms in a cloud environment, a set of experiments based on an e-commerce workload were conducted, and the response time of requests was observed. In the following figures, the horizontal axis represents an increasing demand with concurrent clients (from 1 to 10) making requests into the database system. The vertical axis is related to the amount of request which is on-time (considering the minimal response time for each operation). The color in bars, represents the number of vCPUs allocated for the VM hosting the DBMS. Figure 7.2 presents measurements of an environment without the adoption of any security mechanism. In all executions, the amount of requests achieving the desired response time was superior to 90%.

The next evaluation includes security in the communication between the client (demanding requests into the server), and the server with VPN (Figure 7.3).

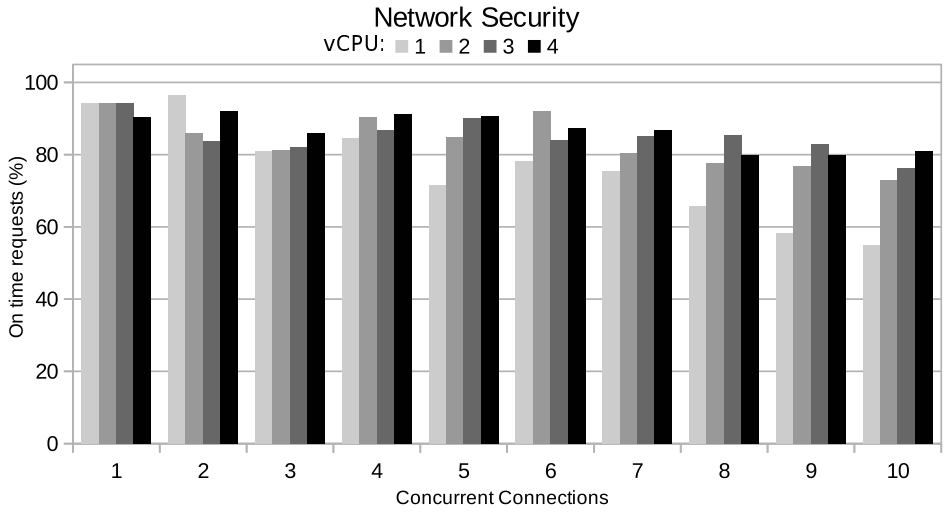

Figure 7.3 – Service level of requests, for a VPN-based cloud environment.

It is possible to observe, a reduction in requests that can achieve the desired response time as more concurrent clients are added. The overhead of the VPN is easily noticeable by its impact on the services. When more than eight concurrent users were added, it was not possible to achieve 90% of succeeding requests. Subsequently, in order to understand the storage security impact, the DBMS was deployed over a CFS, to support confidentiality for persisted data.

Figure 7.4 presents the impact of a CFS, which does not add significant overhead compared to the VPN scenario. This behavior reflects the fact that the VM's memory is large enough for the DBMS to make only in-memory operations generating few file system page fault. Only scenarios with one vCPU could not achieve the desired response time for more than eight concurrent clients. Finally, an environment considering a VPN and also a CFS is evaluated.

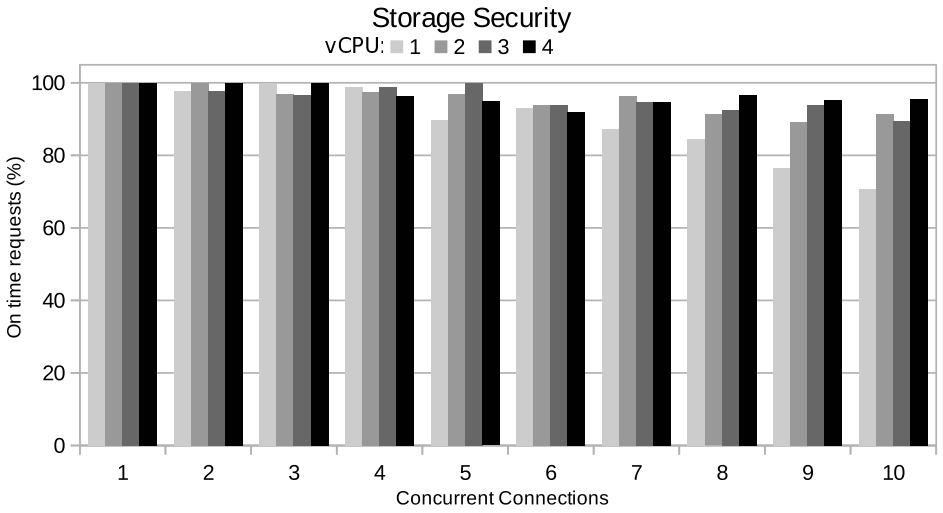

Figure 7.4 – Service level of requests, for CFS-based cloud environment.

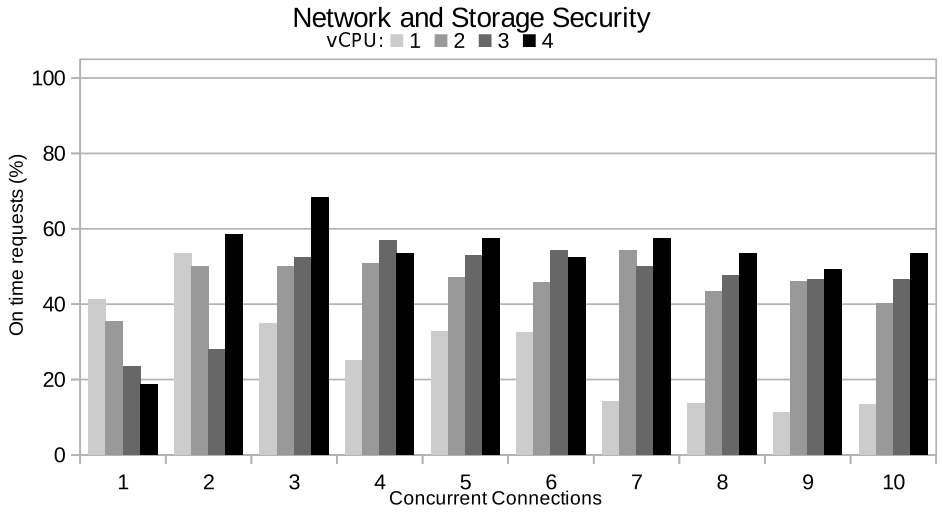

Figure 7.5 – Service level of requests, for VPN-CFS-based cloud environment.

Figure 7.5 presents the impact of combining two security levels, in this case for communicating and storing data. In all scenarios, even for a single connection, no more than 50% of the requests are within the desired response time. The isolated test scenarios contribute with CPU allocation and data volumes for tuning the variables of the cost model formulas, which are demonstrated in next section.

7.2 Applying the Confidentiality Costs Model

In the test-cases presented in Section 7.1, the evaluation shows that cloud providers have to resize resources if they want to achieve SLAs when security is needed. This can be also verified with the model developed and validated in Chapters 4 and 5. It

considers the data volume communicated and persisted in the environment as the input of the formulas.

The test case using confidentiality for communication, presented in Figure 7.3, produced the values in terms of data volume and CPU load used to fit the network modeling formula. By rewriting the network modeling equation 4.4:

$$N_{(s,r,b)} = \frac{E \times s + D \times r}{(s+r)/b} \tag{7.1}$$

the variables to be fitted are E and D, since the others are part of the function input. For this case the multi-linear regression yielded $E = 0.00000006385$ and $D = 0.000002805$, and the standard error of the values was expressed in the R-squared 0.9991.

Similarly, the test case applying confidentiality for the storage, using a CFS, produced the values for fitting the persisted data volume and the CPU load used by the CFS in the storage modeling. By rewriting the storage equation 5.2:

$$S_{(w,r,TP_r,TP_w)} = \frac{E \times w + D \times r}{(r/TP_r) + (w/TP_w)} \tag{7.2}$$

the variables to be fitted are E and D, since the others are part of the function input. For this case the multi-linear regression yielded $E = 0.0000000222$ and $D = -0.00000000549$, and the standard error of the values was expressed in the R-squared 0.991.

Table 7.1 presents the volumes and the CPU overhead of each confidentiality scenario where the variables' weights were obtained from. It was possible to notice an increase in the data volume as more concurrent connections were added. The data volumes also produced a linear relation within the CPU overhead in both scenarios. However, even when adding security to network and storage, data throughput still increased as more clients were added in parallel.

Clients	Network with VPN			Storage with CFS		
	sent[1]	received[1]	CPU Overhead	read[1]	write[1]	CPU Overhead
1	9380	1797	6.88%	16	71386	2.08%
2	11442	2183	7.85%	10384	60244	1.75%
3	15777	3010	10.87%	3040	79802	2.24%
4	20098	3853	13.72%	2864	100118	2.68%
5	24940	4764	16.63%	5680	142990	3.57%
6	29741	5687	19.65%	944	161318	4.00%
7	32959	6306	21.48%	5616	177396	4.34%
8	35154	6710	22.70%	1184	191127	4.75%
9	35036	6715	22.45%	7840	222755	5.46%
10	34770	7022	23.54%	3344	225332	5.63%

Table 7.1 – Measured values for data volume and CPU overhead, used for fitting prediction formulas. [1]Data volumes are expressed in KBytes.

When applying the weights for encrypting and decrypting data for both communications and storage together, it is then possible to predict a scenario where confidentiality is applied simultaneously for the network and storage axes. Table 7.2 presents the measured values for network and storage when using cryptography tools for confidentiality support. This table also presents the *Measured CPU Overhead* as well as the comparison within the *Predicted CPU Overhead*.

Now, it is possible to apply the modeling for the processing axis to estimate the total overhead of confidentiality when applied in all three layers.

From the modeling for confidentiality in the processing axis, presented in Chapter 6, one can consider applying the overhead for the database operations of the *e-commerce* benchmark presented in Section 6.3. The operations mapped in Table 6.1 are the same in the evaluation presented in this section. From that profiling, it is possible to map the transaction allocations and the increase in storage (which is 4.5 times bigger), based in columns *read* and *write* of Table 7.2.

Column *Full-Stack overhead for CryptDB*, in Table 7.2, presents the impact of storing this experiment in an Encrypted Database, in this case the CryptDB. The prediction for this column needs to consider the overall overhead added by the cryptographic operations as well as the increasing in the storage data volumes, applied by CryptDB. Considering confidentiality in communication and storage, from the current experiment, and the overhead of the confidentiality in the processing axis, it was possible to estimate the impact of the Full-Stack Confidentiality, which was 10% in average (learned from previous experiment in Chapter 6). This values are an estimation based on CryptDB paper [PZB11] since they could not be verified practically due to TPC-C demands operations not supported by available version of CryptDB.

Clients	Network with VPN and Storage with CFS						Full-Stack overhead for CryptDB
	sent[1]	received[1]	read[1]	write[1]	CPU Overhead Measured	CPU Overhead Predicted	
1	2958	564	3456	60482	3.51%	4.78%	9.13%
2	5328	1018	96	67913	5.35%	6.36%	11.36%
3	7583	1455	1744	77344	7.01%	7.97%	13.69%
4	9590	1836	3488	82308	8.68%	9.30%	15.39%
5	11117	2130	528	86064	9.60%	10.35%	16.74%
6	13240	2533	8688	92357	11.28%	11.74%	18.57%
7	15393	2941	3856	106361	12.83%	13.40%	21.34%
8	17759	3395	2656	134359	14.89%	15.50%	25.63%
9	19188	3668	4496	154359	16.59%	16.82%	28.50%
10	20652	3950	2000	161244	17.56%	17.90%	30.12%

Table 7.2 – Model application considering the trained formulas for a scenario with confidentiality guarantees in communication and storage, and a prospection of Full-Stack confidentiality costs using CryptDB. [1]Data volumes are expressed in KBytes.

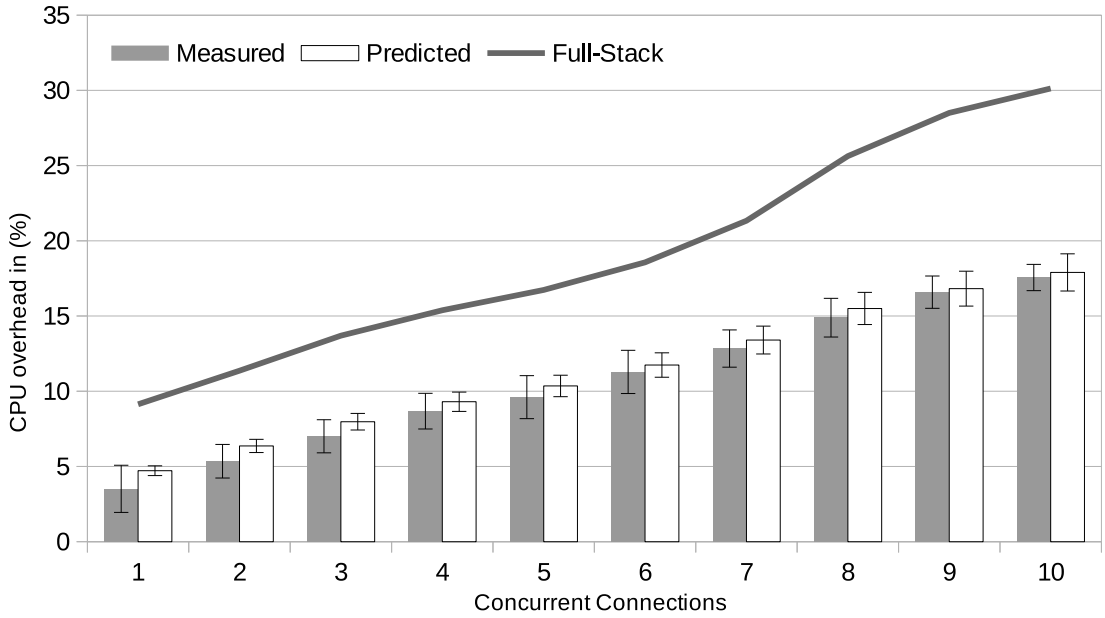

Figure 7.6 – Environment overhead comparison in terms of CPU allocation. Bars represent a scenario only with network and storage security. The line represent the estimated CPU overhead including the Processing axis.

Finally, for this evaluation, the CPU overhead was observed and compared with the prediction produced by applying the model, presented in Chapters 4, 5, and 6, in this work. From Figure 7.6 we can conclude that the larger the volume is in both network and storage, the higher the CPU overhead is, compared with a standard environment. This behavior validates the characteristics developed in Network and Storage modelings, which associate data volumes and cryptography overhead. The figure also depicts the comparison bars between the measured and the predicted CPU overhead, which achieves an accuracy close to 95% in average. The solid line in this figure presents the estimated overhead for a Full-Stack Confidentiality scenario, applying the cryptographic techniques and tools, presented and discussed in this research, for the communication (using VPN), storage (using CFS), and processing (using CryptDB) axes.

7.3 Applying the Model in Public Cloud Providers

Several Cloud Computing providers have been developing software solutions for companies of many sizes. The solutions range from static web hosting to robust BigData processing and Artificial Intelligence applications. Regarding the security, the in-transit solutions are already a commodity through the well-established HTTPS protocol [NFL+14]. Moreover, when a privacy requirement demands that data should be kept encrypted at rest, cloud computing providers have started to offer solutions for encrypting the data itself, as in object storage solutions [Amad], and for encrypting the entire VM's disk. The

solutions allow a user to upload the key used for the cryptography process, but those keys should be managed by the cloud provider, which can be a security issue as discussed later.

To verify the applicability of the modeling proposed in this thesis, we measure the overhead of adding security using the on-board features of current cloud providers. The Amazon AWS [Amaa] provider was chose for being a pioneer in cloud computing solutions and the biggest player in the current scenario. From the available products supporting encryption, the Elastic Block Store (EBS) [Amab] is the AWS product supporting the creation of the VM's disks used for persisting data, as well as for the VM's operating system. This product supports disk encryption as a feature offered by the cloud provider. In other words, the user rents an encrypted disk with the guarantee that data are ciphered at rest.

In this scenario, we conducted experiments with two VMs (a dedicated VM without sharing CPU cycles - in AWS they are identified as EC2-M4), one of them hosting a database and the other one hosting the client benchmark software. The virtual machines were placed in the same availability zone[1] and the same local network. From this environment, the *e-commerce* benchmark execution, used in previous evaluations, was reproduced for scenarios considering the encryption of the disk where the database is persisted. The execution considered disks with encryption supported by the cloud provider in two disk types: standard magnetic disks and disks with provisioned IO operations. According to AWS documentation [Amab], the disk encryption process encrypts data inside the bare metal host, so both data transmission to the storage unit and its persistence are ciphered.

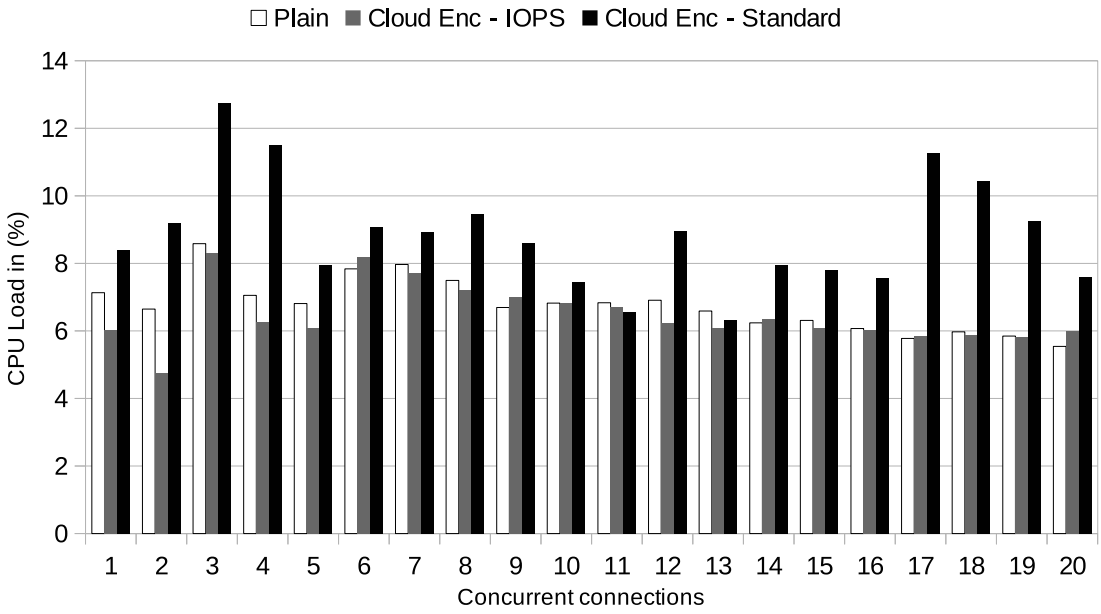

Figure 7.7 – Disk encryption comparison between Cloud encryption and user encryption.

Figure 7.7 shows the CPU load comparison of the database node for the three disks' scenarios. It is possible to observe that there is no significant CPU load in the

[1]An availability zone is a physical data center where VMs are physically allocated.

Virtual Machine, which is characterized when the Cloud Provider does not charge users' allocation with the Disk Encryption process since it is a feature of the cloud's storage service. From this observation, it would be necessary to measure the overhead in the bare-metal host, which is not allowed from the users' perspective. The experiment also allows the creation of Virtual Private Networks, deployed by users. These VPNs added similar overhead compared to the first scenario presented in this chapter.

Although the impact of confidentiality mechanisms could not be observed in the users' application, the cloud provider is impacted by the cryptography operations in the persistence flow. This impact could change price aspects of providers' service, which would motivate them to apply a similar evaluation on the lower layer to understand and predict the overall impact of offering Disk Encryption in their services.

7.4 Summary

In order to build a confidential cloud computing environment and to measure the CPU impact of cryptographic mechanisms in the cloud software stack, this chapter started with an experiment in a controlled environment where the cost model could be evaluated to predict an environment composed of secure communication and storage. The potential impact on the service availability is demonstrated in Figure 7.5 where, on average, 50% of the requests of the e-commerce benchmark could not be served on time. After applying the modeling of network and storage, developed in Chapters 4 and 5, it was possible to predict this overhead with an accuracy close to 95%. From this evaluation, an estimation for a full-stack confidentiality is then calculated from the observations of the processing axis, presented in Chapter 6. Although its validation could not be reproduced in practice, due to benchmark and CryptDB compatibility, we demonstrated the empirical estimation of the Full-Stack Confidentiality Architecture, based on CryptDB overhead measurements.

This chapter also shows the evaluation of a real cloud scenario, considering a public cloud provider. The provider supported the creation of Encrypted Disks, and it was possible to understand that no CPU overhead is transferred to users' VMs during the encryption of data persistence flow. However, it would be interesting to predict such overhead on the provider's side in order to understand the costs of supporting confidentiality.

This chapter answered the research question *4 - How can a security overhead modeling help users and providers to create a confidential cloud environment?* Once it is possible to understand the data volume and the cryptography type (algorithms and key sizes), users and providers can predict the overhead of adding confidentiality in the communication, storage, and processing axes and then recalculate the costs of the cloud computing environment. This may leverage some future work in scheduling algorithms

and new security strategies for balancing the confidentiality level and the costs in renting cloud computing environments.

8. CONCLUSION

8.1 Concluding Remarks

In recent years, companies and government agencies have increased their security demands to ensure strict privacy features in cloud computing environments. Even though cloud security has been discussed both in industry and academia, cloud services rarely support data leakage avoidance, especially for the processing phase of sensitive data. This gap prevents most users, especially the ones who have privacy demands, from adopting cloud services, and, consequently, they will not earn the benefits from reduced costs, infinite scalability, and the many other advantages of cloud computing.

With the intent of investigating and proposing solutions, many works [KKA14, MJ15, RK14, RC15, KKKM13] have identified and mapped the vulnerabilities of computational systems on cloud environments. The alternatives for offering confidentiality in a full-stack manner, considering all layers of cloud environments, points to cryptography as the technology that supports confidentiality in communication (using VPNs), storage (using CFSs), and also in the processing phase (through Homomorphic Encryption). Such a scenario is proposed in this thesis as the Full-Stack Confidentiality Architecture for Cloud Computing, where its design is presented, considering the placement of cryptography components for preventing confidentiality risks, such as information leakage, even during data processing, considering the shared tenancies feature of the cloud.

Although the proposed architecture increases privacy levels, the security mechanisms that support the confidentiality principle are commonly neither part of software sizing estimations nor the cloud's pricing. This may lead to under-provisioning and consequently an additional investment to maintain the service security. Based on this context, this Ph.D. work also conducted the research for identifying and modeling the overheads of current solutions in security to build a cloud computing environment with confidentiality for data in-transit, at-rest, and on-processing.

It was possible to ensure that for the communication among services in cloud environments there are standard models following Internet-based protocols such as HTTPS. This protocol demands the creation of a secure tunnel between peers by exchanging cryptographic keys, as in Virtual Private Networks. The research in Chapter 4 also shows that cryptographic algorithms add CPU overhead to the communication, and the modeling proposed a linear relationship between the data volume and the CPU overhead. This modeling is mapped considering several algorithms such as AES, Blowfish, and Camellia. The evaluation demonstrates the accuracy of the cost model which was between 91% and 98%. This model was the starting point in this research, and helped to answer part of the re-

search questions related to the feasibility of supporting confidentiality for communication and the estimation of their costs.

The research work then investigates the security of the storage in cloud computing environments. Similarly to communication, the confidentiality principle applied in storing data in standard cloud computing environments also uses symmetric cryptography. Data is encrypted and decrypted during the persistence flow, which ensures that no plain text is kept at rest in third party devices. This characteristic is important due to security reasons since the cloud computing nature considers sharing computational resources in order to reduce client costs. For this axis, the modeling presented in Chapter 5 also considers the relation of data volume and the CPU overhead added by cryptography tools. In the evaluation, an environment using a Cryptography File Systems could demonstrate the overhead added by security, and the modeling could estimate this overhead with an accuracy close to 98%, using the AES algorithm for ciphering persisted data. This research complements the answers for the research questions regarding availability and confidentiality costs.

The confidentiality principle, already established for communication and storage, is also applied for the processing phase of data in cloud computing environments. As for the shared environment in the cloud, the virtualization technology leverages its success by also sharing the CPU time, allowing it to place users' VMs into the same bare-metal servers through the virtualization consolidation techniques. Despite their advantages, they also opened issues related to security such as the ones exploited by the cross-VM attack, leveraging information leakage in co-resident VMs. Chapter 6 presents solutions based on cryptography, not yet mature for production, to support the processing of encrypted data without disclosing it such as the Full Homomorphic Encryption [Gen09], which is a possibility for making additive and multiplicative operations; however, it was not possible to identify an algorithm with these characteristics. Nevertheless, the techniques for Searching over Encrypted Database are further developed, promoting full confidentiality when handling data, but impacting the overall system in terms of both CPU utilization and data volume. Based on the literature, this chapter presents a high-level modeling which considers the diverse complexity of database-encrypted operations. In the model demonstration, a database benchmark is analized in order to identify all querying operations which demand to be secure. Then the overhead of each operation is calculated, based on the evaluation presented by the authors of this solutions. This demonstration was conducted to show the application of the confidential processing modeling and to complement the answers for the research questions, based on the Full-Stack Confidentiality Cloud environment.

The three modelings, that may be combined, aim to support the cloud manager's decisions when adding the confidentiality principle in the software stack. Chapter 7 presents a scenario where a cloud provider delivers a Database-as-a-Service which

is classified as a PaaS service. When the confidentiality principle is applied the communication and storage axes, measurements show a reduction in the service quality, specifically affecting the response time of users' requests. Based on the observation of the data volumes in communication and in the persistence flow, the modeling developed in this research was applied to estimate the overall impact in terms of performance and, in consequence, the quality of the services. CPU overhead was predicted with an accuracy close to 95%.

The presented experiments demonstrate that the relation among the variables identified in this work is a mathematical representation for understanding the adoption of cryptography in computational environments, in this case, cloud computing environments.

8.2 Hypotheses Validation

The development of this research thesis validated the hypotheses that it is viable to implemen the confidentiality principle in the entire data's life-cycle and also model the confidentiality impact in cloud computing environments. The validations were conducted considering the state of the art in the researched area, as well as performing experiments with cryptography algorithms in cloud computing environments, for three axes: communication, storage and processing. Four research questions were answered to support the hypotheses:

1. *What are the confidentiality risks and solutions in cloud computing environments, and how is a confidential cloud created?* There are several risks and threats related to confidentiality in cloud environments, especially in public providers where resources are shared in multi-tenancies with different users. The taxonomy presented in Chapter 2 points out the confidentiality principle as an important player in coping with the challenges of cloud computing adoption by users with strict privacy levels. By adding solutions such as cryptography in the cloud software stack, users can relish the benefits of cloud computing with higher guarantees of avoiding information leakage during the communication, storage, and even the processing phase. The confidentiality's components placement is presented in Chapter 3 as a Full-Stack Confidentiality Architecture, where cloud services are deployed. These services can support the confidentiality principle through handling users' data using cryptographic tools and protocols during the three main phases of data's life cycle: communication, persistence, and processing.

2. *What is the overhead for adding confidentiality mechanisms in the cloud computing stack?* Based on the Full-Stack architecture, it was possible to identify the components impacting cloud's overall overhead for supporting confidentiality. The costs

of applying confidentiality in communication, storage, and processing are discussed and modeled in Chapters 4, 5, and 6, respectively. Regarding communication and storage, the overhead of using cryptography has a linear relation within the data volume, both communicated and persisted. From a different perspective, the new solutions for processing over encrypted data produced a modeling where the complexity of each operation needs to be evaluated against a particular dataset. Most of the solutions for handling encrypted data add a high overhead to the computational systems, as shown in Chapter 6, but some solutions for Encrypted Databases have low-cost operations, such as hash comparisons, and could be feasible for some applications.

3. *How can the overhead of combined security mechanisms for communicating, storing, and processing data in a cloud environment be estimated?* From the evaluation motivated by the previous research question, Chapters 4, 5, and 6 presented modelings which consider the variables of data volume, cryptography algorithms, key length, and processing characteristics to predict the overhead added by the confidentiality principle in cloud computing environments. The operations, when applied individually to a training dataset, produce an overhead to be used in the formulation fitting, which then finally can be used to estimate the overhead of supporting data confidentiality during their life.

4. *How can a security overhead modeling help users and providers to create a confidential cloud environment?* Once it is possible to understand the data volume and the cryptography type (algorithms and key sizes), users and providers can predict the overhead of adding confidentiality by following the architecture proposed in Chapter 3. The validation developed in Chapter 7 considered an e-commerce benchmark producing a workload in an environment with confidentiality guarantees in the communication and storage axes. After fitting the individual formulas, a prediction was demonstrated, and it was possible to achieve an accuracy close to 95% in terms of extra CPU load in the cloud nodes. From this evaluation, and based in the processing axis modeling presented in Chapter 6, we also estimated the overhead of the full-stack confidentiality architecture.

8.3 Future Work

Security is still an increasing demand for computational systems due to many reasons, from *ransonware* attacks to new financial and business models using, for instance, cryptocurrencies. From the results obtained in this research, it is possible to suggest topics for future research considering the scope of *confidentiality and cloud computing*.

There are several security mechanisms that have been added to cloud computing services, including secure object storage, disc encryption, and authentication with cryptography key management. These mechanisms can compose an extension to the modeling presented in this work, considering not only the confidentiality principle but also integrity, availability, authenticity and trustworthiness, etc. It will be necessary to understand the software implementation which supports those principles as well as to map them within the cloud computing environment.

As a relevant topic, it would be productive for researchers and also the industry creating cloud security components to contribute to current cloud projects such as OpenStack[1] and Cloud Stack[2]. Once they become part of the solution, users would start to consider confidentiality in their deployments. Regarding the addition of new cryptography components, an important topic to be researched is the impact in terms of management costs and time to deploy security components to support not only confidentiality but integrity and availability principles.

Nonetheless, the permanent evolution in building new solutions for cloud computing also demands a review of security principles. The rising of the *container-based* clouds reopens some issues in sharing computational resources since the isolation feature of the hardware virtualization is not part of the container's software stack. With the aim of improving systems performance, the *containers*, which replace the Virtual Machines, cut off the virtualization layer by sharing a kernel among their users, instead of sharing hardware (CPU, memory, disks, and network). Although they offer higher risks for the isolation of user's assets, they also offer higher performance, making the adoption of stronger cryptography algorithms feasible. We understand that the more the performance is improved by computational systems, the more feasible it will be to add security mechanisms.

[1] http://openstack.org
[2] http://cloudstack.apache.org

REFERENCES

[ABF+16] Alves, D. C.; Batista, B. G.; Filho, D. M. L.; Peixoto, M. L.; Reiff-Marganiec, S.; Kuehne, B. T. “CM Cloud Simulator: A Cost Model Simulator Module for Cloudsim”. In: Proceedings of the IEEE World Congress on Services (SERVICES), 2016, pp. 99–102.

[ACKM04] Alonso, G.; Casati, F.; Kuno, H.; Machiraju, V. “Web services”. Springer, 2004, 354p.

[AEKR14] Arasu, A.; Eguro, K.; Kaushik, R.; Ramamurthy, R. “Querying encrypted data”. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, 2014, pp. 1259–1261.

[AGP+] Abdalla, M.; Gierlichs, B.; Paterson, K. G.; Rijmen, V.; Sadeghi, A.-R.; Smart, N. P.; Stam, M.; Ward, M.; Warinschi, B.; Watson, G. “Algorithms, key size and protocols report”. Accessed: Dec, 2016, Source: http://www.ecrypt.eu.org/csa/documents/D5.2-AlgKeySizeProt-1.0.pdf.

[Amaa] Amazon Web Services, Inc. “Amazon AWS”. Accessed Nov, 2013, Source: http://aws.amazon.com/.

[Amab] Amazon Web Services, Inc. “Amazon Elastic Block Store (EBS) – Block Storage for EC2”. Accessed Jan, 2017, Source: https://aws.amazon.com/ebs/.

[Amac] Amazon Web Services, Inc. “Amazon Virtual Private Cloud (VPC)”. Accessed Jan, 2017, Source: https://aws.amazon.com/vpc/.

[Amad] Amazon Web Services, Inc. “Protecting Data Using Encryption”. Accessed Jan 2017, Source: http://docs.aws.amazon.com/AmazonS3/latest/dev/UsingEncryption.html.

[AMN09] Al-Mandhari, W.; Nakajima, N. “Throughput enhancement with channel interference cancellation in multi-hop/multi-radio wireless mesh network”. In: Proceedings of the 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology, 2009, pp. 248–251.

[BBE+13] Bari, M.; Boutaba, R.; Esteves, R.; Granville, L.; Podlesny, M.; Rabbani, M.; Zhang, Q.; Zhani, M. “Data center network virtualization: A survey”, *Communications Surveys Tutorials, IEEE*, vol. 15–2, 2013, pp. 909–928.

[BDF+03] Barham, P.; Dragovic, B.; Fraser, K.; Hand, S.; Harris, T.; Ho, A.; Neugebauer, R.; Pratt, I.; Warfield, A. “Xen and the art of virtualization”, *SIGOPS Oper. Syst. Rev.*, vol. 37–5, Oct 2003, pp. 164–177.

[BHW06] Bowen, P.; Hash, J.; Wilson, M. “Sp 800-100. information security handbook: A guide for managers”, Technical Report, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2006.

[BKL+09] Blaze, M.; Kannan, S.; Lee, I.; Sokolsky, O.; Smith, J.; Keromytis, A.; Lee, W. “Dynamic trust management”, *Computer*, vol. 42–2, Feb 2009, pp. 44–52.

[Bla93] Blaze, M. “A cryptographic file system for unix”. In: Proceedings of the 1st ACM Conference on Computer and Communications Security, 1993, pp. 9–16.

[BPH14] Baumann, A.; Peinado, M.; Hunt, G. “Shielding applications from an untrusted cloud with haven”. In: Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), 2014, pp. 267–283.

[Bur] Burgess, M. “Google’s next submarine cable will connect Singapore to Australia”. Accessed Jan, 2017, Source: http://www.wired.co.uk/article/google-facebook-plcn-internet-cable.

[BW07] Boneh, D.; Waters, B. “Conjunctive, subset, and range queries on encrypted data”, *Theory of Cryptography*, vol. 4392, 2007, pp. 535–554.

[CF12] Chadwick, D. W.; Fatema, K. “A privacy preserving authorisation system for the cloud”, *Journal of Computer and System Sciences*, vol. 78–5, 2012, pp. 1359 – 1373, jCSS Special Issue: Cloud Computing 2011.

[Cla05] Clark, T. “Storage virtualization: technologies for simplifying data storage and management”. Addison-Wesley Professional, 2005, 234p.

[CLHK11] Chuang, I. H.; Li, S. H.; Huang, K. C.; Kuo, Y. H. “An effective privacy protection scheme for cloud computing”. In: Proceedings of the 13th International Conference on Advanced Communication Technology (ICACT2011), 2011, pp. 260–265.

[Clo] Cloud Security Alliance. “Privacy Level Agreement [V2]:A Compliance Tool for Providing Cloud Services in the European Union”. Accessed Nov, 2015, Source: https://downloads.cloudsecurityalliance.org/assets/research/pla/downloads/2015_05_28_PrivacyLevelAgreementV2_FINAL_JRS5.pdf.

[Cor] Corbet, J. “The 2.5 Kernel gets crypto”. Accessed Nov, 2015, Source: http://lwn.net/Articles/14157/.

[DGBL+16] Dowlin, N.; Gilad-Bachrach, R.; Laine, K.; Lauter, K.; Naehrig, M.; Wernsing, J. “Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy”, Technical Report, Microsoft Research, 2016, 12p.

[DW10] Diesburg, S. M.; Wang, A.-I. A. “A survey of confidential data storage and deletion methods”, *ACM Comput. Surv.*, vol. 43–1, Dec 2010, pp. 2:1–2:37.

[EKK13] Ebrahim, M.; Khan, S.; Khalid, U. “Symmetric algorithm survey: A comparative analysis”, *International Journal of Computer Applications*, vol. 61–20, 2013, pp. 12–19.

[EMEBHM16] El Makkaoui, K.; Ezzati, A.; Beni-Hssane, A.; Motamed, C. “Data confidentiality in the world of cloud”, *Journal of Theoretical and Applied Information Technology*, vol. 84, 2016, pp. 305–314.

[FGT14] Furfaro, A.; Garro, A.; Tundis, A. “Towards security as a service (secaas): On the modeling of security services for cloud computing”. In: Proceedings of the International Carnahan Conference on Security Technology (ICCST), 2014, pp. 1–6.

[FHMS12] Fahl, S.; Harbach, M.; Muders, T.; Smith, M. “Confidentiality as a service – usable security for the cloud”. In: Proceedings of the IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, 2012, pp. 153–162.

[FK09] Franciosi, F.; Knottenbelt, W. J. “Towards a QoS-aware Virtualised File System”. In: Proceedings of the UK Performance Engineering Workshop, 2009, pp. 1–12.

[FN94] Fiat, A.; Naor, M. “Broadcast encryption”, *Advances in Cryptology - CRYPTO’ 93*, vol. 773, 1994, pp. 480–491.

[FR14] Fielding, R.; Reschke, J. “Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing”. Accessed Jun, 2014, Source: http://www.ietf.org/rfc/rfc7230.txt, Jun 2014.

[Fru05] Fruhwirth, C. “New methods in hard disk encryption”, Technical Report, Institute for Computer Languages, Theory and Logic Group, Vienna University of Technology, 2005.

[FW12] Freitas, L.; Watson, P. “Formalising workflows partitioning over federated clouds: Multi-level security and costs”. In: Proceedings of the IEEE Eighth World Congress on Services, 2012, pp. 219–226.

[GA03] Govindavajhala, S.; Appel, A. W. “Using memory errors to attack a virtual machine”. In: Proceedings of the Symposium on Security and Privacy, 2003, pp. 154–165.

[Gen09] Gentry, C. “A fully homomorphic encryption scheme”, Ph.D. Thesis, Stanford University, 2009, 209p.

[GFC14] Gong, S.; Fu, S.; Chen, Z. “Research on the key technologies of storage virtualization”. In: Proceedings of the 2012 International Conference on Cybernetics and Informatics, 2014, pp. 1273–1279.

[GGT12] Goiri, Í.; Guitart, J.; Torres, J. “Economic model of a cloud provider operating in a federated cloud”, *Information Systems Frontiers*, vol. 14–4, 2012, pp. 827–843.

[GMR+12] Gonzalez, N.; Miers, C.; Redígolo, F.; Simplício, M.; Carvalho, T.; Näslund, M.; Pourzandi, M. “A quantitative analysis of current security concerns and solutions for cloud computing”, *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 1–1, 2012, pp. 11.

[Gooa] Google, Inc. “Encryption at rest in google cloud platform”. Accessed Jan, 2017, Source: https://cloud.google.com/security/encryption-at-rest/default-encryption/.

[Goob] Google, Inc. “Google cloud platform”. Accessed Jan, 2017, Source: https://cloud.google.com/.

[Gooc] Google Inc. “Virtual Network - Your Private Cloud Network”. Accessed Jan, 2017, Source: https://cloud.google.com/virtual-network/.

[Gou] Gough, V. “EncFS Encrypted Filesystem”. Accessed May, 2016, Source: http://www.arg0.net/#!encfs/c1awt.

[GR95] Guttman, B.; Roback, E. A. “Sp 800-12. an introduction to computer security: The nist handbook”, Technical Report, National Institute of Standards & Technology, Gaithersburg, MD, United States, 1995.

[Gud80] Gudes, E. “The design of a cryptography based secure file system”, *IEEE Transactions on Software Engineering*, vol. SE-6–5, Sept 1980, pp. 411–420.

[HA12] Hashemi, S. M.; Ardakani, M. R. M. “Taxonomy of the Security Aspects of Cloud Computing Systems - A Survey”, *International Journal of Applied Information Systems*, vol. 4–1, September 2012, pp. 21–28, published by Foundation of Computer Science, New York, USA.

[Hai16] Haimbala, J. “Avoiding dark cloud: Secure storage and trusted computing”, Ph.D. Thesis, University of Westminster, 2016, 73p.

[Hal07] Halcrow, M. “ecryptfs: A stacked cryptographic filesystem”, *Linux Journal*, vol. 2007–156, Apr 2007, pp. 2–12.

[HCAL17] Han, Y.; Chan, J.; Alpcan, T.; Leckie, C. “Using virtual machine allocation policies to defend against co-resident attacks in cloud computing”, *IEEE Transactions on Dependable and Secure Computing*, vol. 14–1, Jan 2017, pp. 95–108.

[HILM02] Hacigümüş, H.; Iyer, B.; Li, C.; Mehrotra, S. “Executing sql over encrypted data in the database-service-provider model”. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, 2002, pp. 216–227.

[HKH+10] Hiroaki, H.; Kamizuru, Y.; Honda, A.; Hashimoto, T.; Shimizu, K.; Yao, H. “Dynamic ip-vpn architecture for cloud computing”. In: Proceedings of the 8th Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT), 2010, pp. 1–5.

[HZKL10] Huang, D.; Zhang, X.; Kang, M.; Luo, J. “Mobicloud: Building secure cloud framework for mobile computing and communication”. In: Proceedings of the Fifth IEEE International Symposium on Service Oriented System Engineering, 2010, pp. 27–34.

[IHG10] Ibrahim, A. S.; Hamlyn-Harris, J. H.; Grundy, J. C. “Emerging security challenges of cloud virtual infrastructure”. In: Proceedings of the APSEC Workshop on Cloud Computing, 2010, pp. 1–6.

[IIES14] Irazoqui, G.; Inci, M. S.; Eisenbarth, T.; Sunar, B. “Fine grain cross-vm attacks on xen and vmware”. In: Proceedings of the IEEE Fourth International Conference on Big Data and Cloud Computing, 2014, pp. 737–744.

[IKC09] Itani, W.; Kayssi, A.; Chehab, A. “Privacy as a service: Privacy-aware data storage and processing in cloud computing architectures”. In: Proceedings of the Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, 2009, pp. 711–716.

[Int06] Intel Corporation. “Intel®Advanced Encryption Standard (AES) New Instructions Set”. , Intel White Paper, 2006, https://software.intel.com/sites/default/files/article/165683/aes-wp-2012-09-22-v01.pdf.

[Int14] Intel Corporation. “Software guard extensions programming reference”. , Intel White Paper, 2014, https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf.

[JR16] Jouini, M.; Rabai, L. B. A. “A multi-dimensional mean failure cost model to enhance security of cloud computing systems”, *International Journal of Embedded and Real-Time Communication Systems*, vol. 7–2, 2016, pp. 1–14.

[JZ11] Jamil, D.; Zaki, H. “Cloud computing security”, *International Journal of Engineering Science and Technology*, vol. 3–4, 2011, pp. 3478–3483.

[KA98] Kent, D. S. T.; Atkinson, R. “IP Encapsulating Security Payload (ESP)”. Accessed Nov, 2015, Source: https://rfc-editor.org/rfc/rfc2406.txt, Nov 1998.

[KD12] Kim, Y.; Doh, K.-G. “A trust management model for qos-based service selection”, *Information Security Applications*, vol. 7690, 2012, pp. 345–357.

[KGP+17] Kelbert, F.; Gregor, F.; Pires, R.; Köpsell, S.; Pasin, M.; Havet, A.; Schiavoni, V.; Felber, P.; Fetzer, C.; Pietzuch, P. “Securecloud: Secure big data processing in untrusted clouds”. In: Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE), 2017, pp. 282–285.

[KK04] Khanvilkar, S.; Khokhar, A. “Virtual private networks: an overview with performance evaluation”, *Communications Magazine, IEEE*, vol. 42–10, 2004, pp. 146–154.

[KKA14] Khalil, I. M.; Khreishah, A.; Azeem, M. “Cloud computing security: A survey”, *Computers*, vol. 3–1, 2014, pp. 1–35.

[KKKM13] Khan, A. N.; Kiah, M. M.; Khan, S. U.; Madani, S. A. “Towards secure mobile cloud computing: A survey”, *Future Generation Computer Systems*, vol. 29–5, 2013, pp. 1278 – 1299, special section: Hybrid Cloud Computing.

[KKTM11] Khadilkar, V.; Kantarcioglu, M.; Thuraisingham, B. M.; Mehrotra, S. “Secure data processing in a hybrid cloud”, *CoRR*, vol. abs/1105.1982, 2011, pp. 1–16.

[KL10] Kamara, S.; Lauter, K. “Cryptographic cloud storage”, *Financial Cryptography and Data Security*, vol. 6054, 2010, pp. 136–149.

[KLM16] Kashyap, R.; Louhan, P.; Mishra, M. “Economy driven real-time scheduling for cloud”. In: Proceedings of the 10th International Conference on Intelligent Systems and Control (ISCO), 2016, pp. 1–6.

[KRRR98] Knudsen, L.; Rijmen, V.; Rivest, R.; Robshaw, M. “On the design and security of rc2”. In: *Fast Software Encryption*, Vaudenay, S. (Editor), Springer Berlin Heidelberg, 1998, *Lecture Notes in Computer Science*, vol. 1372, pp. 206–221.

[LD93] Leutenegger, S. T.; Dias, D. “A modeling study of the tpc-c benchmark”. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, 1993, pp. 22–31.

[LLLZ16] Li, K.; Liu, C.; Li, K.; Zomaya, A. Y. “A framework of price bidding configurations for resource usage in cloud computing”, *IEEE Transactions on Parallel and Distributed Systems*, vol. 27–8, Aug 2016, pp. 2168–2181.

[LMN07] Lee, H. K.; Malkin, T.; Nahum, E. “Cryptographic strength of ssl/tls servers: Current and recent practices”. In: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, 2007, pp. 83–92.

[LS11] Liao, W.-H.; Su, S.-C. “A dynamic vpn architecture for private cloud computing”. In: Proceedings of the Fourth IEEE International Conference on Utility and Cloud Computing (UCC), 2011, pp. 409–414.

[LWW+10] Li, J.; Wang, Q.; Wang, C.; Cao, N.; Ren, K.; Lou, W. “Fuzzy keyword search over encrypted data in cloud computing”. In: Proceedings of the INFOCOM, 2010, pp. 1–5.

[McG06] McGraw, G. “Software Security: Building Security In”. Addison-Wesley Professional, 2006.

[McK09] McKinsey & Company. “Clearing the air on cloud computing”, Technical Report, "McKinsey & Company", 2009.

[MEH15] Makkaoui, K. E.; Ezzati, A.; Hssane, A. B. “Challenges of using homomorphic encryption to secure cloud computing”. In: Proceedings of the International Conference on Cloud Technologies and Applications (CloudTech’15), 2015, pp. 1–7.

[Mer03] Mercuri, R. T. “Analyzing security costs”, *Communications of the ACM*, vol. 46–6, 2003, pp. 15–18.

[MG11] Mell, P.; Grance, T. “The NIST definition of cloud computing”, *NIST special publication*, vol. 800–145, 2011, pp. 1–7.

[Mic] Microsoft Inc. “Azure Storage Service Encryption for Data at Rest”. Accessed Jan, 2017, Source: https://docs.microsoft.com/en-us/azure/storage/storage-service-encryption.

[Mil86] Miller, V. S. “Use of Elliptic Curves in Cryptography”. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 417–426.

[MJ15] Mahalle, S.; Jaiswal, R. “Article: Cloud computing security: A survey”, *International Journal of Computer Applications*, vol. 115–6, April 2015, pp. 21–25.

[MKL09] Mather, T.; Kumaraswamy, S.; Latif, S. “Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance”. O’Reilly Media, Inc., 2009.

[MNGV16] Mashayekhy, L.; Nejad, M. M.; Grosu, D.; Vasilakos, A. V. “An online mechanism for resource allocation and pricing in clouds”, *IEEE Transactions on Computers*, vol. 65–4, April 2016, pp. 1172–1184.

[MW11] Milojicic, D.; Wolski, R. “Eucalyptus: Delivering a Private Cloud”, *Computer*, vol. 44–4, April 2011, pp. 102–104.

[NC] Norcott, W. D.; Capps, D. “Iozone filesystem benchmark”. Accessed Nov, 2015, Source: http://www.iozone.org.

[NFL+14] Naylor, D.; Finamore, A.; Leontiadis, I.; Grunenberger, Y.; Mellia, M.; Munafò, M.; Papagiannaki, K.; Steenkiste, P. “The cost of the "s" in https”. In: Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies, 2014, pp. 133–140.

[NSMZ16] Noor, T. H.; Sheng, Q. Z.; Maamar, Z.; Zeadally, S. “Managing trust in the cloud: State of the art and research challenges”, *Computer*, vol. 49–2, Feb 2016, pp. 34–45.

[OKS08] Opitz, A.; König, H.; Szamlewska, S. “What does grid computing cost?”, *Journal of Grid Computing*, vol. 6–4, 2008, pp. 385–397.

[Ope] OpenStack Foundation. “OpenStack - Swift”. Accessed Jan, 2017, Source: http://swift.openstack.org/.

[PGM17] Paladi, N.; Gehrmann, C.; Michalas, A. “Providing user security guarantees in public infrastructure clouds”, *IEEE Transactions on Cloud Computing*, vol. 5–3, July 2017, pp. 405–419.

[PKZ11] Puttaswamy, K. P. N.; Kruegel, C.; Zhao, B. Y. “Silverline: Toward data confidentiality in storage-intensive cloud applications”. In: Proceedings of the 2Nd ACM Symposium on Cloud Computing, 2011, pp. 10:1–10:13.

[PLM+11] Popa, R. A.; Lorch, J. R.; Molnar, D.; Wang, H. J.; Zhuang, L. “Enabling security in cloud storage slas with cloudproof”. In: Proceedings of the USENIX Conference on USENIX Annual Technical Conference, 2011, pp. 31–31.

[PRZB11] Popa, R. A.; Redfield, C. M. S.; Zeldovich, N.; Balakrishnan, H. “Cryptdb: Protecting confidentiality with encrypted query processing”. In: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, 2011, pp. 85–100.

[PZB11] Popa, R. A.; Zeldovich, N.; Balakrishnan, H. “CryptDB: A practical encrypted relational DBMS”, Technical Report, DSpace - MIT, 2011.

[Rac] RackSpace US Inc. “RackSpace - The open cloud company”. Accessed Nov, 2013, Source: http://rackspace.com.

[RAD78] Rivest, R. L.; Adleman, L.; Dertouzos, M. L. “On data banks and privacy homomorphisms”, *Foundations of secure computation*, vol. 4–11, 1978, pp. 169–180.

[Raj11] Rajkumar Buyya, James Broberg, A. M. G. “Cloud Computing Principles and Paradigms (Wiley Series on Parallel and Distributed Computing)”. Wiley, 2011, 1 ed., 664p.

[RC11] Rocha, F.; Correia, M. “Lucy in the sky without diamonds: Stealing confidential data in the cloud”. In: Proceedings of the IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W), 2011, pp. 129–134.

[RC15] Rahman, N. H. A.; Choo, K.-K. R. “A survey of information security incident handling in the cloud”, *Computers & Security*, vol. 49, 2015, pp. 45 – 69.

[RCM09] Roschke, S.; Cheng, F.; Meinel, C. “Intrusion detection in the cloud”. In: Proceedings of the Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, 2009, pp. 729–734.

[RGVM13] Rocha, F.; Gross, T.; Van Moorsel, A. “Defense-in-depth against malicious insiders in the cloud”. In: Proceedings of the IEEE International Conference on Cloud Engineering (IC2E), 2013, pp. 88–97.

[RJX08] Riley, R.; Jiang, X.; Xu, D. “Guest-transparent prevention of kernel rootkits with vmm-based memory shadowing”. In: Proceedings of the Recent Advances in Intrusion Detection, 2008, pp. 1–20.

[RK14] Roman, M.; Khan, S. “Cloud computing security: A survey”, *Global Journal of Technology*, vol. 8, 2014, pp. 15–28.

[RSA78] Rivest, R. L.; Shamir, A.; Adleman, L. “A method for obtaining digital signatures and public-key cryptosystems”, *Commun. ACM*, vol. 21–2, Feb 1978, pp. 120–126.

[RTSS09] Ristenpart, T.; Tromer, E.; Shacham, H.; Savage, S. “Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds”. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, 2009, pp. 199–212.

[SC17] Singh, A.; Chatterjee, K. “Cloud security issues and challenges: A survey”, *Journal of Network and Computer Applications*, vol. 79, 2017, pp. 88 – 115.

[SCF+15] Schuster, F.; Costa, M.; Fournet, C.; Gkantsidis, C.; Peinado, M.; Mainar-Ruiz, G.; Russinovich, M. “Vc3: Trustworthy data analytics in the cloud using sgx”. In: Proceedings of the IEEE Symposium on Security and Privacy, 2015, pp. 38–54.

[SDR17] Storch, M.; De Rose, C. A. F. “Cloud storage cost modeling for cryptographic file systems”. In: Proceedings of the 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), 2017, pp. 9–14.

[SDRZM15] Storch, M.; De Rose, C. A.; Zorzo, A. F.; Michelin, R. “Multi-channel Secure Interconnection Desgin for Hybrid Clouds”. In: Proceedings of The Fourteenth International Conference on Networks, ICN15, 2015, pp. 67–73.

[SGR09] Santos, N.; Gummadi, K. P.; Rodrigues, R. “Towards trusted cloud computing”. In: Proceedings of the 2009 Conference on Hot Topics in Cloud Computing, 2009, pp. 1–5.

[SJL+16] Seol, J.; Jin, S.; Lee, D.; Huh, J.; Maeng, S. “A trusted iaas environment with hardware security module”, *IEEE Transactions on Services Computing*, vol. 9–3, May 2016, pp. 343–356.

[Sta01] Standard, N.-F. “Announcing the Advanced Encryption Standard (AES)”, *Federal Information Processing Standards Publication*, vol. 197, 2001, pp. 1–51.

[Sta10] Stallings, W. “Cryptography and Network Security: Principles and Practice”. Upper Saddle River, NJ, USA: Prentice Hall Press, 2010, 5th ed., 752p.

[Sta11] Stamp, M. “Information Security: Principles and Practice”. Wiley Publishing, 2011, 2nd ed., 606p.

[STT+12] Sharma, B.; Thulasiram, R. K.; Thulasiraman, P.; Garg, S. K.; Buyya, R. “Pricing cloud compute commodities: a novel financial economic model”. In: Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID 2012), 2012, pp. 451–457.

[STTB15] Sharma, B.; Thulasiram, R. K.; Thulasiraman, P.; Buyya, R. “Clabacus: A risk-adjusted cloud resources pricing model using financial option theory”, *IEEE Transactions on Cloud Computing*, vol. 3–3, July 2015, pp. 332–344.

[Tox14] Toxen, B. “The nsa and snowden: Securing the all-seeing eye”, *Queue*, vol. 12–3, Mar 2014, pp. 40–51.

[Tru] Trusted Computing Group. “Trusted computing”. Accessed Jan, 2017, Source: https://trustedcomputinggroup.org/trusted-computing/.

[TVRB15] Toosi, A. N.; Vanmechelen, K.; Ramamohanarao, K.; Buyya, R. “Revenue maximization with optimal capacity control in infrastructure as a service cloud markets”, *IEEE Transactions on Cloud Computing*, vol. 3–3, July 2015, pp. 261–274.

[vDGHV10] van Dijk, M.; Gentry, C.; Halevi, S.; Vaikuntanathan, V. “Fully homomorphic encryption over the integers”, *Advances in Cryptology - EUROCRYPT 2010*, 2010, pp. 24–43.

[VMC08] Viega, J.; Messier, M.; Chandra, P. “Network Security with OpenSSL: Cryptography for Secure Communications”. O’Reilly Media, 2008, 384p.

[VT14] Varadharajan, V.; Tupakula, U. “Security as a service model for cloud environment”, *IEEE Transactions on Network and Service Management*, vol. 11–1, March 2014, pp. 60–75.

[Wat12] Watson, P. “A multi-level security model for partitioning workflows over federated clouds”, *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 1–1, Jul 2012, pp. 15.

[WDZ03] Wright, C.; Dave, J.; Zadok, E. “Cryptographic file systems performance: What you don’ t know can hurt you”. In: Proceedings of the Second IEEE International Security in Storage Workshop (SISW ’03), 2003, pp. 47–47.

[WL08] Wang, Z.; Lee, R. B. “A novel cache architecture with enhanced performance and security”. In: Proceedings of the 41st IEEE/ACM International Symposium on Microarchitecture, 2008, pp. 83–93.

[WRAK13] Waqar, A.; Raza, A.; Abbas, H.; Khan, M. K. “A framework for preservation of cloud users’ data privacy using dynamic reconstruction of metadata”, *Journal of Network and Computer Applications*, vol. 36–1, 2013, pp. 235 – 248.

[WTM+14] Win, T. Y.; Tianfield, H.; Mair, Q.; Said, T. A.; Rana, O. F. “Virtual machine introspection”. In: Proceedings of the 7th International Conference on Security of Information and Networks, 2014, pp. 405–410.

[WWRL10] Wang, C.; Wang, Q.; Ren, K.; Lou, W. “Privacy-preserving public auditing for data storage security in cloud computing”. In: Proceedings of the IEEE INFOCOM, 2010, pp. 1–9.

[WZC+14] Wei, L.; Zhu, H.; Cao, Z.; Dong, X.; Jia, W.; Chen, Y.; Vasilakos, A. V. “Security and privacy for storage and computation in cloud computing”, *Inf. Sci.*, vol. 258, Feb 2014, pp. 371–386.

[WZL15] Weng, C.; Zhan, J.; Luo, Y. “Tsac: Enforcing isolation of virtual machines in clouds”, *IEEE Transactions on Computers*, vol. 64–5, May 2015, pp. 1470–1482.

[Yon] Yonan, J. “OpenVPN–an open source SSL VPN solution”. Accessed Nov, 2013, Source: https://openvpn.net/.

[YWRL10] Yu, S.; Wang, C.; Ren, K.; Lou, W. “Achieving secure, scalable, and fine-grained data access control in cloud computing”. In: Proceedings of the INFOCOM, 2010, pp. 1–9.

[YYA09] Yau, S. S.; Yin, Y.; An, H. G. “An adaptive tradeoff model for service performance and security in service-based systems”. In: Proceeding of the IEEE International Conference on Web Services, 2009, pp. 287–294.

[ZCB10] Zhang, Q.; Cheng, L.; Boutaba, R. “Cloud computing: state-of-the-art and research challenges”, *Journal of Internet Services and Applications*, vol. 1–1, 2010, pp. 7–18.

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br