SCUOLA di DOTTORATO
Università degli Studi Mediterranea di Reggio Calabria

In last years, Smart Cities are becoming very popular within the scientific community. There are many definition of Smart Cities: according to Giffinger a city becomes smart only if the performance indicators improve for several interconnected areas, such as economy, people, governance, environment, living and mobility; according to other authors, Smart Cities have to include e-health.

In this thesis, we proposed innovative systems for Smart Cities.

First of all we described the basic characteristics and infrastructure of these services.

In particular, we focus on four aspects. First, we analyse the overhead introduced by the container virtualization. Second, we discuss how to distribute microservices among federated tenants. Third, we evaluate the overhead introduced by securing a Message Oriented Middleware able to manage microservices. Finally, we compare different Secret Share techniques for storing files.

Then we considered four specific use-cases.In particular, we discussed: i) a Big-Data analytics platform for telemedicine that allows physicians to follow the evolution of the care of their patients by looking at coloured circles; ii) an innovative tool for the management of Electronic Health Records (EHRs) that, by using the public Cloud, helps hospital practitioners to find the right EHR per each patient; iii) a way to manage pictures taken from drones adding redundancy and security by using Nested Secret Share algorithms; and iv) a system for traffic monitoring and reducing vehicular accidents applicable both to human driven vehicles and autonomous ones.

The proposed systems were validated through specific experiments.

SCUOLA DI DOTTORATO DELL'UNIVERSITA' DEGLI STUDI *MEDITERRANEA* DI REGGIO CALABRIA

Antonino Galletta

Innovative Cloud and Edge based systems

Università degli Studi Mediterranea di Reggio Calabria

SCUOLA DI DOTTORATO DELL'UNIVERSITA' DEGLI STUDI MEDITERRANEA DI REGGIO CALABRIA

DOTTORATO di RICERCA in INGEGNERIA CIVILE, AMBIENTALE e della SICUREZZA

Dottorato consorziato tra
Università degli StudiMediterraneadi Reggio Calabria
Università degli Studi di Messina

Coordinatore: Prof. Felice Arena

Dott. Ing. Antonino Galletta

Innovative Cloud and Edge based systems for smart life

Tutor: Prof. Massimo Villari
Co-tutor: Dr. Maria Fazio, Dr. Antonio Celesti
Curriculum "Curriculum in Scienze e Tecnologie, Materiali, Energia e Sistemi Complessi per il Calcolo Distribuito e le Reti " - SSD ING-INF/05
AA 2018/19 - XXXII ciclo

Universitá degli Studi "Mediterranea" di Reggio Calabria

Dipartimento di Ingegneria Civile, dell'Energia, dell'Ambiente e dei Materiali

Dottorato di Ricerca in Ingegneria Civile Ambientale e della Sicurezza

Curriculum in Scienze e Tecnologie, Materiali, Energia e Sistemi Complessi per il Calcolo Distribuito e le Reti SSD ING/INF05
Ciclo XXXII

Innovative Cloud and Edge based systems for smart life

AUTHOR:
ANTONINO GALLETTA

HEAD OF THE DOCTORAL SCHOOL:
Prof. Dr. Felice Arena

ADVISOR:
Prof. Dr. Massimo Villari

CO-ADVISORS:
Dr. Maria Fazio
Dr. Antonio Celesti

Accademic Year 2018-2019

Abstract

In the last years, Smart Cities are becoming very popular within the scientific community. There are many definitions of Smart Cities: according to Giffinger a city becomes smart only if the performance indicators improve for several interconnected areas, such as economy, people, governance, environment, living and mobility; according to other authors, Smart Cities have to include e-health. In this thesis, we proposed innovative systems for Smart Cities. First of all, we described the basic characteristics and infrastructure of these services. In particular, we focus on four aspects. First, we analyze the overhead introduced by container virtualization. Second, we discuss how to distribute microservices among federated tenants. Third, we evaluate the overhead introduced by securing a Message Oriented Middleware able to manage microservices. Finally, we compare different Secret Share techniques for storing files. Then we considered four specific use-cases. In particular, we discussed: i) a Big-Data analytics platform for telemedicine that allows physicians to follow the evolution of the care of their patients by looking at colored circles; ii) an innovative tool for the management of Electronic Health Records (EHRs) that, by using the public Cloud, helps hospital practitioners to find the right EHR per each patient; iii) a way to manage pictures taken from drones adding redundancy and security by using Nested Secret Share algorithms; and iv) a system for traffic monitoring and reducing vehicular accidents applicable both to human-driven vehicles and autonomous ones. The proposed systems were validated through specific experiments.

Keywords: Cloud, Edge, IoT, Smart Cities, e-health

Contents

List of Figures

List of Tables

Earlier Publications

This thesis is the outcome of the doctoral degree started three years ago. It is based on selected works (listed here below) already published, or under review, in scientific conferences proceedings and journals. Parts of these papers are contained in verbatim.

[1] 2018 Carnevale, Lorenzo; Celesti, Antonio; Di Pietro, Maria; Galletta, Antonino; How to conceive future mobility services in smart cities according to the Fiware Frontiercities experience; IEEE Cloud Computing;

[2] 2019 Celesti, Antonio; Mulfari, Davide; Galletta, Antonino; Fazio, Maria; Carnevale, Lorenzo; Villari, Massimo; A study on container virtualization for guarantee quality of service in Cloud-of-Things; Future Generation Computer Systems;

[3] 2017 Galletta, Antonino; Carnevale, Lorenzo; Celesti, Antonio; Fazio, Maria; Villari, Massimo; Boss: A multitenancy ad-hoc service orchestrator for federated openstack clouds; 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud);

[4] 2019 Celesti, Antonio; Fazio, Maria; Galletta, Antonino; Carnevale, Lorenzo; Wan, Jiafu; Villari, Massimo; An approach for the secure management of hybrid cloud-edge environments Future Generation Computer Systems;

[5] 2019 Galletta, Antonino; Taheri, Javid; Villari, Massimo; On the Applicability of Secret Share Algorithms for Saving Data on IoT, Edge and Cloud; Devices 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and

IEEE Smart Data (SmartData);

[6] 2018 Galletta, Antonino; Carnevale, Lorenzo; Bramanti, Alessia; Fazio, Maria; An innovative methodology for big data visualization for telemedicine; IEEE Transactions on Industrial Informatics;

[7] 2019 Galletta, Antonino; Buzachis, Alina; Fazio, Maria; Celesti, Antonio; Taheri, Javid; Villari, Massimo; Smart hospitals enabled by edge computing; (Under review);

[8] 2019 Galletta, Antonino; Taheri, Javid; Fazio, Maria; Celesti, Antonio; Villari, Massimo; Enhancing the security and privacy of drones by using Nested Secret Share techniques; ACM Transactions on Storage (Under review);

[9] 2017 Celesti, Antonio; Galletta, Antonino; Carnevale, Lorenzo; Fazio, Maria; Lay-Ekuakille, Aime; Villari, Massimo; An IoT cloud system for traffic monitoring and vehicular accidents prevention based on mobile sensor data processing; IEEE Sensors Journal.

CHAPTER 1

Introduction

Recently, new achievements in ICT (Information and Communication Technologies) are changing our lives. New concepts and paradigms have been deployed. Among these, Smart City has had a wide diffusion all over the world due to the growing political and scientific involvement and environmental emergencies. Among many Smart City definitions, Giffinger et Al. [10] wrote that a city becomes smart only if the performance indicators improve for several interconnected areas, such as economy, people, governance, environment, living and mobility. According to [11] and [12], e-health is a subarea of the ICT strategies of smart cities. Smart mobility is emerging as a solution for many issues related to air and noise pollution, traffic congestion [13], transport and goods distribution slowness. Many research companies have investigated Smart Mobility trends in the last years. In 2016, Navigation Research [14] highlighted that more than 50% of the Smart City design referred to sustainable and Smart Mobility solutions for urban areas. The same analysis foresaw how the smart mobility market will be worth over 20 billion US Dollars by 2020, whereas Market, in [15], foresaw a technological investment growth from 410 billion US Dollars in 2014 to 1100 billion US Dollars in 2019. All these investments are also possible thanks to the increasing number of connected Internet of Things (IoT) devices spread over the Internet, foreseen by Gartner (around 9.7 billion by 2020) [16].

In this thesis, we will propose new services for smart life. In particular, first of all we will discuss and analyse basic characteristics of these innovative services. In order to do that we carried out four research questions:

- **Question 1:** *How to deploy services on distributed Cloud-Iot environments?*
- **Question 2:** *How to deploy microservices on different federated tenants?*
- **Question 3:** *How to manage virtualised resources in a secure manner?*
- **Question 4:** *How to store files adding redundancy and security?*

Then, considering four specific use-cases, we extended research questions as follows:

- **Question 5:** *How to manage efficiently clinical Big-Data?*
- **Question 6:** *How to manage efficiently Electronic Health Records (EHRs)*
- **Question 7:** *How to add long term availability and security to drones?*
- **Question 8:** *How to reduce car accidents in Smart Cities?*

1.1 Scientific Contributions

In this Section we will answer to research questions pointed out in the previous Section.

Question 1: *How to deploy services on distributed Cloud-Iot environments?*

Contribution 1: Containerization of Services in Cloud-Iot environments.

We discuss the containerization of services and analyse the overhead introduced by container virtualization when multiple concurrent micro-services are executed in parallel within the same IoT device in order to optimize both virtual sensing and actuating resources.

Details related to this contribution and experiments are discussed in Chapter 3. Contribution 1 has originally been presented in [2].

Question 2: *How to deploy microservices on different federated tenants?*

Contribution 2: Distributing microservices among federated tenants.

We proposed an innovative way to deploy microservices among several federated Openstack Clouds. In particular, we developed BOSS Broker OpenStack Service an OpenStack orchestrator that starting from an ad-hoc OpenStack-based Heat Orchestration Template (HOT) service manifest produces several HOT microservice manifests including deployment instructions for involved Federated Clouds. Therefore, users can formalize advanced deployment constrains in terms of data export control, networking, and disaster recovery.

Further details and experiments are discussed in Chapter 4. Contribution 2 has originally been presented in [3].

Question 3: *How to manage virtualised resources in a secure manner?*

Contribution 3: Enabling the Secure Management of Hybrid Cloud-Edge Environments.

We discussed the impact of the security of a MOM (Message Oriented Middleware) for Cloud-to-Edge based on an IMP (Instant-Message Protocol). More specifically, we discuss how the overall communication system can be secured considering both the digital signature and data encryption mechanisms. In order to plan a security strategy, we analyse both the intra-module and the inter-module communications of a MOM based on XMPP.

Chapter 5 discusses more in details this contribution. Contribution 3 has originally been presented in [4].

Question 4: *How to store files adding redundancy and security?*

Contribution 4: Enabling the secure store of files by using Secret Share techniques.

We discussed the use of Secret Share techniques for storing files. In particular, our contribution in this work can be summarised as the followings. Firstly, we analyse and compare Secret Share algorithms; and secondly, we identify the most suitable environment to run each Secret Share algorithm considering the size of the file and the computational capacity of the node. To be more specific, we discuss about Shamir's Secret Share schema and the Redundant Residue Number System (RRNS).

These techniques are analysed in details in Chapter 6. Contribution 4 has originally been presented in [5].

Question 5: *How to manage efficiently clinical Big-Data?*

Contribution 5: An innovative Big-Data visualization tool for telemedicine.

We present a new graphical tool for the visualization of health data, that can be easily used for monitoring health status of patients remotely. After having selected a treatment for a specific patient, the doctor or the medical operator is able to follow the evolution of his/her care. Moreover, selecting a zone of interest simply drawing a polygon on the map, it is possible to select one of the included markers for visualizing a patient health status by means of a circular view.

Chapter 7 provides details of the proposed system. Contribution 5 has originally been presented in [6].

Question 6: *How to manage efficiently Electronic Health Records (EHRs)*

Contribution 6: An innovative way for managing Electronic Health Records (EHRs) and avoid mistakes.

We propose a tool for the management of EHRs. In particular, the system by using the RRNS (Redundant Residue number System) splits and stores EHRs on public Clouds. In order to avoid any mistake, due to the exchange of EHRs, and to perform the recomposition on-demand specific Edge devices are installed on ambulances and hospital beds.

Further details are discussed in Chapter 8. Contribution 6 has originally been presented in [7].

Question 7: *How to add long term availability and security to drones?*

Contribution 7: Adding long term availability and security to drones by using Nested Secret Share techniques.

We considered fleets of drones that go in mission to take photos of enemies, we analyse Shamir's Secret Share schema and the Redundant Residue Number System (RRNS). Our contribution can be summarised as the followings. Firstly, we analyse and compare SS (Secret Share) algorithms; then we propose NSS (Nested Secret Share) techniques; and finally we compare SS and NSS discussing the the suitability for drones.

Further details are discussed in Chapter 9. Contribution 7 has originally been presented in [8].

Question 8: *How to reduce car accidents in smart Cities?*

Contribution 8: An innovative system for traffic monitoring and reducing vehicular accidents.

We propose a fast real time processing of Big traffic data able to prevent accidents. In particular, considering a real reference scenario of mobile GPS-based sensors installed in private/public transportation and volunteer vehicles, we discuss an IoT Cloud system for traffic monitoring and alert notification based on OpenGTS and MongoDB for the fast processing of big traffic data. Such a system, can be applied also for autonomous vehicles.

The proposed system is discussed in Chapter 10. Contribution 8 has originally been presented in [9].

1.2 Structure of the Thesis

In Chapter 2, we show basic technologies used on the rest of the thesis.

Chapters 3 to 10 compose the core of the thesis. In particular Chapters 3-6 describe the basic characteristics of e-health and Smart Cities solutions presented in Chapters 7-10. More in details:

Chapter 3 discusses how to containerise services and measure the overhead introduced by containerisation.

Chapter 4 discusses how to deploy microservices among several federated Clouds.

Chapter 5 presents a secure way to manage a Message Oriented Middleware by using digital signature and encryption.

Chapter 6 analyses Secret Share techniques to store data. In particular, a comparison between Shamir's secret schema and Redundant Residue Number System is provided.

Chapter 7 provide a specific tool for Big-Data visualization for telemedicine.

Chapter 8 discusses the applicability of Secret Share techniques in hospital for the management of Electronic Health Records (EHR).

In Chapter 9, considering a Smart City scenario composed of drones that go in missions to take pictures, we proposed Nested Secret Share techniques in order to enhance the level of redundancy and security of data.

Chapter 10 discusses a solution to monitor vehicles and reduce the number of accidents. The system can be applied on both autonomous vehicles or not.

Each core Chapter present more or less the same structure composed of:

1. introduction to the problem;

2. review of the state of the art;

3. discussion about background and specific technologies;

4. design and implementation of the system;

5. performance evaluation;

6. conclusion summarising specific contribution.

Finally, Chapter 11 conclude the thesis providing lights on the future.

CHAPTER 2

Background

This chapter provides background information about technologies that we use as basis of this thesis.

Cloud Computing

Cloud Computing, according to NIST definition [17], is a computation paradigm that enable the ubiquitous access to virtualised resources configured on-demand. It is composed of five essential characteristics, provides services in according to three service models, and it is based on four deployment models.

Essential characteristics:

1. **On-demand self-service.** The provisioning of resources is done in real time in according to the workload;

2. **Broad network access.** Services are provided over the network by using standard APIs (Application Programming Interfaces) that allow the use of any device;

3. **Resource pooling.** Resources of Cloud providers are pooled to serve multiple customers at the same time and can be reassigned based on the workload;

4. **Rapid elasticity.** Resources assigned to services must be scaled based on the workload;

5. **Measured service.** Physical resources are monitored and optimised by using metering tools.

Service Models:

1. **Software as a Service (SaaS).** Customers will use applications and facilities running on a Cloud platform. Services are accessed from various devices either dummy devices and users cannot control any hardware resource.

2. **Platform as a Service (PaaS).** Cloud users' can create their services and applications by using specific programming languages and libraries supported by the Cloud provider. Also in this case they cannot control any hardware resource.

3. **Infrastructure as a Service (IaaS).** This service model allows Cloud users' to provision hardware resources and to deploy and run any arbitrary software.

Figure 2.1 provide a graphical representation of Cloud service models.

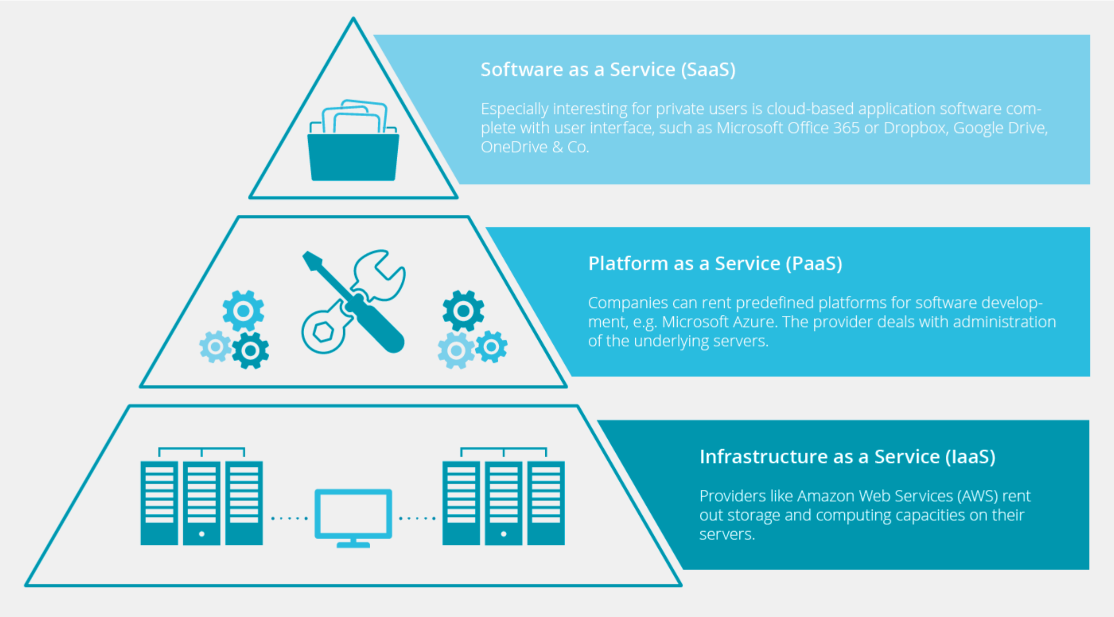

Software as a Service (SaaS)
Especially interesting for private users is cloud-based application software complete with user interface, such as Microsoft Office 365 or Dropbox, Google Drive, OneDrive & Co.
Platform as a Service (PaaS)
Companies can rent predefined platforms for software development, e.g. Microsoft Azure. The provider deals with administration of the underlying servers.
Infrastructure as a Service (IaaS)
Providers like Amazon Web Services (AWS) rent out storage and computing capacities on their servers.

Figure 2.1: Cloud service models Source: `https://www.plesk.com/blog/various/iaas-vs-paas-vs-saas-various-cloud-service-models-compared/`

Deployment Models:

1. **Private Cloud.** The Cloud data center is reserved for a specific company or organization.

2. **Community Cloud.** The Cloud data center is reserved for clients that belong to multiple organizations that share the same mission.

3. **Public Cloud.** The Cloud data center is accessible to anyone.

4. **Hybrid Cloud.** This model consists on the use of two or more models discussed above.

2.1 Openstack

Openstack is an IaaS cloud provider, it is an open source project deployed in collaboration with NASA (National Aeronautics and Space Administration). It is composed by several services able to communicate each other through messages. Each module exposes its service to the outside world by restful APIs. In the figure2.2 is showed an architectural schema of a typical Openstack installation.

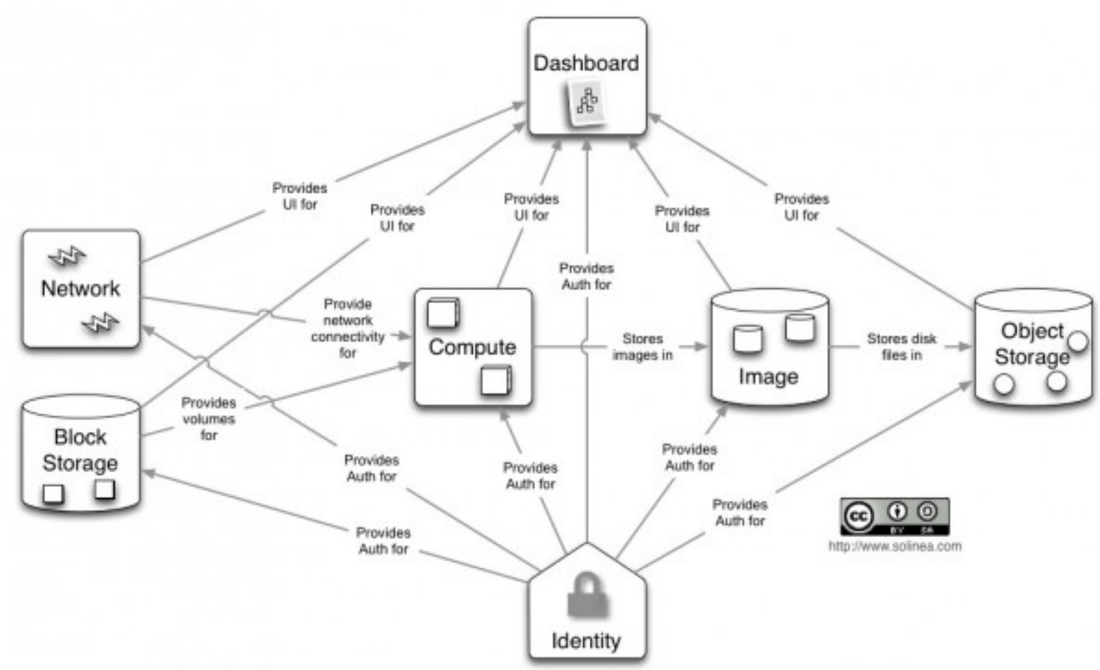


Figure 2.2: Openstack schema Source: `http://www.guruadvisor.net/images/numero0/openstack_logical_architecture1.jpg`

In particular it is composed by seven blocks with specific tasks. As showed in the picture only identity and dashboard services are directly connected to other all services.

In the next paragraphs we will describe briefly each block showed in figure2.2.

2.1.1 Keystone

Keystone is the module that provides the Identity service in terms of authentication and authorization. It define several concepts in order to provide its service:

1. **User**: an entity that uses the services;

2. **Role** includes a list of rights and privileges. It represent the position in the project;
3. **Tenant** or **Project** represent a collection of users that share the same resources;
4. **Token** is simply an apha-numeric string used for the authentication;
5. **Password** a string used in the authentication phase to retrieve the Token;
6. **Domain** represent the set of projects groups and users.

In the default installation Keystone uses a RDBMS SQL-like (MariaDB) in order to store the right now mentioned informations, but it can be attached to an external LDAP service. Keystone provides two kinds of services: authentication and authorization. Authentication service checks if the provided token or credentials are valid. Authorization service, instead, work in a next phase; it checks if the provided token could make a requested action.
In the figure2.3 is showed the keystone work-flow. In particular is depicted an user that uses its credentials in order to retrieve a Token, using that he makes a request to a generic service. The service, using Keystone features, checks if the provided Token is valid to make the requested action. Based on Keystone response the service make a specific action.

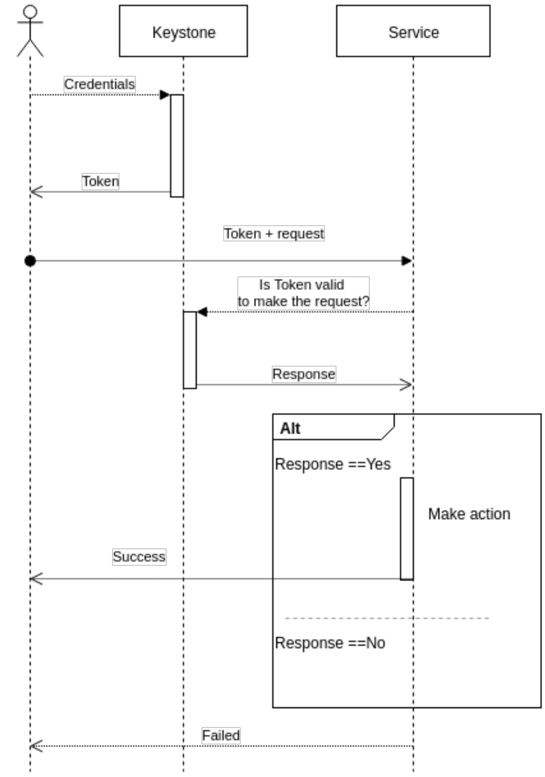


Figure 2.3: Keystone workflow

2.1.2 Nova

Nova in one of the original components in the first release (Austin). It implements the compute service with high-scalability functions. Its structure is very complex. Nova interacts strongly with other OpenStack services. Uses Keystone features in order to implement authentication and authorization functionalities, from Glance takes the image that uses in order to deploy the Virtual Machines, interacts with Neutron to give network connectivity to the VMs, and by Horizon receives inputs from users.

2.1.3 Neutron

Neutron implements network services, is able to manage Layer 2 and 3 of the ISO/OSI Stack. It provide virtual routers, virtual switches, SDN controller ect. Neutron cloud be used joined with Open vSwitch and OVN in order to allow Virtual Machines to communicate with physical switches and other network hardware. The version 8, actual stable, is not compliant with OVN. From Neutron version 9 in order to join Neutron and OVN it's needed to install the driver: `networking-ovn`.

2.1.4 Glance

The image manager service is provided by Glance. It stores images that could be used by Nova in order to start new virtual machines. It also provide a functionality to snapshotting running VMs, in particular, starting from a running Virtual Machine Glance is able to create a new image that could be started. Starting from the Openstack Newton version the Glance APIs v1 will be deprecated in favor of v2.

2.1.5 Swift

Swift implements Object Storage service. It is able to manage a lot of unstructured data, is widely scalable. Swift could work in a single node or multi node configurations. Thanks to his replication features Swift offers redundancy and high availability. Exist also a commercial version called SwiftStack. From an high level point of view the structure of Swift, as showed in figure2.4, could be divided in two levels: **Proxy Node** and **Storage Node**.

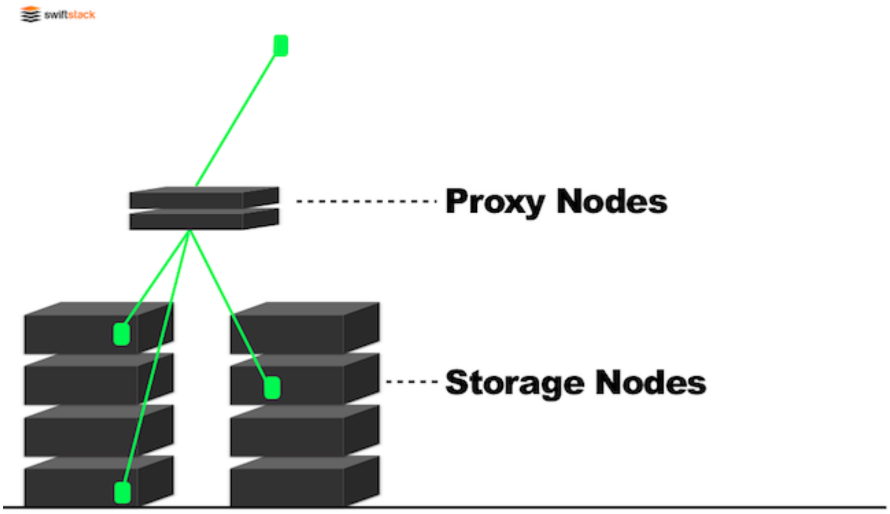


Figure 2.4: Swift structure Source: `https://swiftstack.com/images/posts/swift%2Dglobal-replication/replica-overview.png`

Proxy Node works as interface between Swift and user applications.
The Storage Node contains user data, have an internal structure like a file system with the fixed level number 3. A graphical representation is provided in figure2.5.

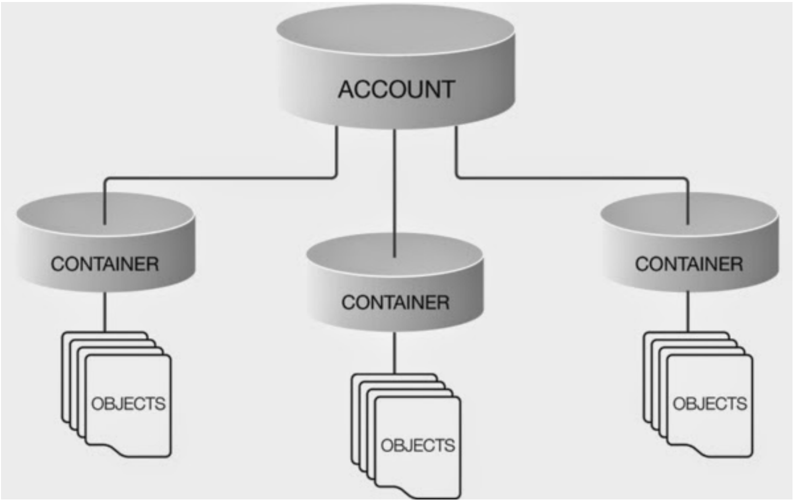


Figure 2.5: Swift Storage Node structure Source: `https://swiftstack.com/static/global/images/swift_architecture_aco.jpg`

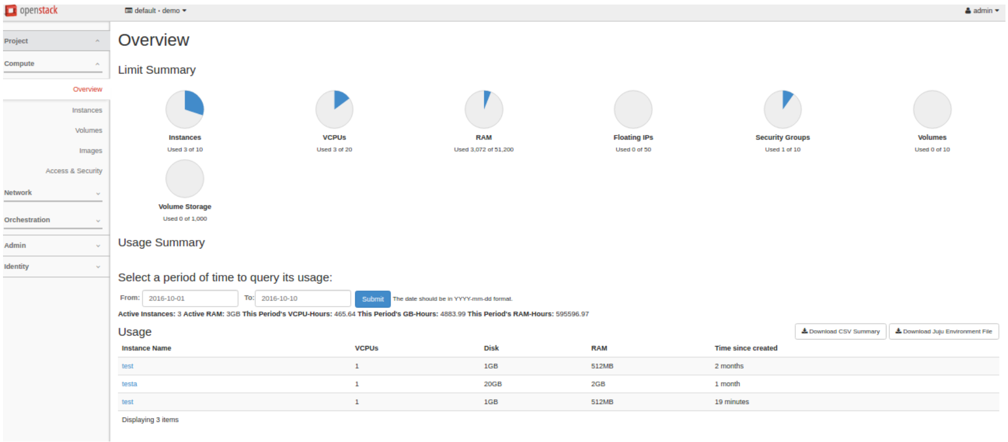
Account layer contains account meta-data and the list of containers. Container layer contains container meta-data and the list of user files. Object layer contains object and related meta-data.

2.1.6 Cinder

Cinder provides the Block Storage service, it is used in order to add persistent storage to the deployed machines. These block devices are known as Cinder volumes. In the earlier Openstack versions Cinder was a component of Nova, starting from 2012 became an independent project. The current version is 8.1.1.

2.1.7 Horizon

Horizon is the Openstack Dashboard. It allows user to interact with all Openstack Services. In the figure2.6 is showed a screenshot.



Overview

Limit Summary

Instances: Used 3 of 10
VCPUs: Used 3 of 20
RAM: Used 3,072 of 51,200
Floating IPs: Used 0 of 50
Security Groups: Used 1 of 10
Volumes: Used 0 of 10
Volume Storage: Used 0 of 1,000

Usage Summary

Select a period of time to query its usage:

From: 2016-10-01 To: 2016-10-10 The date should be in YYYY-mm-dd format.

Active Instances: 3 **Active RAM:** 3GB **This Period's VCPU-Hours:** 465.64 **This Period's GB-Hours:** 4883.99 **This Period's RAM-Hours:** 595596.97

Usage

Instance Name	VCPUs	Disk	RAM	Time since created
test	1	1GB	512MB	2 months
testa	1	20GB	2GB	1 month
test	1	1GB	512MB	19 minutes

Displaying 3 items

Figure 2.6: Horizon screenshot

2.1.8 Heat

Heat provide the orchestration service, was introduced from Openstack IceHouse release. It is composed by four services:

1. **heat**: a CLI able to communicate with heat-api;
2. **heat-api**: provider restful API, is able to communicate with heat-engine;
3. **heat-api-cfg**: is able to interact with AWS CloudFormation, in fact provides Query API compatible to AWS CloudFormation and convert query AWS CloudFormation style to RPC, suitable for heat-engine;
4. **heat**: is able to digest template and orchestrate the cloud Infrastructure.

Heat uses a declarative template YAML based in order to provide its services, this template is called **HOT** Heat Orchestration Template, il listing2.1 is showed a basic example, for future details visit the `officialguide`.

Listing 2.1: Basic HOT template

```
heat_template_version: 2015-04-30
description: Simple template to deploy a single compute instance
resources:
my_instance:
type: OS::Nova::Server
properties:
key_name: my_key
image: F18-x86_64-cfntools
flavor: m1.small
```

2.1.9 FIWARE

FIWARE (fiware.org), a royalty-free and public Cloud platform, is one of the enabling technologies promoted by the European Commission (EC) in 2014 by means of 16 industrial accelerators according to the Future Internet Accelerator Programme. This provides a powerful set of Application Program Interfaces (APIs) in order to simplify the development of smart applications in several application fields. Specifically, it aims to yield an open standard platform and a sustainable, global ecosystem. To this end, the architecture includes a set of general-purpose platform functions called Generic Enablers (GEs). For each GE, a GE Reference Implementation (GEri) has been defined by the FIWARE community. The advantage of using FIWARE is that software architects can rely on a consolidated set of open source general-purpose platform functions that are supported by a world-wide community. In fact, there are GEs for many specific needs that allow developers to apply agile development strategies based on the modern IoT as a Service (IoTaaS) paradigm [18]. In Figure 2.7 is shown the cross-correlation between FIWARE GEs and solutions in smart city applications.

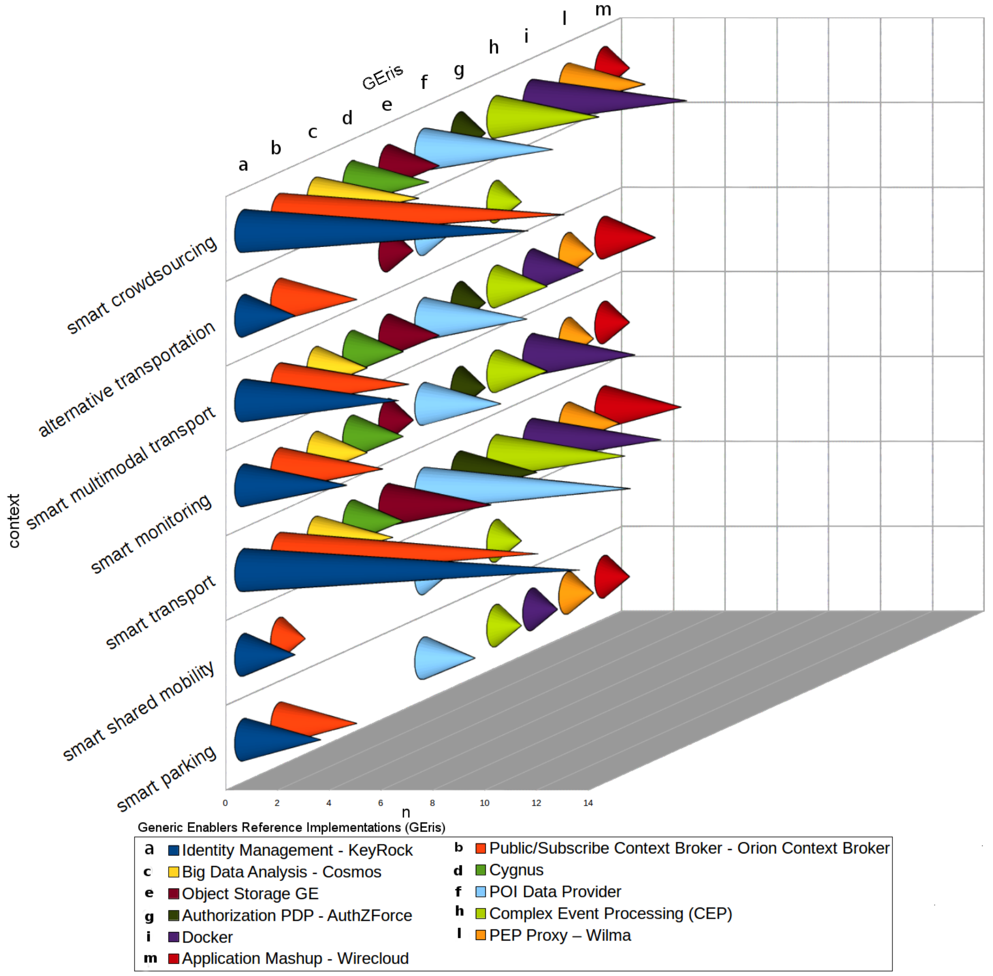


Figure 2.7: Cross-correlation between FIWARE GEs and solutions in smart city applications.

Edge Computing

Edge Computind, according to Gartner [19] and NIST [20], is a computation paradigm that allow the processing and the storage of data closer to end users. The aim of Edge Computing is to reduce response time and use less bandwidth. Edge computing devices, usually, present limited hardware capabilities therefore they cannot process a huge amount of data or run complex programs.

In order to solve such a issue, in last years new software architecture based on microservices and containers have been proposed.

2.1.10 Microservices

A Microservice, according to Gartner [21] and NIST [22], is the basic element that results from the decomposition of a monolithic application. It is able to accomplish specific tasks and interacts with other microservices by using standard APIs.

Microservices architectures present the following advantages [23]:

1. **Ready for market faster.** Because the development of small part of code is faster and simpler;

2. **Highly scalable.** Adding or removing microservices is very fast and simpler;

3. **Resilient.** If one, or more, microservices fail the whole application continue to work. Only functionalities depending on these microservices will be unavailable.

4. **Easy to deploy.** Because each module can be deployed independently from others.

5. **Accessible.** Maintenance and updates can be done only on specific microservices.

6. **More Open.** The microservices approach allows the combination of two or more different technologies.

The enabling technology for microservices is container. That is, usually, each container encapsulates a microservice.

CHAPTER 3

Container Virtualization for Concurrent Microservices in Edge-of-Things Computing

The Internet of Things (IoT) Cloud concept is emerging as innovative kind of distributed system including a set of Single Board Computers (SBCs), smart phones, smart gateways and other Iot devices interconnected by means of the public Internet to a Cloud system. They offer IoT as a Service (IoTaaS) consisting of one or more micro-services. Typically, Edge-of-Things (EoT) computing allows to move services from the Cloud to these IoT devices in real-time. The container-based virtualization is a lightweight solution that can be adopted in IoT devices, for enhancing service provisioning, setup, and management of micro-services. In this chapter we analyse the overhead introduced by container virtualization when multiple concurrent micro-services are executed in parallel within the same IoT device in order to optimize both virtual sensing and actuating resources. Experiments proves that the introduced overhead is acceptable considering the obvious advantages brought by the adoption of container virtualization in terms of resources partitioning.

3.1 Introduction

Currently, in the Internte of Things (IoT) scenario, Gartner has estibated roughly 6.4 billion of embedded devices connected over the Internet, the International Data Corporation has estimated roughly 9 billion of embedded devices (excluding smartphones, tablets, and computers), whereas IHS has estimated roughly 17.6 billion of IoT devices (all types of IoT devices included) and this number is destined to dramatically grow up by 2020 [24].

In this scenario, the synergy among IoT and Cloud Computing solutions allows IT operators to offer their services in an innovative way. It represent chance to increase vendors business. These new business perspectives are bringing the rising of IoT Clouds. We define an IoT Cloud a private, public, or hybrid Cloud provider that integrates its system with distributed Single Board Computers (SBCs), smart phones, smart gateways and other Iot devices, in order to delivery besides traditional Cloud Services (IaaS, PaaS, SaaS), an innovative kind of hybrid service level, named *IoT as a Service* (IoTaaS) [25]. In this scientific work we consider IoT and SBC devices as synonyms. An IoTaaS can be a service deployed in one or more IoT devices in a distributed fashion. The Cloud computing introduced new concepts and paradigms, among them, resource virtualization is the most important. One of major resource virtualization technique is the HyperVisor Virtualization (HVV), that needs software module named virtual machine monitor (VMM) able to provide the abstraction of VMs. However, since IoT devices have limited hardware resources, the hypervisor virtualization is not so applicable because it raises performance concerns. Recently, was introduced the Operating System virtualization technique, also named Linux Container Virtualization (LCV), that consists into the split of the physical resources in order to create multiple isolated instances. This virtualization technique represent a lightweight alternative to HyperVisor Virtualization. Thanks to recent technological advancements, apart from the hypervisor virtualisation more and more providers are looking at the container-based virtualization to manage their datacenters. Moreover, on the contrary of hypervisor virtualization, container-based virtualization better suits IoT devices in terms of performance. In fact, it was demonstrated that the overall overhead added by container virtualization in IoT devices such as Raspberry Pi boards is negligible [26]. LCV is revolutionizing IoT Clouds [27], indeed in terms of resource management it is bringing more advantages than the HVV, LCV allows to manage efficiently virtual IoT capabilities into IoT devices.Currently, for the best of our knowledge, other scientific works analysisng concurrent micro-services running in containers deployed on the same IoT device have not been proposed yet. The objective of this tutorial chapter is to fulfil such a gap in literature. In particular, considering the advantages in adopting virtualization techniques in an hybrid IoT Cloud scenario in terms of IoTaaS provisioning, setup, and management, we specifically discuss how to setup concurrent micro-services running in containers deployed in the same IoT device. In order to achieve this goal, we specifically discuss a hardware/software configuration including a Raspberry Pi B+, i.e., one of the major IoT devices currently available in the market, and the Docker container engine. Moreover, in order to investigate the performance of concurrent micro-services running in IoT devices, we consider a smart mall scenario in which each Raspberry Pi B+ device runs a Docker container engine, that executes, in turn, different containers, each one running a specific COaP server providing a particular service to users.

The reminder of this work is structured as following. In the next Section, we analyse the related works. In Section 3.3 we present the LCV, whereas the applicability of such a technology in IoT Cloud scenarios is motivated in Section 3.4. In Section 3.5, we discuss how to LCV on Raspberry Pi devices [28] and the Docker container engine. In Section 3.6, we analyze performances of the system, in particular, the overhead introduced by the container virtualization considering several concurrent micro-services deployed on different containers on the same IoT device. Conclusions are provided in Section 3.7.

3.2 Related Work

The interest of both academic and industrial communities on container virtualization for IoT scenarios is proved by the increasing number of initiatives that are appearing in literature. A performance analysis, highlighting advantages and disadvantages of container virtualization technologies on IoT devices is presented in [29]. Moreover, in [26], a study performed on Raspberry Pi devices, demonstrates that the overall overhead added by containers is negligible. The advantages of using the containter virtualization technology in both independent and federated IoT Cloud scenarios are respectively discussed in [27] and [18]. A container-based edge Cloud PaaS architecture based on Raspberry Pi clusters addressing edge cloud requirements such as cost-efficiency, low power consumption, and robustness is discussed in [30]. An approach that allows to dynamically allocate container-based resource offered by IoT and Edge computing devices is discussed in [31]. The design and implementation of a container-based virtual client architecture for interactive digital signage systems is discussed in [32], by proposing a piece of middleware able to manage virtual digital signage clients and IoT devices, to reduce the server workload and to simplify the service deployment. The deployment requirements of IoT gateways by means of container virtualization is discussed in [33]. In [34], CIoud4IoT, a platform able to perform horizontal (roaming) and vertical (offloading) migration of IoT functions is presented. Virtualizatoin functionalities are implemented by means of the Kubernetes container engine. CIoud4IoT is organized organised in three tiers: Cloud, Edge and IoT gateways. Another similar lightweight Edge Gateway-as-a-Service for IoT is discussed in [35] A piece of framework for Cloud computing able to manage both real-time IoT data and scientific non-IoT data is presented in [36]. In order to test Cloud services, an experimental study is performed considering the Docker containers for providing Software as a Service (SaaS) in a hybrid cloud environment. Authors in [37] provide an efficient auto-scaling scheme (EAS) that by means of the use of the GPU of numerous distributed desktop PCs it is able to provide a very high QoS compared to the CPU thanks to the adpoption of numerous parallel threads for processing the information. In [38] authors, by means of the network virtualization and Openflow technologies provided a Distributed SDN testbed for IoT devices. In [39], authors present an IoT virtualization framework. It is able to support connected object such as sensors, and to expose them by means of web services. In order to addess the issue of connectivity, it uses several adapter specific per each sensor. Furthermore authors considered advanced semantic access policies. In spite of more and more scientific works are appearing in literature focusing on virtualization in IoT, a study on concurrent micro-services running in different containers deployed on the same IoT device is missing. The objective of this chapter is to fulfil such a gap.

3.3 Overview on Container based Virtualization

Virtualization is on the basis of Cloud computing because it allows to virtualize computing, storage and networking devices. Virtualization, by means of a specific software layer named hypervisor is able to provide several isolated execution environments named Virtual Machines (VM). Examples of HyperVisor Virtualization software solutions are KVM, Xen, WM-Ware, Virtual Box, Virtual PC.

Recently, a lightweight alternative to HVV is the Linux Container Virtualization (LCV). Such a technology partitions the resources of physical computer in multiple isolated user-space instances [40]. Example of LCV software solutions includes include Docker, Kubernates, OpenVZ, Virtuozzo, etc.

The main difference between HVV and LCV is that HVV provides abstraction for guest OSs, instead LCV share the same single OS kernel among containers[40]. In simple words, HVV creates a hardware abstraction level, whereas LCV works by using system calls. From users point of view there is no difference between containers and VMs. Moreover, since LCV uses less hardware capabilities than HVV.

Containers can be used in two different modes:

1. **Application Container**: Each container is specialized in order to run a specific application;

2. **System Container**: an user space runs into a container.

LCV has a more flexible setup, configuration, optimization and management of their sensing and actuating capabilities.

A comparison among HVV and LCV is provided below.

- **Start-Up-Time**. LCV is faster than HVV for the start-up phase;
- **Dynamic-Runtime Control**. LCV allows to start or kill an application running within a container from the host OS, whereas HVV typically requires virtual network/serial connections from the host OS.
- **Speed**. Only tests in a specific scenario can asses the latency and throughput overhead.
- **Isolation**. HVV provides a strong isolation contrarily to LCV.
- **Flash Memory Consumption**. LCV allows sharing both OS kernel and other parts of the user space, whereas since HVV requires isolated VM images no sharing is possible.
- **Dynamic Resource Assignment or Separation**. Both solutions support such a feature.
- **Direct Hardware Access**. LCV in opposite to HVV requires specific drivers in the host OS kernel in order to access to HW peripherals.

From this analysis we highlight that LCV wins against HVV except for two aspects, as described above.

3.4 Management of Resource in IoT-Cloud

IoT Cloud providers are able to provide their services using both the HVV and LCV paradigms. IoT devices, that executes customized software, are connected to a Cloud platform able to manage heterogeneous sensing data coming from several sensors. Commonly, IoT device, due to its limited computation capacity, runs minimal applications specific for sensing and actuating. Instead, Cloud datacenter run massive computation tasks such as storage and processing of acquisition data.

The described approach is the more frequent, but it is not the best, indeed it is not able to scale in according to user and system requirements. As discussed in Section 3.2, several initiatives have considered applications deployed through LCV in IoT devices, in order to take the advantages of both HVV and LCV. In Figure 3.1 is shown an example of IoT Cloud exploiting both HVV and LCV technologies. The IoT-Cloud system is responsible for:

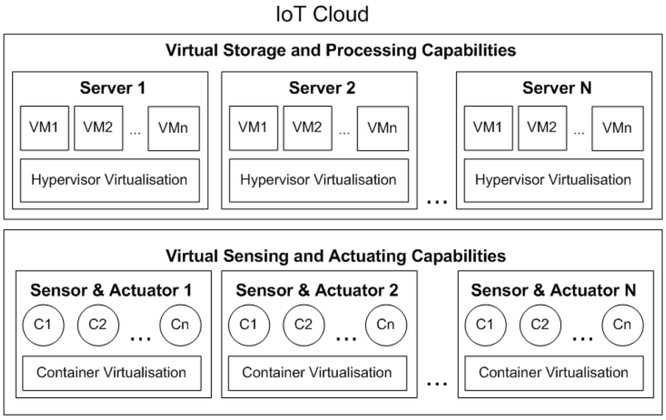


Figure 3.1: Example of IoT Cloud exploiting both both HVV and LCV virtualization technologies.

1. instantiation;

2. guaranteeing Quality of Service (QoS), reliability, Security and scalability;

3. management and optimization of IoTaaS.

Moreover, due to LCV capabilities applied in IoT devices, it is possible to deploy IoTaaS in several distributed containers.

Acquisition data, coming from IoT devices are uniformed and collected into the Cloud platform that is able to trigger actuators connected to IoT devices. HVV and LCV introducing an abstraction layer, hide all the physical capabilities. Usually IoT-Cloud providers adopt both paradigms, but in IoT devices is used only LCV. IoT Clouds operators in order to provide dynamically services to their clients deploy in their infrastructure several container and VMs. It allows operator to relocate, optimize, and arrange its own virtual resources. Using this infrastructure, IoT-Cloud operator is able to satisfy any service allocation request coming from its clients.

Thanks to LCV applied to SBCs, assuming that each micro-service runs inside a dedicated container, IoT Clouds can perform different tasks including:

- **Deployment of Distributed IoTaaS**. It is possible to arrange distributed IoTaaS integrating containers deployed on different IoT devices;

- **IoTaaS Relocation and Optimization**. In order to enforce load balancing strategies, an IoT Cloud can migrate a container from an IoT device to another;

- **IoTaaS Consolidation**. Considering location-aware IoTaaS built combining the containers installed in different IoT devices it is possible to enforce software consolidation strategies according to a particular application logic;

This brings new business opportunities for IoT Cloud providers in terms of cost-effective assets relocation and optimization, power saving, and on-demand resources provisioning.

3.5 How to Setup LVC in Linux-Based SBC

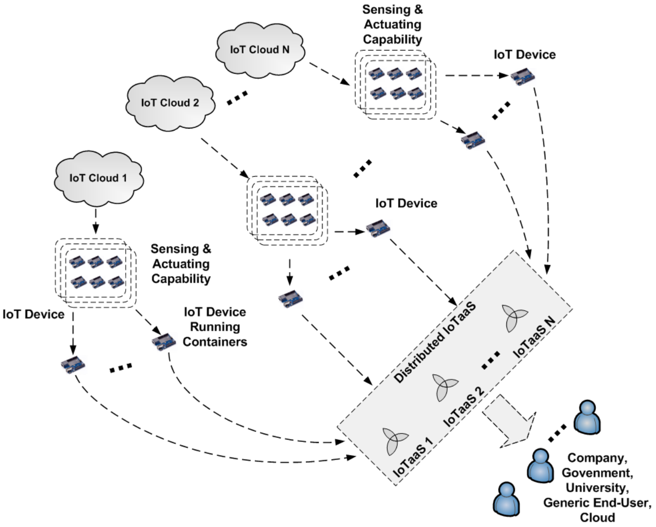

Sensing & Actuating Capability
IoT Cloud N
IoT Device
IoT Cloud 2
IoT Cloud 1
IoT Device
Sensing & Actuating Capability
IoT Device
IoT Device Running Containers
Distributed IoTaaS
IoTaaS N
IoTaaS 2
IoTaaS 1
Company, Govenment, University, Generic End-User, Cloud

Figure 3.2: Exaple of distributed IoTaaS scenario enabled by container virtualization. Several IoTaaS instances includes different micro-services running in containers deployed in IoT devices belonging to different IoT Clouds.

In this Section, we discuss how adopt the LCV in Linux-based SBCs. In particular, we focus on an SNC architecture based on a Raspberry Pi B+ model running the Docker container engine. In LCV, the virtualization layer acts as an application on top of the OS. According to this approach, the host OS kernel can run several isolated guest virtual environments that called containers. Typically, the Linux OS kernel works with a suitable virtualization layer that allows to directly run virtual execution environments offering near native performances and eliminating the overhead introduced by the hypervisor mediation. In such a context, the container engine is a piece of software responsible to automate the instantiation, management, and execution of any application.Hereby, in this work we consider Linux-based SBCs. IoT devices, communicate with external actuators and sensors by means of GPIO ports. In Figure 3.3 is shown the system architecture.

In our system we adopted the Raspberry Pi B+ model as IoT device. For the aim of this work it presents several advantages than a generic computer. the major of these is the presence of the GPIO ports, that allow it to manage actuators and sensors connected. We remark that as external sensor or actuator we can attach button, relays, LED, motors, switches, temperature sensor, proximity etc. [41]. Usually the Raspberry is typically equipped with a

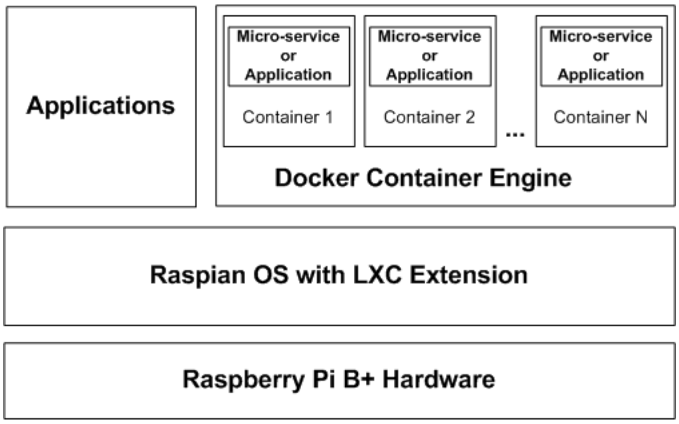

Applications
Micro-service or Application
Container 1
Micro-service or Application
Container 2
...
Micro-service or Application
Container N
Docker Container Engine
Raspian OS with LXC Extension
Raspberry Pi B+ Hardware

Figure 3.3: Software architecture for container-based virtualization based on Docker deployed on a Raspberry Pi B+ board.

Linux OS. In our work we adopted Raspbian, it, by means of LXC modules allows to run several container in a single SBC [42]. As shown in Figure 3.3, applications are deployed on the top of Raspbian OS.

In this work, as enabling technology for the deploy of containers we adopted Docker. An important feature of Docker is the presence of Docker Hub, that allow users to share applications and automating workflows. Considering Figure 3.3 micro-services can be deployed inside different containers managed by Docker. This allows IoT-Cloud operators to deliver and run the same microservices on different IoT devices quickly. Container virtualization in IoT enables users to deploy a distributed IoTaaS on several IoT Cloud providers as well as in traditional Infrastructure as a Service (IaaS) Cloud providers [43]. Figure 3.2 shows a sample scenario including several distributed IoTaaS instances composed of different micro-services running in containers and deployed on IoT Cloud A, B and N.

3.6 Performance Analysis

In this Section, we analyze the overhead of concurrent containers running in an IoT device considering a real Hardware/Software configuration including a Raspberry Pi B+ device in which runs Docker. In particular, our objective is not to analyse the behaviour “behind the scenes” of Docker when parallel containers are executed, but to test the behaviour of the system in a real a well-known IoT scenario. We remark, that currently, for the best of our knowledge, other scientific works analysing concurrent micro-services running within containers deployed on the same IoT device do not exists, and since this is the first scientific work focusing on such an aspect, it was not possible to compare it with other similar existing solutions. In particular, we considered the mean response times of txThings, a CoAP server

Table 3.1: Response time of a CoAP server directly deployed on the Raspian OS (No Container Configuration) vs. 1 CoAP server deployed in a Docker container (1 Container Configuration).

	No Container			Container		
Req.	**Avg [ms]**	**STDev**	**Conf95%**	**Avg [ms]**	**STDev**	**Conf95%**
1	27,900	3,033	1,085	49,500	5,495	1,966
2	36,200	10,672	3,819	71,300	20,876	7,470
3	42,600	16,260	5,818	93,367	35,108	12,563
4	54,167	21,469	7,682	132,467	39,448	14,116
5	71,567	25,168	9,006	169,500	42,318	15,143
6	89,867	28,299	10,126	210,933	40,075	14,340
7	108,767	32,531	11,641	253,933	45,631	16,329
8	131,700	34,752	12,436	300,400	47,818	17,111
9	153,467	37,095	13,274	348,833	51,811	18,540
10	176,167	38,226	13,679	396,600	52,926	18,939
11	197,467	38,066	13,621	447,467	54,089	19,355
12	221,267	39,674	14,197	497,867	53,843	19,267

Table 3.2: Response time of 2 CoaP servers respectively deployed on 2 different Docker containers (2 Concurrent Containers) receiving client'requests in parallel.

	Container A			Container B		
Req.	**Avg [ms]**	**STDev**	**Conf 95%**	**Avg [ms]**	**STDev**	**Conf 95%**
2	52,133	14,943	5,347	84,733	19,375	6,933
4	98,900	41,081	14,700	149,600	47,147	16,871
6	164,400	54,298	19,430	225,367	67,290	24,079
8	248,867	56,509	20,221	316,967	69,724	24,950
10	333,833	63,670	22,784	400,200	68,013	24,338
12	419,800	61,874	22,141	460,767	64,997	23,258

Table 3.3: Response time of 3 CoaP servers respectively deployed on three different Docker containers (3 Concurrent Container configuration) receiving clients' requests in parallel.

	Container A			Container B			Container C		
Req.	**Avg [ms]**	**STDev**	**Conf 95%**	**Avg [ms]**	**STDev**	**Conf 95%**	**Avg [ms]**	**STDev**	**Conf 95%**
3	70,567	39,618	14,177	108,233	32,848	11,754	123,467	30,669	10,975
6	165,467	78,534	28,103	219,900	70,919	25,377	247,367	48,891	17,495
9	276,833	99,144	35,478	349,267	90,481	32,378	377,100	63,071	22,569
12	399,800	107,513	38,472	453,600	87,155	31,187	459,733	54,597	19,537

implementations, acting as micro-services, running Docker's Container. Our testbed consists of two separate Raspberry Pi B+ devices with the following hw/sw configuration:

- CPU: Broadcom BCM2835 @ 700 MHz;
- RAM: 512 MiB;
- Storage: 16 GB SDHC card;

- OS: Raspbian Linux OS - 3.18.8 kernel version.

One of two devices, the server, runs several container at the same time. For instance, considering a scenario in which users want to manage efficiently a small mall by means of COAP several servers run on the same IoT device at the same time. In particular, we consider a server for managing efficiently energy consumption of light appliances, indeed them can turned on/off when customers enter or leave a room; a second server could send advertisement to customers that pass close to a shop; finally another CoAP server could provide generic information. Each virtual environment runs a specialized CoAP servers which listen on specific UDP port. The second device acts as CoAP client. Several CoAP clients send requested to the same CoAP server that send back ACK messages as reply. Client and Server are interconnected by means of a gigabit network. In our test, we calculated the time needed to manage a request from the client. In particular, we calculate this time as the difference between the instant when the client receives the ACK message and the instant when it sends a request. In order to start the CoAP server into a container we used the command shown in the Listings 3.1.

```
docker run −t −i ——net="host" −p 7777:7777
shop/coapserver python serverm.py 7777
```

Listing 3.1: Command to start a Docker container with a CoAP server.

In order to have reliable time performances we executed experiment 30 subsequent times and considered mean values. In particular, we realized 4 different configurations:

- **No Container**. In this configuration, we ran a single CoAP server on the OS;
- **1 Container**. In this configuration, we ran a single CoAP server on a container;
- **2 Concurrent Containers**. In this configuration, we ran a two different CoAP servers on two different container named for simplicity A and B;
- **3 Concurrent Containers**. In this configuration, we ran a three different CoAP servers on three different container named for simplicity A, B and C;

In the all presented configuration, we sent from 1 to 12 presence requests. In particular, considering the "No Container" and "1 Container" configurations, we respectively sent 12 COaP requests each one. As far as "2 Concurrent Containers" configuration, we sent 6 COaP requests respectively to container A and B in parallel. In the end, considering the "3

Concurrent Containers" configuration, we sent 4 COaP requests respectively to container A, B, and C in parallel.

Table 3.1 shows the average, standard deviation, and 95% confidence interval values obtained considering the response times of the CoAP server according to No Container and 1 Container configurations. In this regard, the chart of Figure 3.4 compares the average response time of the CoAP server according to No Container and 1 Container configurations, highlighting the constant overhead introduced by the Docker container virtualization. For 1 request sent to the CoAP server directly running on Raspian OS (No Container) we observe roughly 27,900ms response time, whereas considering 1 request sent the CoAP server running on the Docker container we got 49,500ms response time with a gap of roughly 21,600 ms. The response time increases quite constantly, observing for 12 requests roughly 221,266 ms in the No Container configuration against roughly 497,866 ms in the 1 Container configuration. Figure 3.5 shows the overhead introduced by 1 Container configuration with respect to No Container Configuration. Specifically, such an overheard also includes the latency introduced by a single physical network interface that acts as a bottleneck for network traffic. For 1 request we observe an overhead of 21,6msec. The overhead increases quite constantly, observing

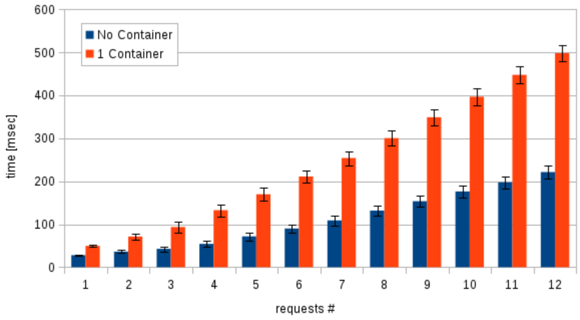


Figure 3.4: Response time of a CoAP server directly deployed on the Raspian OS (No Container configuration) v.s. a CoAP server deployed on a Docker container (1 Container configuration).

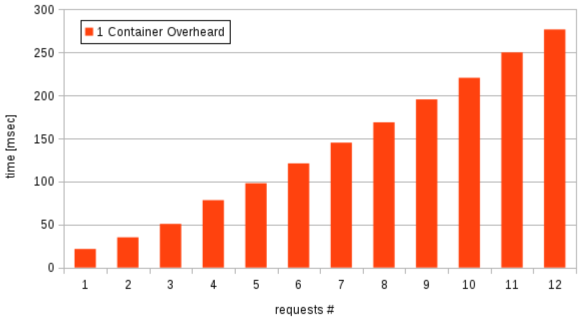


Figure 3.5: 1 Container configuration overheard.

276,6 ms for 12 requests. We can conclude that the overhead of LCV is minimum with respect to the processing time of CoAP servers. This highlights that the overhead introduced by 1 Docker container engine is acceptable considering Raspberry Pi B+ devices.

Table 3.2 shows the average, standard deviation, and 95% confidence interval values obtained considering the response times obtained by two concurrent CoAP servers respectively deployed in Docker container A and B (2 Concurrent Containers configuration). In particular, the system is able to satisfy 2, 4, 6, 8, 10, and 12 client's requests in parallel. More specifically considering 2 client's requests, one is managed by the CoAP server on Container A and one by the CoAP server on Container B, and so on. Figure 3.6 shows a histogram comparing the COaP response times considering No Container, 1 Container and 2 Concurrent Containers configurations. Considering the latter, since the two CoAP servers run in parallel con Container A and B, the response time is given by the slowest CoAP server. In particular, the response time considering the two configurations is very similar and from 10 requests on (5 to Container A and 5 to Container B), the time to satisfy all the 10 requests, that is roughly 400,200 ms is slightly less than the one obtained in 1 Container that is roughly 396,600 msec proving the advantages of the parallel processing. Figure 3.7 shows the overhead introduced respectively by 1 container and 2 Concurrent Containers configurations with respect to the No Container configuration. Specifically, such an overheard also includes the latency introduced by a single physical network interface that acts as a bottleneck for network traffic. For 2 requests, we observe an overhead values in terms of response time of roughly 35,1 msec and 48,533 msec respectively for 1 Container and 2 Concurrent Containers configurations. The overhead increases quite constantly, and for 12 requests we surprisingly observe that it is even less in the 3 Concurrent Container configuration with roughly 239,5 msec compared with 276,6 msec obtained in the 1 Container configuration.

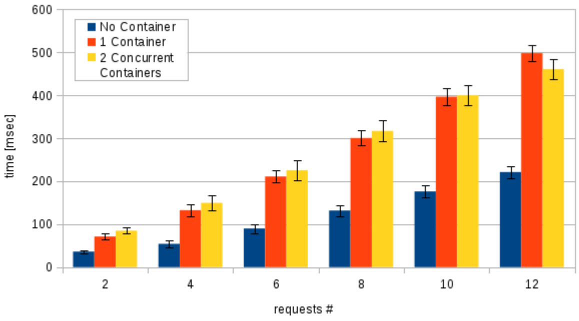


Figure 3.6: Response time of a CoAP server directly deployed on the Raspian OS (No Container configuration) v.s. a CoAP server deployed on a Docker container (1 Container configuration) v.s. 2 CoAP servers deployed on two Docker containers (2 Concurrent Containers configuration.

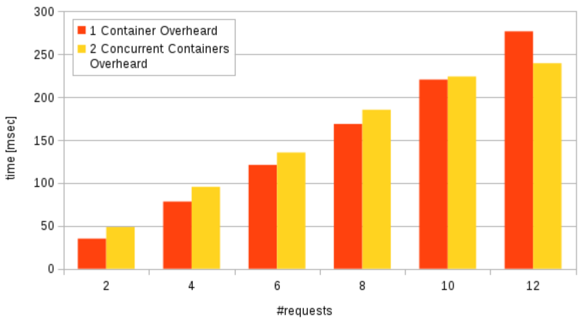


Figure 3.7: 1 Container configuration overhead v.s. 2 Concurrent Containers configuration overheard.

Table 3.3 shows the average, standard deviation, and 95% confidence interval values obtained considering the response times obtained by the CoAP server deployed on a Docker container (1 Container) by the CoAP servers running on three different Docker containers and satisfying client's requests in parallel (3 Concurrent Containers configuration). In particular, the system according to 3 Concurrent Containers is able to satisfy 3, 6, 9 and 12 client's requests in parallel. More specifically considering 3 client's requests, 1 is managed by the CoAP server on Container A, 1 by the CoAP server on Container B, and 1 by the CoAP server onContainer C and so on. We can observe a similar trend comparing 1 Container with

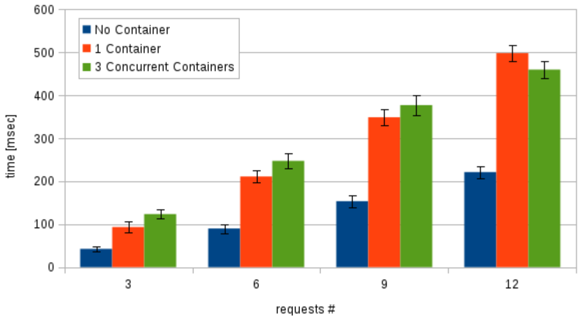


Figure 3.8: Response time of a CoAP server directly deployed on the Raspian OS ("No Container configuration") v.s. a CoAP server deployed on a Docker container ("1 Container" configuration) v.s. 3 CoAP servers deployed on three Docker containers ("3 Containers" configuration.

3 Concurrent Containers as shown the histogram depicted in Figure 3.8, considering 3, 6, 9, and 12 requests. Here, in 2 Concurrent Containers we consider 3 parallel CoAP servers respectively running on Containers A, B, and C and the response time is given by the slowest one. Even here we can observe a similar trend, and for 12 requests, 2 Concurrent Containers (taking 459,733 ms) is slightly faster than 1 Container (taking 497,867 ms) including a single CoAP server running in a Container. Figure 3.9 shows the overhead introduced respectively by 1 container and 3 Concurrent Containers configurations with respect to the No Container

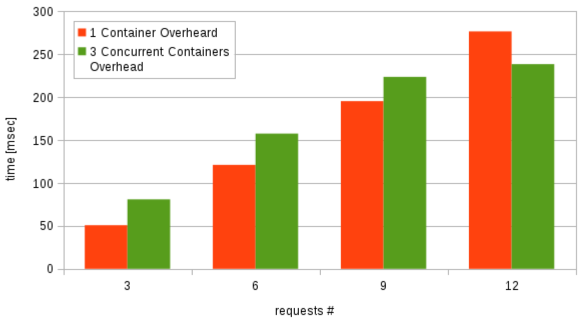


Figure 3.9: 1 Container configuration overhead v.s. 3 Concurrent Containers configuration overheard.

configuration. Specifically, such an overheard also includes the latency introduced by a single physical network interface that acts as a bottleneck for network traffic. For 3 requests, we observe an overhead values in terms of response time of roughly 50,76 msec and 80,86 msec respectively for 1 Container and 3 Concurrent Containers configurations. The overhead increases quite constantly, and for 12 requests, also here, we surprisingly observe that it is even less in the 3 Concurrent Container configuration with roughly 236,46 msec compared with 276,6 msec obtained in the 1 Container configuration.

Instead, considering 4 concurrent containers, we notice a drastic performance degradation that makes the Raspberry Pi B+ not usable.

In the end, the histogram depicted in Figure 3.10 summarizes the response times of No Container, 1 Container, 2 Concurrent containers, and 3 Concurrent containers configurations considering 12 requests. We can observe that increasing the number of concurrent Containers there is not a performance degradation. On the contrary, how it is shown in Figure 3.11,

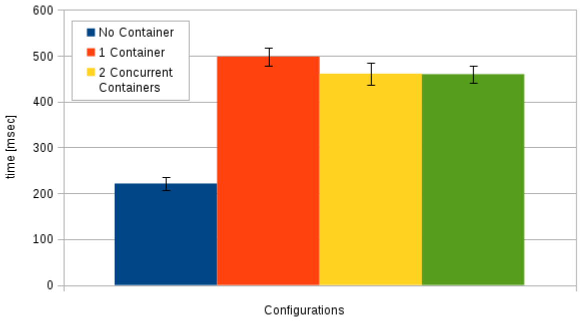


Figure 3.10: Comparison among 1 Container, 2 Concurrent Containers and 3 Concurrent Containers configurations response times considering 12 COaP requestes.

increasing the number of containers we improve the overhead, in fact, for 1 Container, 2 Concurrent Containers and 3 Concurrent containers it is respectively 276,6 msec, 239,6 and 238,46 msec.

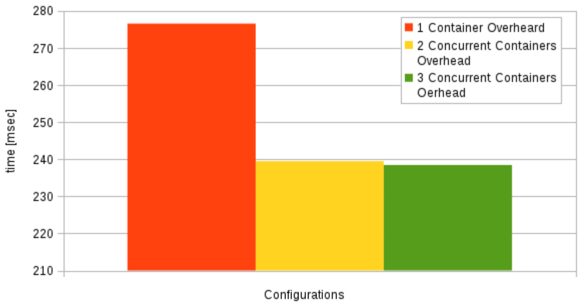


Figure 3.11: Comparison among 1 Container, 2 Concurrent Containers and 3 Concurrent Containers configurations overheads considering 12 COaP requests.

From such experiments, we can deduct that by increasing the number of concurrent containers until 3 ones, we roughly maintain the same performance, but we can have the advantage to manage in a more flexible fashion the deployment of micro-services within Raspberry Pi B+ IoT devices. Apart from the advantage in managing micro-services inside the same IoT device, this enables also the easy setup of distributed IoTaaS.

In conclusion, we can state that containers introduce an overhead in all configurations. But comparing the performance of 1 Container with 2 and 3 Concurrent Containers there is an additional acceptable overhead introduced by LCV. This proves that the LCV technology, implemented by means of the Docker container engine is a good solution for the provisioning, setup, and management of complex IoTaaS in future IoT scenarios.

3.7 Conclusion

LCV is a lightweight alternative to HVV that can be used in IoT devices for improving the IoTaaS provisioning, setup, and management. Considering different application scenarios, Linux Container Virtualization allows IoT Cloud providers to deploy, in a flexible fashion, micro-services in SBCs. In this chapter, we studied the overhead by concurrent containers running in the same IoT device. In order to achieve such a goal we specifically considered on a Raspberry Pi B+ device running the Docker container engine managing several concurrent CoAP servers deployed in dedicated contatiners. From our experiments we highlighted increasing the number of concurrent containers that the overhead introduced by LCV is acceptable considering the obvious advantages of its utilization in emerging IoT Cloud scenarios. In future works we plan to investigate other related challenges including service optimization and security.

CHAPTER 4

Distributing microservices among federated tenants.

Nowadays, more and more Cloud providers are appearing on the market. In this context, a typical issue is represented by the management of distributed services deployed on different federated Cloud providers. Assuming that a distributed service consists of several microservices, in this chapter, we specifically focus on the setup of virtual environments and on deployment tasks required in each Federated Cloud providers. In particular, we present BOSS an Orchestration Broker that starting from an ad-hoc OpenStack-based Heat Orchestration Template (HOT) service manifest produces several HOT microservice manifests including deployment instructions for involved Federated Clouds. Therefore, users can formalize advanced deployment constrains in terms of data export control, networking, and disaster recovery. Experiments prove the goodness of the proposed system in terms of performance. Nowadays, Edge computing allows to push the application intelligence at the boundaries of a network in order to get high-performance processing closer to both data sources and end-users.

4.1 Introduction

In this chapter, we describe an innovative solution able to manage more Federated Clouds [44] considering different aspects in the context of Cloud Brokering where Geographic Constrains are considered. Our starting point was the European project called "Cloud for Europe". The aim of this project is the implementation of the European marketplace for the Public Administration. In this scenario, the purpose of this scientific work is to set the virtual environment for the deployment of a distributed service. In such a scenario, we consider that distributed services are composed by several microservices that can be instantiated in different federated datacenters. In order to add data export control, slicing network and disaster avoidance features we, starting from the Heat Orchestration Template, introduced a new template called "Distributed Service Manifest" (DSM). Users by means of restful API can transmit customized DSMs to the Broker OpenStack Service (BOSS). BOSS is the deployed system able to elaborate ad-hoc service manifest documents including application deployment instructions destined to selected federated Cloud providers and users' requirements. BOSS analyses the DSM and automatically extracts the Microservice Manifests (MM) compliant with Openstack Heat. An important feature of BOSS is the ability to select target federated datacenter in according to constrains provided by users inside the DSM. Experiments conducted in a real scenario, prove that our approach is able to deploy distributed services in federated Cloud environment. The rest of the chapter is organized as follows: Section 4.2 describes related works. Motivation of this work are discussed in Section 4.3. In Section 4.4, starting from the Heat Orchestration Template we present the Distributed Service Manifest. Overview and basic assumptions are described in Section 4.5. The workflow of the system is presented in section 4.6. Experiments, are discussed in Section 4.7. Finally, conclusions and lights on the future are summarized in Section 4.8.

4.2 Related Work

Cloud federation is a very hot topic in scientific community, it raises many challenges in several research fields as described in [45], [46], [47], [18], [48] and [49]. Most of the works in the field focus the study of architectural models that are able to efficiently support the collaboration among different cloud providers focusing on various aspects of the federation. However the new trend is to adopt the Federation for new interesting scenarios: IoT, Edge, Cloud and Osmotic Computing [50].

In [51], authors propose a component-oriented method able to perform automated provisioning of Cloud business applications that can manage the overall application's lifecycle. Such a method is based on Cloud orchestration tools that manage the deployment and dependencies of provided components. Specifically, a Linked Unified Service Description Language (LUSDL) to describe services for matching user's requirements is proposed. Designing and deploy an Edge-Cloud network implies first to determine where to install facilities among available sites, then assigning sets of access points, while supporting VM orchestration and considering partial user mobility information, as well as the satisfaction of service-level agreements. With this regards, authors in [52] proposed a link-path formulations supported by heuristics to compute solutions in acceptable time. More in details, authors qualify the advantage to consider the mobility for both VMs and users as up to 20% less users not satisfied in their SLA with a little increase of opened facilities.

In [53] authors propose a new platform Cloud4IoT. Cloud4IoT offers orchestration, automatic deployment, and dynamic configuration of IoT support software components and data-intensive applications for data processing and analytics, thus enabling plug-and-play integration of new sensor objects and dynamic workload scalability. Cloud4IoT introduces the concept of Infrastructure as Code in the IoT context. It empowers IoT operations with the capability of Cloud services. Furthermore it shifts traditionally centralized Cloud architectures towards a more distributed and decentralized computation paradigm, as required by IoT technologies, bridging the gap between Cloud Computing and IoT ecosystems. In [54] a set of ontologies to semantically represent Cloud resources is proposed. In particular, three Cloud resource description standards are considered, that are:

1. Open Cloud Computing Interface (OCCI);

2. Cloud Infrastructure Management Interface (CIMI);

3. Topology and Orchestration Specification for Cloud Applications (TOSCA).

The proposed piece of framework encourage the creation of a common semantic knowledge base of Cloud resources described using the above mentioned standards. In [55], the approach proposed by the authors facilitates the evaluation of different application topologies by enabling Cloud users to easily evaluate and provision different Topology and Orchestration Specification for Cloud Applications (TOSCA) options based on performance and cost metrics. In particular, it is shown the technical feasibility of the approach based on a case study with the WordPress blogging application where various topologies were automatically provisioned and evaluated.

In [56], authors analyze the orchestration of networking services. They propose a new control orchestration protocol able to provides a transport application programming interface solution for joint Cloud/network orchestration, allowing interoperability of heterogeneous control planes to provide provisioning and recovery of quality of service (QoS)-Aware end-To-end services.

In the paper in [57], authors propose the use of a Cloud Service Broker called a Virtual Cloud Bank (VCB). VCB is the automatic provisioning of tenant-centric cloud services. The work is interesting, indeed, the key contributions are:

- a service that analyze model to specify and register several types of Cloud services;
- a robust and automatic provisioning of Cloud services without the need for any Cloud domain experts;
- the conceptualization of a VCB architecture;
- the intermediation of Cloud services that match tenant requirements;
- a tenant able to analyze model to collect and analyze tenant requirements.

Authors in [58] focus their work in the context of orchestration of energy-aware services. They propose a novel energy-aware resource orchestration framework for distributed Cloud infrastructures, in order to manage both network and IT resources like a typical optical backbone. A high-level overview of the system architecture is provided. In their system the key component is the Path Computation Element.

4.3 Motivations

Resource Brokering in federated Cloud is very hot topic in scientific community. It is considered the step over the federation Cloud.

Motivations behind this scientific work come from the European project called "Cloud for Europe" (C4E). The aim of this project is the implementation of the European marketplace for the Public Administration (PA). In C4E we define two kinds of entities:

1. Service Flow Unit (SFU);
2. Service Flow (SF).

SFU is the micro-service that must be deployed (e.g. microservice for OCR). SF represents the chain of SFUs that compose the distributed service (e.g. system to produce and store passports). The purpose of this scientific work is to prepare the virtual environment able to run distributed services.

SF and SFU in the proposed solution are mapped by means of two different manifest templates. Respectively they are the Distributed Service Manifest (DSM) and the Microservice Manifest (MM). In the next section we are going to present proposed manifests.

4.4 The Service Manifest

In the previous section we introduced the concept of service manifests. Here we present proposed solutions. In particular we remark that our starting point was the Heat Orchestration Template (HOT), the Openstack template. In details HOT is composed by an **header row** and four sections:

1. **description**;
2. **parameters**;
3. **resources**;
4. **output**.

Header row is very useful, in fact it contains the version of the template.

In description section usually are present some human readable strings that explain the function of the template.

Parameter section include the parameters that user can pass during the deploy phase in order to customize the manifest. Currently HOT supports as parameters the following types: String, number, comma_delimited_list, JSON and boolean. Each parameter could include the default clause, an item that contains the default value. If users do not provide any value the parameter will assume the default one. In order to add validation rules each parameter may include the constrain block. This block could contains one or more rules. Supported types of parameter constraints are: length, range, modulo, allowed_values, allowed_pattern and custom_constraint.

Resource section must contain at least one item definition, it is the most important section. In fact any input or output parameter has to refer to an element defined in that section.

The output section is used to provide back a result to users. It allows to retrieve whatever resource parameter.

A well formed HOT must contains at least header row and resource section. Description, parameters and output sections are optional.

The Listing4.1 shows an example of HOT.

Listing 4.1: Heat Orchestration Template

```
heat_template_version: 2015-04-30

description: Complete example
```

```
parameters:
  key_name:
    type: string
    label: Key Name
    description: Name of key-pair
  image_id:
    type: string
    label: Image ID
    description: Name of the image
  instance_type:
    type: string
    label: Instance Type
    description: Type of instance (flavor)

resources:
  my_instance:
    type: OS::Nova::Server
  properties:
    key_name: { get_param: key_name }
    image: { get_param: image_id }
    flavor: { get_param: instance_type }

outputs:
  instance_ip:
    description: The IP address of the
      deployed instance
    value: { get_attr:
      [my_instance, first_address] }
```

In order to create both DSM and MM we followed the Openstack guideline. In this way our service manifests are full compliant with HOT. In particular, in order to adopt as resource orchestrator Openstack Heat, MMs present the same structure of HOTs. Instead DSM is an evolution of the previous one. The innovation of our system is the introduction of 3 parameters: **Location**, **Chain** and **Affinity**.

Parameter Location represent the geographic shape where resources can be deployed. It is defined as a String codified using the *ISO 3166-1 Alpha 3*standard. This parameter is very useful for C4E purposes. In fact it adds **data export control** features to the system. The adoption of this parameter make the system robust, in fact it ensures the correct placement of virtual resources whether the border of the geographic shape will change. In order to grant to user a certain degree of freedom in manifest creation we used the "allowed values" constrain in the parameter definition. In particular users to deploy a specific resource can be choose only already present values.

Chain parameter it is used to specify what resources are interconnected each others. In particular in our system we defined the network slice concept. Network Slice (NS) is the virtual LAN, created by means of Open Virtual Network (OVN).In our system we codified it as comma_delimited_list, in this way users may specify more than one NS for each resource.

In order to add **disaster avoidance** features, in our system we introduced the Affinity parameter. It is used to indicate that a resource can be deployed in the same datacenter with other resources of the same DSM or not. We codified it as boolean, in particular False means that the resource must stay alone in the datacenter.

A valid DMS must contains all parameters described above.

4.5 Architectural Design

In this section we starting from an overview about of the contest in which this work is focused present our innovative concepts in terms of three pillars:

1. data export control

2. slicing network

3. disaster avoidance

Figure 4.1 shows an high level overview of the system:

In particular we have BOSS that accepts as input a DSM, splits and stores it into a NoSql database, after that creates several MMs and deploy them in specific Clouds in according to data export control and disaster avoidance parameters.

In our work we consider to have several federated Clouds. Each Cloud provides its services to several users and tenant.

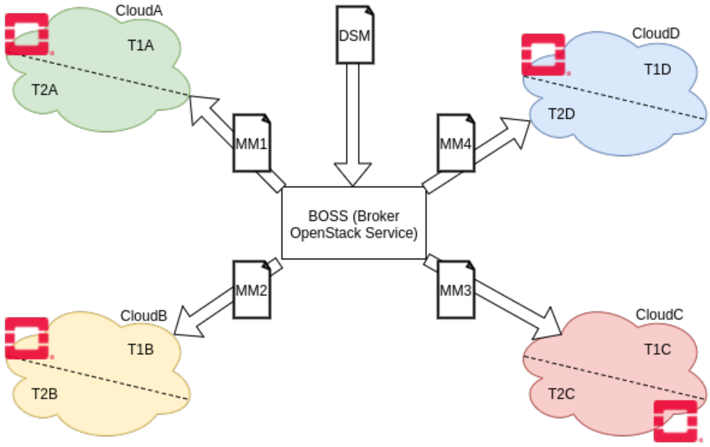


Figure 4.1: High Level Overview

In general, in this work we consider that users from one Cloud are independent from others. In order to create the federation they have to make an a priori agreement with multiple Cloud providers. In this work, for reasons of simplicity, we are considering each Cloud composed by a single datacenter. In a more complex scenario we should consider a Cloud as set of datacenters, but for the purpose of this work it is enough. We remark that in the proposed system each datacenter is mapped as GeoJSON point. GeoJSON become standard in August 2016. It is used for encoding geographic data structures. Currently supported types are: Point, LineString, Polygon, MultiPoint, MultiLineString, and MultiPolygon. Datacenters are located into geographic Countries. They are mapped as GeoJSON Polygon or Multi-Polygon. For compatibility reasons in our system we adopted another standard in order to identify each Country: the ISO 3166-1 Alpha 3.

In our system, we assume that all Clouds are Openstack-based. In particular we adopted Newton version because it is natively compliant with OVN. OVN is very useful for our purposes, in fact, it allows to create network slices. Network slice is very challenging topic, in fact it allow to create isolated virtual networks in which we can run specific services. Each datacenter and in general resources can stay in more than one slice at the same time. Figure 4.2 shows a scenario in wich we have three different network slices: respectively slice blue, violet and green.

As showed in the figure below Clouds 8 and 9 at the same time are into two different slices. Respectively green and violet Cloud8, blue and violet Cloud9. The topology of each slice is full mesh. This means that each Cloud must be connected to all other that belong to the same slice. Considering a slice composed by 2 different Clouds. Each of them must be connected with the remaining one, in this case only one link is enough. In fact we are

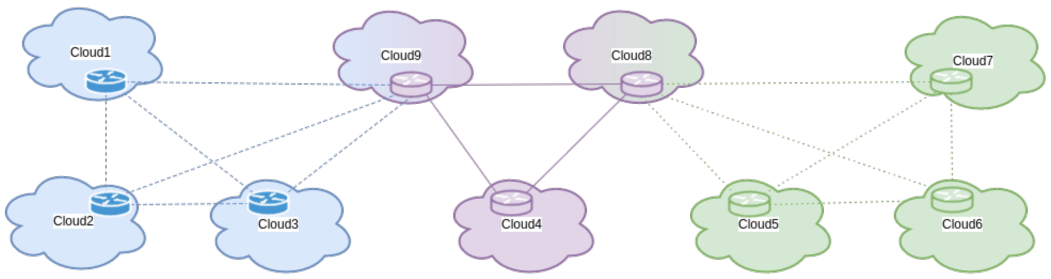


Figure 4.2: Topology with three slices

considering link undirected: if Cloud1 it is connect with Cloud2 then Cloud2 is connected with Cloud1 and vice versa. Now we are considering a slice composed of 3 Clouds, in this case in order to have a full mesh network we need three links: one between Cloud1 and Cloud2, one between Cloud2 and Cloud3 and another one to connect Cloud3 and Cloud1, Figure below shows this scenario.

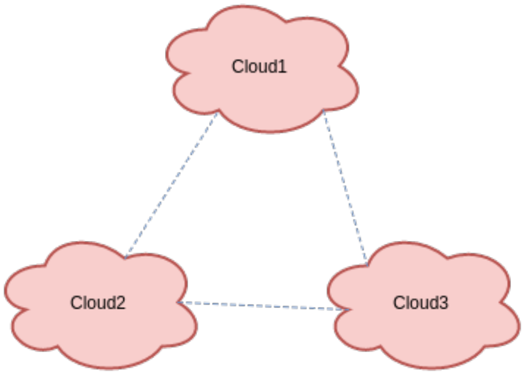


Figure 4.3: Full mesh topology

In general, considering a generic set of N Clouds we define as *Combination C(n,k)* of N elements by k, the total amount of groups that can be created that observe the following constrains:

- each element contains exactly k different elements;
- any two groups must be different at least for one element.

Whit reference to the presented case the number of links necessary to deploy a slice is equal to

l=N*(N-1)/(2!)

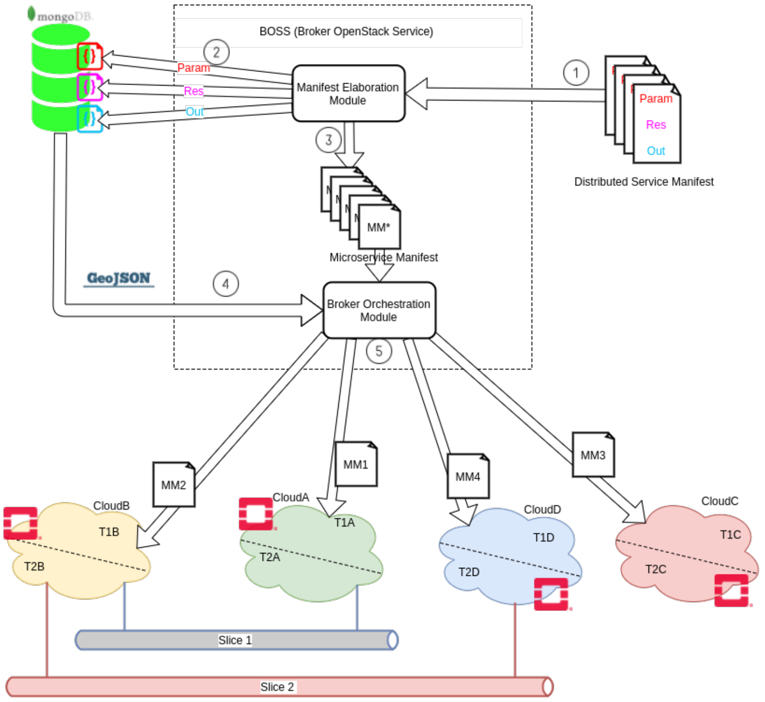


Figure 4.4: Workflow

4.6 BOSS Workflow

In this section we are discussing about of the implementation of the proposed system. In particular, starting from the workflow of the deploy of the DSM, we will explain in details the most important blocks that compose our system.

In our system producer of data are users, consumer are federated Clouds, BOSS acts as interface between them. Users to interact with BOSS can use the provided restful API, BOSS instead uses Cloud's functionalities by means of Openstack4J library.

Figure 4.4 shows the workflow for the deploy of the DSM.

- Step 1: User uploads the DSM

- Step 2: Manifest Elaboration Module (MEM) splits the DSM. In particular, in the example showed DSM is composed by three sections: Param, Res ad Out, respectively red, violet and cyan. The split algorithm, performed by the MEM, divides the DSM and store elements of each section in specific MongoDB collections using a data deduplication algorithm. The data deduplication algorithm is out of the scope of this scientific work.

- Step 3: MEM creates MMs. MicroserviceManifest is an HOT that contains only one resource, related parameters and outputs. MMs are sent with information about of data export control, slice and disaster avoidance to the Broker Orchestration Module (BOM)

- Step 4: BOM per each MM, starting from information about data export control and user's data executes a geo-query on MongoDB in order to retrieve the list of datacenters that can deploy the specific microservice. After that, checking the disaster avoidance attributes BOM selects the suitable datacenter.
- In the last Step (5), BOM by means of Openstack4J library deploy the microservice.

4.7 Performance

In this section we discuss about of the experimental results. In particular we conducted three kinds of analysis: DSM split and store that in our workflow section are depicted as steps 1 and 2, DSM recomposition that in the workflow section it is presented as step 3 and GeoQuery and datacenter selection that in the workflow section they are presented as step 4. Step 5 is out our tests because it depends on external component. We carry out these tests in two different scenarios: "local" in which all machines belong to the same local area network and "remoteVPN" in which machines are interconnected by Virtual Private Network.

In order to deploy our prototype we used two different machines: a blade server that runs a MongoDB stand-alone instance and a workstation that runs BOSS. Blade server is equipped by: CPU Intel(R) Core(TM) Xeon E7-8860V3 CPU @ 3.20GHz, RAM 32GB, OS: Ubuntu server 16.04 LTS 64 BIT. The hardware configuration of the workstation is: CPU Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz, RAM 16GB, OS: Ubuntu server 16.04 LTS 64 BIT. Experiments were repeated 30 times in three different configurations. In particular, in our scenario we consider an increasing number of DSM. Respectively: 10, 100 and 1000.

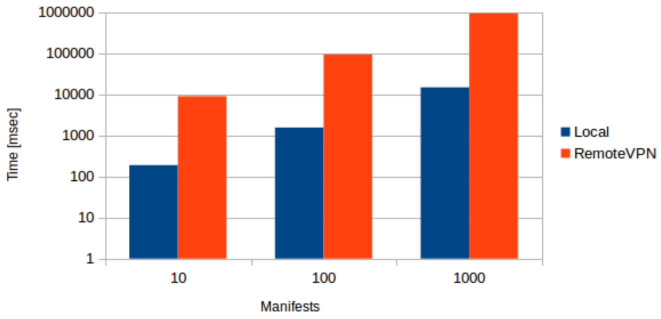


Figure 4.5: DSM split and store

Figure 4.5 shows the behavior of the split and store on MongoDB phases, in the picture the "blue" bar is referred to the local scenario, instead the red one the remoteVPN. As we expect the computation time grows up with the number of DSM to be processed in both scenarios, of course computation time of the local scenario are lower. As showed the trend is linear, and values are acceptable.

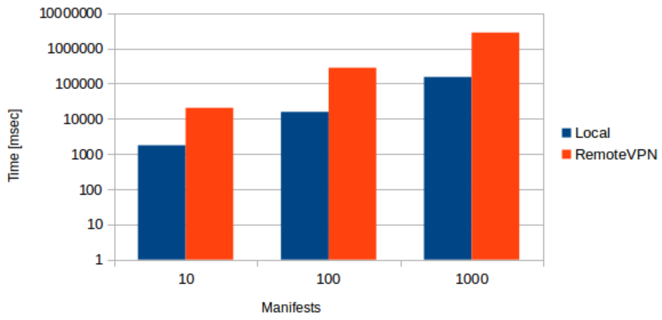


Figure 4.6: DSM recomposition

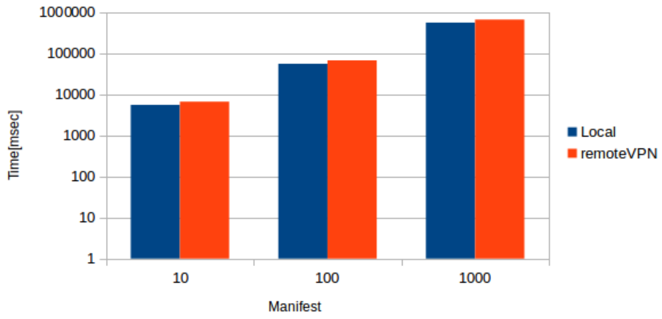


Figure 4.7: GeoQuery and datacenter selection

Figure 4.6 shows the behavior of the recomposition phase. Also in this case the computation time grows up with the number of DSM to be processed. As showed the trend is linear, local scenario is faster than the remote one, computation time is slower than the DSM split phase, but values are still acceptable.

Figure 4.7 shows the behavior of the GeoQuery and datacenter selection phases. Also in this case the computation time increases with the number of manifests to be processed. As depicted the trend is linear and local scenario is more powerful than the remote one.

4.8 Conclusion and Future Work

In this chapter, considering the European Cloud For Europe project we deployed a brokering system able to setup the virtual environment for the deployment of microservices.

The approach described is conducted by means of the Distributed Service Manifest. DSM can be considered as a step over the Heat Orchestration Template. In fact, following the guideline provided by Openstack we introduced three parameters that adding functionalities in terms of data export control, slicing network and disaster avoidance. In this work we also introduced BOSS, our Broker module. This scientific work is the first initiative adopting a Distributed Service Manifest for distributed deployment for this reason we did not find any solution to compare performance of our system. However experiments showed that our system response time present a linear trend. The execution time grows up with the increasing number of considered DSM. In future work we plan to switch from the Cloud Computing to the Osmotic Computing.

CHAPTER 5

Enabling the Secure Management of Hybrid Cloud-Edge Environments

Nowadays, with the advent of Cloud-assisted Internet of Things or Cloud-of-Things (CoT) paradigm, new needs such as the real-time communication appeared, but traditional systems, based on the Cloud Computing paradigm do not support it. In order to avoid latency in communication the management of time-sensitive services have to be moved to the edge of the CoT. For this reason, the realization of a performing and secure Cloud-to-Edge middleware solution is a very strategic goal for future business related to CoT services. However, it needs to be deeply investigated, as highlighted by the Cloud Security Alliance (CSA), which identified several rules to take in account. However, achieving a compromise between performance and security is not trivial at all. A valuable approach to develop an efficient Cloud-to-Edge system is based on instant-message communication solutions. In current Cloud environments, Message Oriented Middleware (MOM) based on an instant-message protocol (IMP) look at performance but overlook security requirements. In this chapter, we aim at overcoming such a gap following the CSA guidance. In particular, we discuss the involved issues for improving such a kind of Cloud-to-Edge system in order to achieve data confidentiality, integrity, authenticity and non-repudiation. Moreover, we analyze a real case of study considering a MOM4Cloud architectural model. Experimental results performed on a real testbed show how the introduced secure capabilities do not affect the overall performances of the whole middleware.

5.1 Introduction

Nowadays, with the advent of Cloud-assisted Internet of Things or Cloud-of-Things (CoT) paradigm, new needs such as the real-time communication appeared, but traditional systems, based on the Cloud Computing paradigm do not support it. In order to avoid latency in communication the management of time-sensitive services have to be moved to the edge of the CoT. The communication system of a piece of Cloud-to-Edge middleware is quite complex because finding the right compromise between performance and security is not trivial at all. In fact, considering a worldwide CoT environment different issues have to be addressed. On one hand, according to the popular phrase of Benjamin Franklin "the time is money", a piece of Cloud-to-Edge middleware needs to quickly react to changes. For example, a Service Level Agreement (SLA) violation might cause loss of money for a Cloud provider. On the other hand, also security can have economic implications. To this regard a security leak can imply the disclosure of private data, hence loss of money for the provider. For this reason the Cloud Security Alliance (CSA) [59] has picked out the critical aspects of the Cloud security. According to the CSA guidance, security and privacy have to ensure the availability of services, resource access control, vulnerability mitigation, privacy of the audited user data. As CoT is an emerging paradigm, software architects have to deal with the lack of ad-hoc security standards and of a consolidated vulnerability model as point of reference.

How usually happens in emerging ICT technologies, software architects begin to face a new technology advancement adapting the existing systems to the new emerging ones. The same is happening with Cloud and Edgge computing, although this strategy is not resulting so effective. As a consequence, researchers and software designers are looking at new innovative solutions. An interesting approach is provided by Message Oriented Middleware (MOM) for Cloud systems. In particular, a MOM based on an Instant Messaging Protocol (IMP) and can considerably simplify the deployment of Cloud, IoT, and Edge services, since it allows to separate the communication and signalling system from the business logic. Securing a MOM is not easy at all. Existing IMPs allow to achieve a high level of reactiveness and performance, but they do not allow to obtain an adequate degree of security.

In our previous work we analysed the performance of secure communications for the management of federated Cloud [60] and IoT [61] environments. The objective of this chapter it to analyze the impact of the security of a MOM for Cloud-to-Edge based on an IMP.

More specifically, we will discuss how the overall communication system can be secured considering both the digital signing and data encryption mechanisms. In order to plan a security strategy, we analyse both the intra-module and the inter-module communications of a MOM based on XMPP [62].

Regarding security, although the XMPP support the SASL/TLS technologies, it presents some limitations. In order to overcome such a gap we will consider the *XEP-0373* [63] specification which describes the use of the protocol with the Open Pretty Good Privacy (OpenPGP) [64] for the integration of authentication and data encryption functionalities.

The rest of the chapter is organized as follows. Section 5.2 describes related works. More detailed motivations are discussed in Section 5.3. Details about the Security Model applied are provided in 5.4. Description of secure communication techniques are discussed in Section 5.5. Considering the CLoud-Enabled Virtual EnviRonment (CLEVER) [65] as case study, that is an implementation of MOM4Cloud, a few implementation highlights regarding the development of specifig security features are discussed in Section 5.6. Experiments evaluating the overhead of the security in the communication system are discussed in Section 5.7. Section 5.8 concludes the chapter.

5.2 Related Works

The influence and impact of mobile Edge computing on existing communication systems is discussed in [66]. In particular, by analysing existing Cloud systems, it is highlighted how even though the bind among Cloud and Edge is not so evident, there are several important features, such as security and resilience that have to be investigated. The challenges regarding the interconnection between Cloud and Edge computing environments are discussed in [67]. In particular, it is underlined that in order to achieve benefits it is needed to achieve high-throughput under high concurrent accesses, real-time processing performance, mobility support, and data persistency. A fuzzy-based security approach for mobile Fog and Edge computing to handle a scenario in which security services change according to mobile users is presented in [68]. In particular, a multi-criteria decision making methodology that is based on an innovative extension of the hesitant fuzzy rough set theory fused with a hesitant fuzzy soft set is presented. An alternative approach based on the fuzzy security service chaining concept for sustainable mobile Fog and Edge computing is discussed in [69]. The proposed architecture decouples the needed security mechanisms from physical resources. Moreover, a security proxy enables the compatibility with traditional security functions. A biometric security through visual encryption approach for Edge and Fog computing named Zero-watermarking is discussed in [70]. It is aimed at helping to protect the ownership of multimedia contents that are easy to copy and distribute. In particular, a biometric security solution able to analyse face images that does not impact the visual quality of images by using both visual cryptography and zero-watermarking is presented. A security analysis of mobile Edge computing consisting in the context of the European funded SESAME project is discussed in [71]. In particular, the security analysis of a piece of framework deployed in virtualised small cell networks extended in the broader of a 5G wireless network environment is discussed. Always in the domain of mobile Edge computing environment whose borders are interconnected with 5G wireless network, a study on security of dense small cells, which exploit Network Functions Virtualization (NFV) and that move the intelligence into the edges of the network is discussed in [72].

Most of the aforementioned solutions do not consider the presence of firewalls (whose ports have to be opened) that represent obstacles for the secure communication between the Cloud to Edge environments and vice-versa. Instead the last one, requires the creation of overlay networks by means of NFV mechanisms that have to be configured on network devices. In this chapter, we propose an alternative communication system solution for the interconnection between the Cloud and the Edge that is able to work at web layer (on port 80) and that is capable to bypass firewalls (because port 80 is commonly opened), guaranteeing at the same time security and privacy.

5.3 Motivations

CoT is not a new paradigm because of it is having a continuous growth in terms of availability and services' differentation. However, it is still not mature in definition of universally recognized standards. For this purpose, the lack of security standards represents one of the major points of weakness. In this area, the CSA conducted a valuable work, but also further security aspects that involve Clouds and their stakeholders have been formulated during the last years. The investigation presented in this section is extrapolated from the CSA guidance [73], which promotes the following:

- a common level of understanding between the consumers and providers of Cloud Computing regarding the necessary security requirements and **attestation of assurance**;
- an independent research into Best Practices for Cloud Computing security;

- educational and awareness programs on the appropriate uses of Cloud Computing and Cloud solutions;
- the creation of wide consensus lists of issues and guidance for Cloud security assurance.

Our work arises by a critical investigation of security requirements and issues reported into the CSA guide.

5.4 Security Model Making a MOM for Cloud/Edge Computing Compliant with CSA Requirements

In this Section, we describe how to make secure the interaction among the distributed software modules of a piece of MOM for Cloud/Edge computing, according to the previously highlighted CSA requirements.

5.4.1 Securing the Inter-Module Communication of a MOM for Cloud/Edge Computing

In a complex Cloud/Edge system as the one we are highlighting hereby, we can identify many actors and software modules needing to exchange data in a secure way. We believe that in the MOM of the future, all stakeholders interacting with Clouds should use strong security mechanisms in order to know **Who** is **Doing What** and **When**. Our innovative solution is aimed to provide these capabilities for both the following purposes:

- *tracking all the operations inside a Cloud system;*
- *guaranteeing the possibility to dynamically setup strong security environments when needed.*

For tracking all operations, we adopt the Digital Signature in conjunction with the Timestamp (T) recording. To setup a strong security communications system we adopt an hybrid model that is using asymmetric and symmetric keys.

Let's assume a generic module x_i that wants to send the n generic command message $CM_{n,y,j}$ to y_j module, where x is the transmitter entity and y is the receiver entity having both i and j respective modules. Our architecture has already an internal PKI, hence all entities have their couple of keys (PU and PR). In particular all entities can access the PUs Directory, for example with a Lightweight Directory Access Protocol (LDAP) service, for getting the PU of each entity compounding our complex system. In order to meet the CSA requirements discussed in Section , the following mechanisms have to be accomplished:

- mechanism a). An important goal to achieve is that all the communications should be tracked. This is possible by enabling x to sign CM_n using the DS technique, and sending the CM_n to y. In addition, x also includes the Timestamp (T) within the message. In this way, we can know what and when x requested to y. In the same way, y can do the same thing, when it send a response message back.
- mechanism b). Whether we predefined another goal in which all communications should be cyphered between two parties, the software modules assumes the following configuration:
  x encrypts CM_n using PU_y and the RSA algorithm, and sends the CM_n to y. Only y with its PR_y key can see the message (satisfying the needs 1,2,3 and 4 reported in the list above). x might also includes the Timestamp (T) into the message for tracking the date and time.
- mechanism c). Finally, if an entity x wants to communicate in total secretness with w, z and y entities, x becomes a *moderator* of the communication and setup the whole security environment. In particular, x has to generate a Random Session Key K_{rx}, (e.g. a symmetric key having 256 bits), and it has to send it to all parties included in the secret communication, using the technique described in *mechanism b)*. In this case x sends three different encrypted messages to w, z and y software modules respectively containing the encrypted K_{rx}. The K_{rx} will be used for a while and renewed again by x.

$$C_{CM} = e(Kr_{(x,z,y,w)}; P_{CM}) \tag{5.4.1}$$

$$P_{CM} = d(Kr_{(x,z,y,w)}; C_{CM}) \tag{5.4.2}$$

The hybrid security model guarantees better performance and the easiest way for distributing session keys among all the parties.

This approach consists in reusing consolidate security technologies useful for increasing the security level inside CoT providers, however, the overhead that these mechanisms imply in complex CoT systems is unclear. For this reason, in the rest of the chapter we will analyze the impact of such a mechanisms by extending a real MOM for Cloud/Edge, i.e., MOM4Cloud, that is based on one of the major IMPs, i.e., the XMPP.

5.5 Securing Communications Between Cloud and Edge Layers

5.5.1 Required XMPP Security Extension

Considering the CSA requirements, the XMPP protocol must to be extended. More specifically, a MOM for Cloud/Edge computing based on the this protocol should support the functionalities below:

- **Digital identity management** [74]. Each software module, during the in-band (i.e., an automatic XMPP client module registration on the instant messaging server) registration, asks for a X.509 digital certificate to a Public Certification Authority (PCA) through the Simple Certificate Enrollment Protocol (SCEP). The public key of each module is managed by a Lightweight Directory Access Protocol (LDAP) publisher server. Thus, when a module performs an in-band registration with the XMPP server, it verifies the corresponding digital certificate in the LDAP.

- **Signed message exchange**. Each module potentially can sign messages of other modules.

- **Encrypted message exchange**. Each module potentially can encipher the whole message or some its parts.

- **Private chat room**. The communication system potentially allow authorized modules to communicate in a separated chat room.

- **Encrypted chat room**. The chat room above described can be encrypted from a "moderator" module through a shared key. Modules, that join the communication, ask the shared key to the "moderator" module, which will send it cyphered by means of their public key. The keys are distributed by means of the PKI.

The aim of this work is to build security functionality on top of the XMPP. XMPP, that supports both the SASL and the TLS technologies, presents several limitations in authentication and encryption: the protocol does not natively supports digital signing and data encryption during the interaction between two entities, i.e., people or software modules. In fact, the security layers is demanded to developers because of the XMPP decentralized nature.

The protocol nature, wich is flexible and extensible, allows to integrate basic security mechanisms, improving the communication security level. The *XEP-0373* specification [63] describes how to include the Open Pretty Good Privacy methodologies (OpenPGP - RFC

4880 - [64]) into Jabber. Specifications for data communications, in terms of cryptographic privacy and authentication, are provided by the OpenPGP. Currently, the XEP-0373 does not represent a standard for authentication and data encryption in XMPP communications, but it describes a possible solution. It defines how the information piece inside tags must processed, by means of XML tags of specific *namespace*. However, the specification does not provide any rule about the public keys exchange. Indeed, this procedure is required to OpenPGP. Although, the chat messaging is related to the human interaction, it is able to be applied to the Cloud Computing systems in which distributed software modules need to interact in a secure real time way.

Given that a XMPP message has a XML encoding, adding an extension means adding a new tag element. In our opinion, in order to make the XMPP secure enough according to the security requirements previously discussed, four basic extensions are required: digital signature, data encryption, session key and timestamp. We briefly describe their features and how their implementation has been accomplished according to the XEP-0373 specification.

- **Digital Signature**. It is based on both a Public Key Infrastucture (PKI), using an asymmetric encryption algoritm, and on Message Digest algorithm 5 (MD5). It compute the MD5 of the message and signs it by means of the key of the sender.

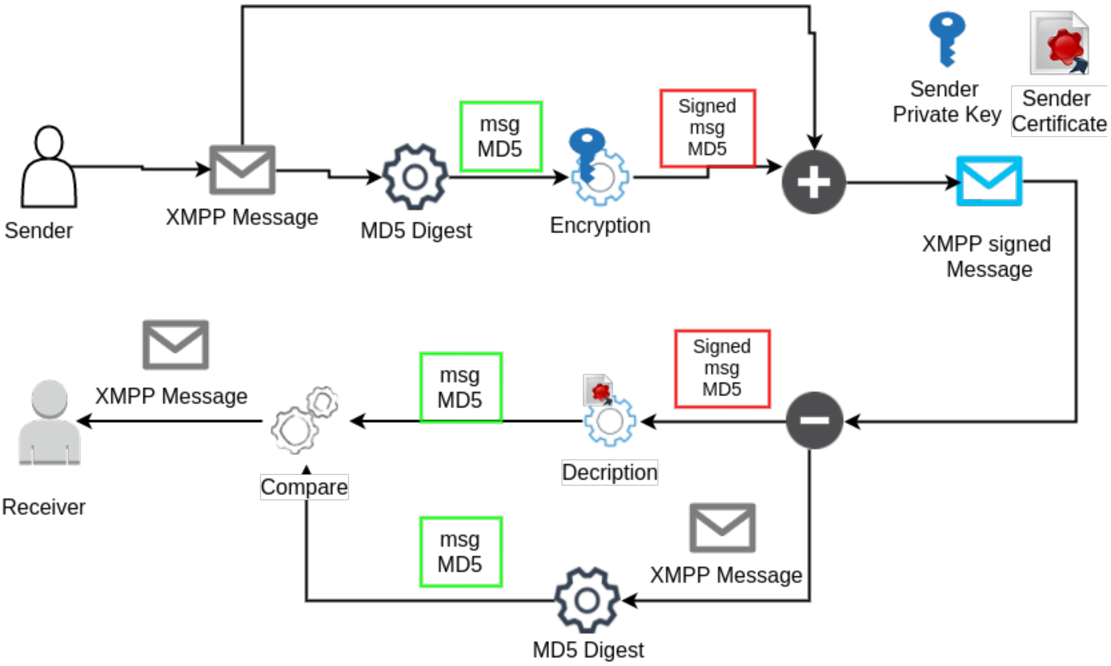


Figure 5.1: XMPP message signing.

Then, it attach the MD5 to the message and send them to the receiver. In such a way, the receiver, by means of the public certificate (retrieved from LDAP) can verify the authenticity of the message. The namespace jabber:x:signed identifies the signed extension. The above described process is represented in Figure 5.1. In XEP-0373 specification, the presence message in a chat room can be signed, in such a way a sender can sign its status. Listing 5.5.1 shows an XMPP message sent from the *HM* to the *CM* both acting in domain *example.org*.

```
<presence from='HM@example.org'
          to='CM@example.org'>
  <status>Online</status>
  <x xmlns='jabber:x:signed'>
      067A1704C780483726677B6C52860B6
      3F3CB87D27316205C72D5E6A680D320
      AFE1990E03C8D64151FBF7009738976
  </x>
</presence>
```

The receiver, the *CM*, by means of the certificate of the *HM* can verify the status in the example above.

- **Data Encryption**. Even this process is based on the PKI. It provides a method to send confidential information to the receiver attached to the message body that present TAGs with the namespace "jabber:x:encrypted". In order to send a chiphered message, the sender retrieve from LDAP the certificate of the receiver, then use it in order to chipher the message. Listing 5.5.1 shows a chipered message sent from the *HM* to the *CM*. The receiver, by means of its private key is able to open and to decipher the received message, Figure 5.2 shows the process.

```
<message to='CM@example.org'
         from='HM@example.org'>
  <body>This message is encrypted.</body>
     <x xmlns='jabber:x:encrypted'>
        1496611EB2B6420C0D2F10B71FEF5554d
        AF41F64337240118A18E63EBD9DD44E7P
        dWpR0uQsuJe7+vh3NWn59/gTc5MDlX8dS
        BA69E48CBDBA41BE450615B1CBBB3ECBp
        904D626F378473429FE649724BCA4CF9
```

```
            0168F20FD5D2CBC2CFCE08B2880
        </x>
    </body>
</message>
```

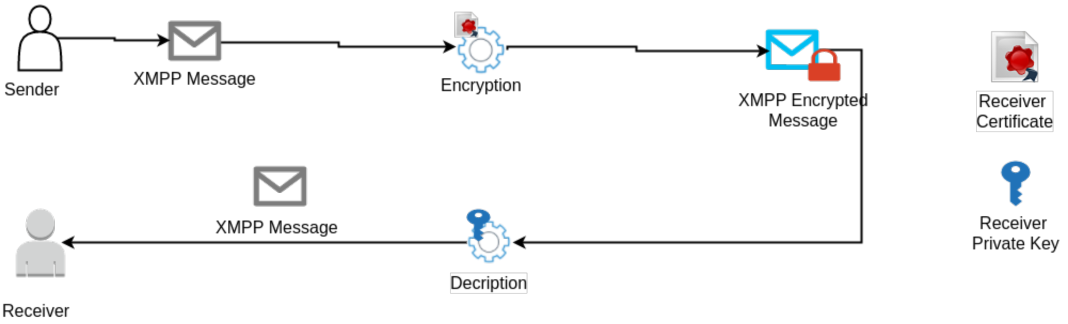


Figure 5.2: XMPP message encryption.

Typically, this extension is used to encrypt small piece of data, e.g., for delivering a session key (also called symmetric key or shared key) to different module, encrypted with their public key.

- **Session Key**. Typically, a message payload can hold a considerable data size. For this kind of data, an asymmetric encryption algorithm is faster than a symmetric one. For this reason, to make the inter-module communication over a chat room, the system uses a hybrid cryptographic scheme. In particular, when different modules have to communicate in a chat room, the module which starts the communication send to the other modules a session key encrypted with their public key using the encryption extension. Such a session key is set with a given Time To Live (TTL) and is periodically regenerated. Thus, when all modules obtained the session key, they can encrypt the message using the Session Key extension defined in the message by the XML name space jabber:x:sessionkey (*<x xmlns='jabber:x:sessionkey'>*). SSL/TLS protocol uses a similar mechanism. This cryptographic schema presents two advantages: processing times for the encryption/decryption of the message very fast and secure session key distribution. Such mechanism can be used when CM start a communication with different HMs.

- Timestamp. Signed MD5 is attached to the message, in such a way, the module that receives the message can verify when it was created and which module has created the message.

5.5.2 Securing the Inter-Module Communication of MOM4Cloud

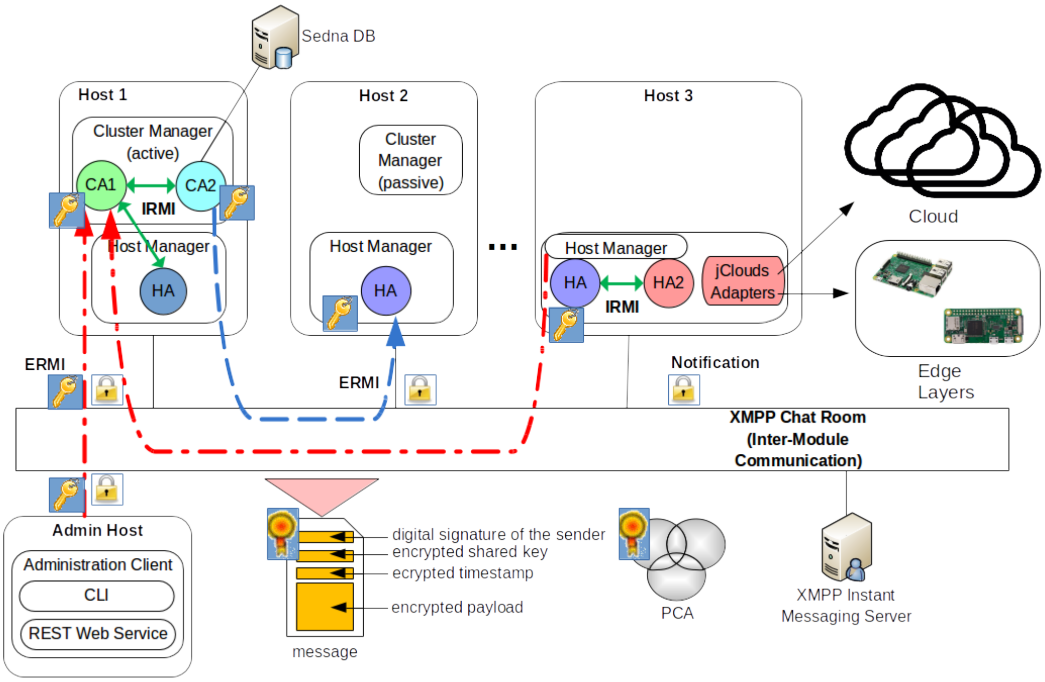


Figure 5.3: Distributed organization of MOM4Cloud modules.

The inter-module communication is based on both ERMI and Notification and it enables the interaction among different distributed MOM4Cloud modules via the XMPP. More specifically, it allows:

- CAtoCM communication for sending request to MOM4Cloud
- CMtoHM communication for cluster administration tasks.
- HMtoHM communication for specif task, e.g., a VM migration.

An inter-module communication is triggered

- when a request task is sent by the administration client, the CM, and the HM.
- when an event occur.
- according to a regular scheduled task, for example for data synchronization.

In order to secure each MOM4Cloud message by means of the XMPP security extensions, i.e., digital signature, encryption, session key, and timestamp, as shown in Figure 5.3 a PCA trusted third party is required.

In the following we describe how we have extended both the CM and the HM modules in order so include the XMPP extensions. When one of the aforementioned situations occurs, a MOM4Cloud message is formatted by the CM and sent to a destination HM via the XMPP on a specific chat room. At destination, the HM interprets the message, execute the task, and if a response is required, it will format and send a response message to the CM

MOM4Cloud, by means of the integration of PKI, SCEP, CA, and LDAP is able to provide a digital identity for each MOM4Cloud element and to secure the inter-module communication. By means of the SCEP, MOM4Cloud modules are able to obtain from the PCA their key pair. Then, a local KeyStore by each module is created. Each KeyStore contains the certificate in PKCS# format and the private key, KeyStores are protected by password. MOM4Cloud modules publish on the LDAP server their certificate. MOM4Cloud modules in order to establish a secure inter-module communication need the access to LDAP to retrieve certificates and their digital identity. In such a way, modules are able both to sign messages by means of their private key and to cipher the message by means of the certificate of the target module. Receiver, in order to verify the signature interrogates LDAP for retrieving the public key of the sender, instead, in order to share a symmetric key it uses the PKI. The authentication/encryption version presents two more steps: the ciphering operation of the message before the sending and the deciphering of the message after the reception.

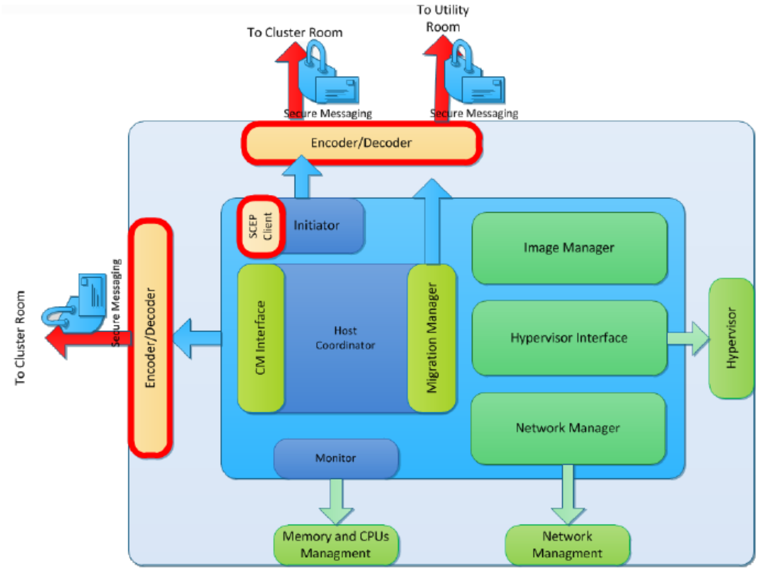


Figure 5.4: Security integration in HM.

Figures 5.4 and 5.5 highlights with thick borders the software modules that have been secured in the HM and in the CM respectively. As depicted in Figures 5.4, a *SCEP Client* has

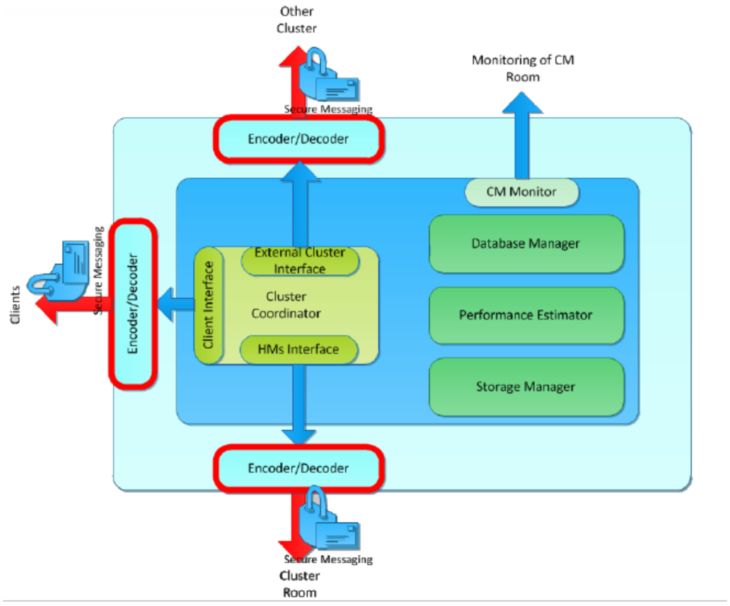


Figure 5.5: Security integration in CM.

been integrated within the *Initiator* module. It is responsible for digital identity initialization. The *SCEP Client* interacts with the PCA requiring X.509 certificates to perform both authentication and data encryption. This is made by integrating the *Encoder/Decoder* module, i.e., a secure inter-module communication interface that enables data integrity, authenticity, confidentiality, non repudiation. More specifically, this module enables mutual authentication and secure communication among the AC and the active CM, among the CM and the HMs, and among different clusters in a federated inter-domain scenario. Considering the HM the *Encoder/Decoder* module acts as a secure communication interface for the Initiator module and the HC module. In the first case, it allows to establish a secure communication through the Cluster and Utility rooms that are respectively used for basic cluster management and for value-added services. In the second case, the *Encoder/Decoder* module is used for securing the communications of the HC module respectively for the interaction with the CM and for performing a secure VM migration into another HM. As shown in Figure 5.5, within the CM the *Encoder/Decoder* module is used by the Cluster Coordinator module respectively for cluster administration communication, inter-domain communication with HMs, and external cluster communication.

5.5.3 CSA Requirements & MOM4Cloud

In this subsection, we analyze the matching between CSA requirements presented before and MOM4Cloud.

With regards the possibility to **sign exchanged messages**, by means of **Timestamps** and valid **Signatures**, it is possible to prove data integrity and assure data non-repudiation. According to item 2, segregation of duties is possible considering c**hat rooms protected by password**. Only authorized modules can access the communication system enabling them to perform given tasks, e.g., only authorized administrators can access the Cluster room, only authorized HM can access an Utility room, and so on. Protected chat rooms allow to carry out a comprehensive compartmentalization of systems and management according to item 3. Onsite inspections from customers is guaranteed from **Sign exchanged messages** and **Timestamps**. The issue reported in bullet 5 are solved by means of the SEDNA database and the logs of the XMPP.

5.6 Implementation

Hereby, we provide several implementation highlights related to development of XMPP security extensions in CLEVER, that is an implementation of the MOM4Cloud architectural model. In order to develop the Java classes, the following tools have been adopted.

- Java API able to sign and validate XML documents by means of **XML Digital Signatures (XMLDsig)** [75];
- Java APIs for the administration of a CA by means of **EJBCA Web Service** [76];
- Public Key Infrastructure (PKI) by means of **Enterprice Java Bean Certification Authority (EJBCA)** [76];
- Simple Certificate Enrollment Protocol (SCEP) by means of **Java Simple Certificate Enrollment Protocol (JSCEP)** [77];
- Lightweight Directory Access Protocol (LDAP) by means of **OpenLDAP** [78];
- Java APIs for LDAP by means of **Java Naming and Directory Interface (JNDI)** [79];
- Java cryptographic algorithms library by means of **Bouncy Castle** [80].

  In our tests we adopted the following algorithms, according to the required functionalities:

  - Data encryption: AES, that is a symmetric-key algorithm based on the Rijndael cipher;

- Signing: RSA-1024, that is an algorithm for public-key cryptography with keys of 1024 bits;
- Digest generation: SHA-1, that is a cryptographic hash function able to produce a 160-bit message digest.

5.6.1 XMPP Security Extension Design

The security extensions were developed through several Java classes implementing the *Smack PacketExtension*: *SecureExtension* and *Provider*. The former allows the definition of XMPP security extensions, while the latter allows the receiver to correctly process the different XMPP security extensions.

The *SecureExtension* class allows to define security extensions to the XMPP. It involves two attributes: *type* and *data*. The first indicates the type of extension and the corresponding namespace, whereas the second represents the extension of the content. The Provider class performs the parsing of the XML messages at the receiver. In order to allow a correct parsing of the encrypted data, digital signature, and session keys, it has been further specialized in three classes: *EncryptedProvider*, *SignedProvider*, *SessionKeyProvider*. Each of these three specialized classes overrides the *parseExtension* method of the *SecureProvider* super class, that returns a PacketExtension object.

In order to integrate the security extensions in the XMPP, we developed the *ConnectorThread* class implementing the *Runnable* interface, representing an XMPP client. It registers the packet Listener and three Providers for the management of the security extensions respectively represented by the *EncryptedProvider*, *SignedProvider*, and *SessionKeyProvider* classes. The *ConnectorThread* allows an XMPP client to establish a connection with the XMPP server, to register an account, to setup the digital identity of the client. As soon as the client establishes the connection with the XMPP server, the SCEP client is called and it sends an *enrollment* request to the trusted PCA, that returns an X.509 digital certificate. After that, the XMPP client creates a *KeyStore* local object of *Map* type containing a private/public key pair. In order to improve the security of the whole system a password will protect the key pair. Once the client obtains its credentials, it waits for incoming requests. The XMPP client is able to create/join a chat room and send clear, encrypted, or signed messages on the chat room. The signed and encrypted messages are formatted using the methods of the *SecureExtension* class. Its main methods are:

- **encryptedMessage()**. It allows to send encrypted messages using the public key of the receiver XMPP client.

- **signedMessages()**. It allows to send signed messages by means of an attached signed message digest.
- **groupChatEncryptedMessage()**. It allows to send encrypted messages on a given chat room.

In order to manage the incoming XMPP messages three Listener classes have been developed extending the PacketListener super class. Each specialized Listener class overrides the processPacket() method. The three Listener classes are:

- *MessagePacketListener*. It manages clear and encrypted messages.
- *PresenceListener*. It manages the presence messages.
- *GroupChatPacketListener*. It manages the presence messages in an encrypted chat room.

5.7 Experimental Assessment and Analysis

In this section, we perform scalability and efficiency tests in order to evaluate the overhead of the introduction of security mechanisms.

5.7.1 System Settings

In our tests, we considered three nodes connected by means of a GB-LAN, Figure 5.6 shows the environment.

In our experiments, the nodes are 3 blade servers IBM X3630M3, characterized by the following hardware configuration: CPU Intel(R) Core(TM) i7-4790K CPU @ 4.0 GHz with 4 cores and 4 threads, RAM 16GB, GFLOPS 181, OS: Ubuntu server 16.04 LTS 64 BIT.

From a logical point of view, a node works as the AC at Cloud layer, that is responsible to connect several tenants (e.g., administrators or third party software systems) to CLEVER.

We deployed CM and HM in the other two nodes that are acting respectively at Cloud and Edge layers.

- Step 1: each tenant, in order to introduce traffic of data in the system sends to the AC the `listvms`command that queries all registered Virtual Machine in the specific HM;
- Step 2: the AC sends to the CM the corresponding XMPP message formatted following the CLEVER specifications.;

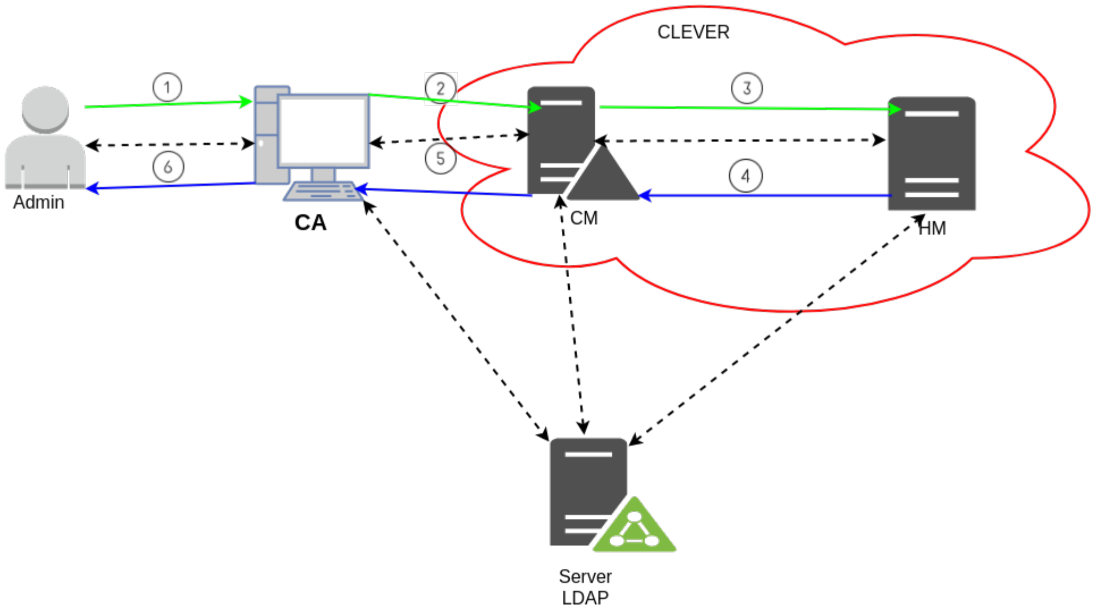


Figure 5.6: Testbed configuration.

- Step 3: CM sends to the specific HM the request;
- Step 4: HM process the request and replies to CM;
- Step 5: CM sends to the AC the answer;
- Step 6: AC display information about VMs to the tenants.

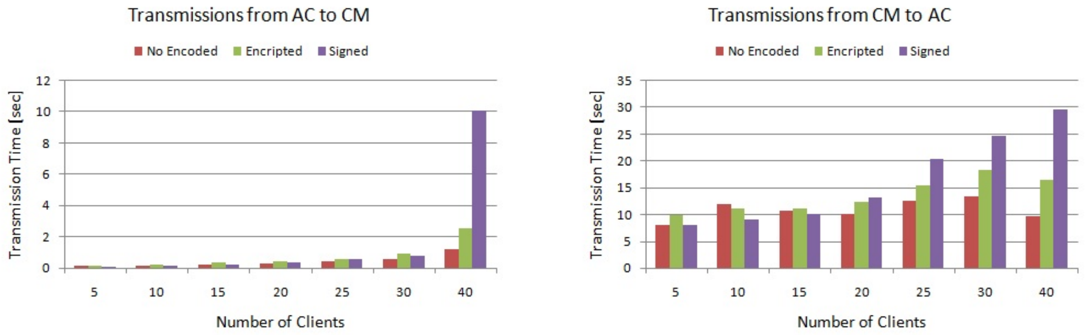
In order to simulate concurrent requests for Cloud services, in our experiments, a thread for each tenant in the AC node was instantiated, they send command at a random time. In such a way a lot of messages in a short time interval are generated, this causes high overhead in the AC for a consistent number of tenants. This scenario, well describes typical service providers (e.g., web portals), indeed it works as collector of tenant queries. For the evaluation analysis we are considering the following metrics:

- Transmission Time (T_{tran}:, the time spent to send a formatted message from a sender to a receiver. It provides informations on the overloading of the system, buffers fillings and nodes congestion.
- Elaboration Time (T_{elab}), that measures the complexity of message management and provides the overhead due to data encryption and signing. We separately deal with the processing time in message transmission T_{elab_TX} and in message receiving T_{elab_RX}.

In the following, we provide the experimental results in terms of the above metrics assuming request/response messages. More specifically, we analyzed AC to CM and CM to HM communications for the request and HM to CM and CM to AC communications for the response. We performed 50 subsequent experiment in order to consider confidence intervals at 95% and average values.

5.7.2 Transmission Time

We evaluate the overhead introduced from the security extensions in terms of delay for trasferring XMPP messages. As previously we described, the testbed machines are interconnected by means of a GB LAN, for this reason we can assume that the network latency is negligible. In such a way, our result can be used in order to evaluate a scenario with a certain network latency.

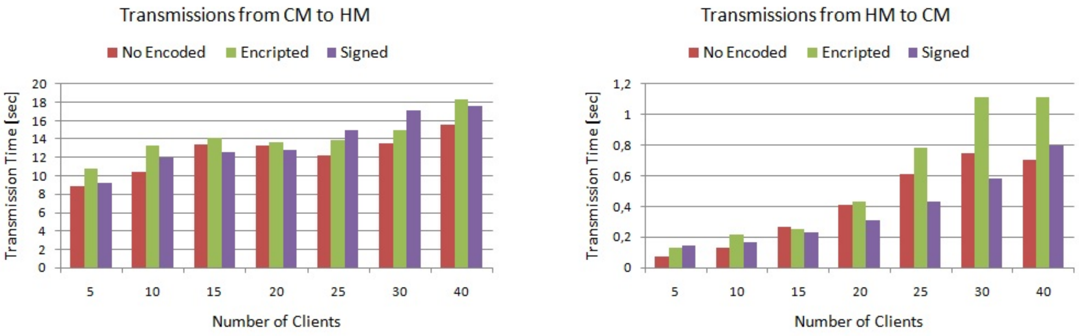


(a) From Admin to CM

(b) From CM to Admin

Figure 5.7: Trasmission time between AC and CM.

Figure 5.7(a), analyzes message exchanges between AC and CM. We remark that we have different thread in the AC that perform their queries in parallel. T_{tran} increases with the tenant number, indeed an higher number of request, sent in a small time interval, are stored in the trasmission buffer. T_{tran} in the secure scenario is greater than the plain one, due to the introduction of security mechanisms. Indeed the ciphering of data, introduces a query to LDAP in order to retrieve the certificate. Thus, transmission buffers hold both data messages and requests for the LDAP server, resulting in a higher transmission delay. Trasmission delay in the encrypted scenario increases with the number of the tenant from $0,16$ to $2,56$ seconds in average. Unortunately this behavior is not constant, indeed in some cases, when the message size is bigger, it can increases more than one hundred times (from $0,10$ to $10,09$ seconds). The SHA-1 produce a hash 160 bits long for any length of a generic message m. Furthermore, the RSA algorithm includes one block more of 128 bytes. The total message digest length is 148 bytes. In our experiments, we considered very mall messages (about 1-2 KBytes), the digital signature increase the data size of about 10%, causing a fast filling of the buffers. Regarding the CM, the system behavior is a bit different as shown in Figure 5.7(b). In fact, unlike the AC, the CM manages messages g back, so that the message insertion rate in the buffers is slower than in the AC buffers. Specifically, T_{tran} measurements for uncoded messages are almost constant. T_{tran} curves for secure messaging are smoother than in Figure 5.7(a), with a maximum increase of about 65% for encrypted messages and 270% for signed messages. The average T_{tran} for the CM (13, 61 sec) is higher than for the AC (0, 97 sec) since the CM is involved in communications also with the HM, thus its transmission suffers of two concurrent data flows. Such a stressed transmission overhead is proved by the results shown in Figure 5.8.



(a) From CM to HM

(b) From HM to CM

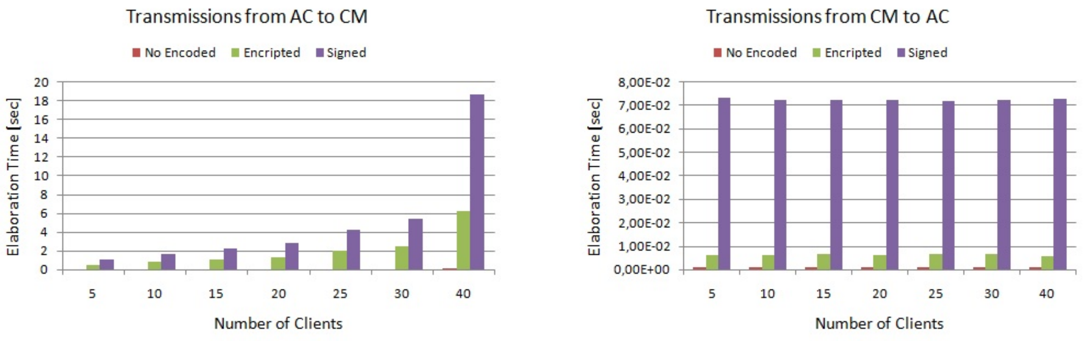
Figure 5.8: Transmission time between CM and HM

Figure 5.8(a) shows that the mean T_{tran} measured at the CM is 13, 45 seconds, which is similar to the T_{tran} observed in transmissions from the CM to the AC, thus proving that transmissions in the CM are fairly divided between the two communication flows. On the contrary, the average T_{tran} measured at the HM and shown in Figure 5.8(b) is very low, since the HM is involved only in transmissions with the CM. It is also lower than the average T_{tran} measured at the AC, since the HM elaborates one message at time, thus reducing transmission delay.

5.7.3 Elaboration Time

$T_{elab-TX}$ evaluates the time all the tasks spend to manipulate a message before sending it to the destination. It includes:

- symmetric key management and message coding for data encryption;



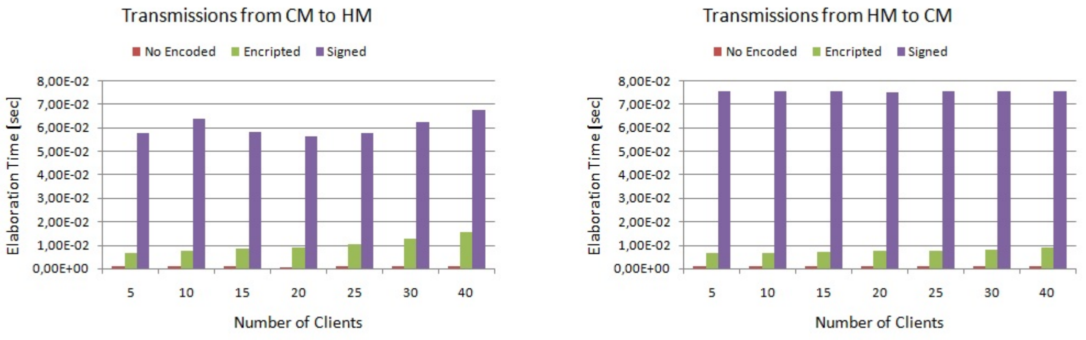
(a) From AC to CM

(b) From CM to AC

Figure 5.9: Elaboration time in transmission.

- coding of message digest for data signature.

As shown in Figure 5.9(a), $T_{elab-TX}$ is comparable to T_{tran} when measured at the AC. Otherwise, it is much lower than T_{tran}. This is the effect of the specific request generation mechanism implemented in the testbed, where concurrent threads work in the AC. Thus, the elaboration and the transmission tasks influence the effective delays measured at the AC node. This system behavior explains why $T_{elab-TX}$ increases by increasing the number of tenants. On the contrary, the progressive management of messages in the CM decouples the elaboration tasks from the transmission ones; the measured $T_{elab-TX}$ is not affected by node overloading and it remains constant, despite of the increasing of the number of tenants as shown in Figure 5.10(b). From the tenant point of view, the experienced $T_{elab-TX}$ is pretty much the delay due to activities being executed in the AC. The results shown in Figures 5.9 and 5.10 prove that the message signing activity strongly impacts the elaboration load. This overhead is introduced by the asymmetric encryption. Indeed it is heavier than the symmetric.



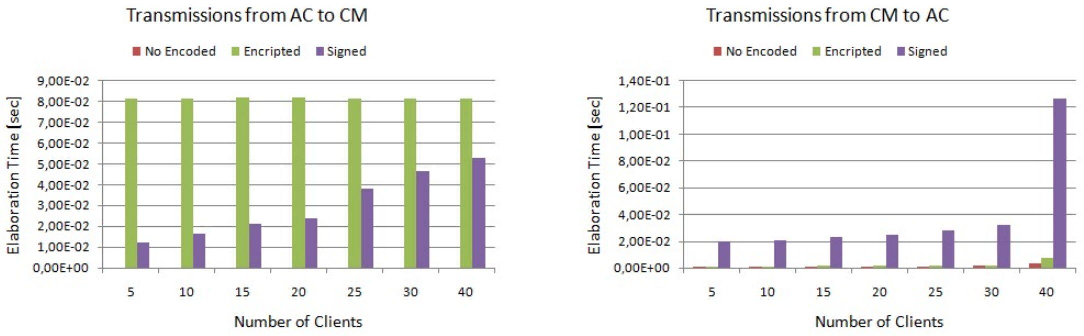
(a) From CM to HM

(b) From HM to CM

Figure 5.10: Elaboration time in transmission.

Analyzing the results related to the elaboration time in the CM and HM shown in

Figure 5.10, we notice that $T_{elab-TX}$ for encrypted messages is higher when the communication occurs from CM to the HM (on average it is 10 msec) than from HM to CM (on average it is 7 ms). This result is due to the management of the symmetric keys mainly affecting the communications from the CM to the HM. To send a message to the HM, the CM has to share a symmetric key to encrypt the message. On the contrary, the HM sends messages to the CM using the same key previously shared. The transmission of signed messages is characterized by similar values of $T_{elab-TX}$ as shown in Figures 5.10(a) and 5.10(b) (around 65 msec), since message signing is carried out only using the private key of the transmitter.



(a) From AC to CM

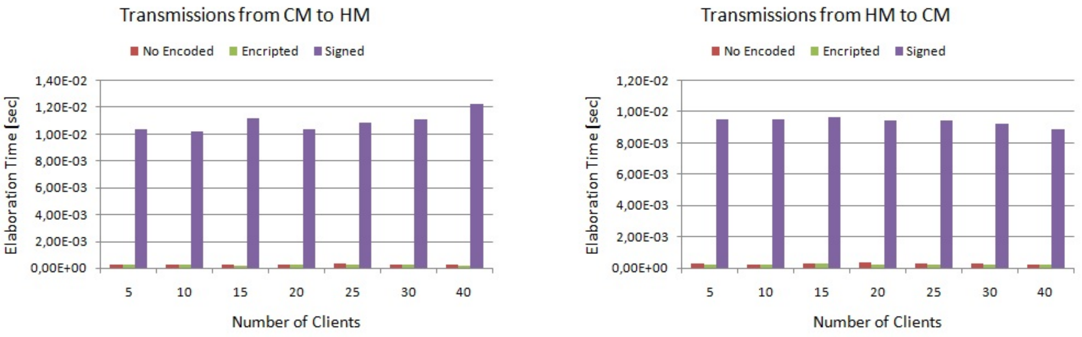
(b) From CM to AC

Figure 5.11: Elaboration time in receiving.

$T_{elab-RX}$ evaluates the time to execute for all the tasks necessary to decode a message whenever it arrives at the destination. It includes:

- message decoding using the shared key for data encryption;
- recovery of the source's public key and decoding of message digest for data signing.

Figure 5.11(a) shows that at the AC the time spent to elaborate received messages is drastically lower than the time experienced in transmission. This is a consequence of the serial message management of the CM, which implies that, usually, one tenant at time receives a message and is involved in effective elaboration. Thus $T_{elab-RX}$ is not influenced by the overload of the node. The decoding of encrypted data is almost independent from the number of tenants, since it is carried out by using the already shared key. On the contrary, to decode signed messages, each tenant in AC has to recover the public key of the CM through a query to the LDAP server, so that $T_{elab-RX}$ raises with the number of tenants since the number of connections to the LDAP server grows. As depicted in Figure 5.11(b), we can provide similar considerations with regard $T_{elab-RX}$. When the CM receives a signed message, it has to retrieve the public transmitter key for the tenant sending the message and causes an overload of the CM with the number of tenants.



(a) From CM to HM

(b) From HM to CM

Figure 5.12: Elaboration time in receiving.

$T_{elab-RX}$ between the CM and the HM is plotted in Figure 5.12. The decoding of encrypted messages using the shared key requires an elaboration time comparable to the time for elaborating uncoded messages (about $0,2$ msec). With reference to signed messages in Figure 5.12(a), $T_{elab-RX}$ is almost constant with the number of tenants since the CM makes just a query to the LDAP server for elaborating the first message of the HM. Then it uses the same key for all the other messages originated from the HM (even if they derive from different tenant requests). The same behavior characterizes the signed messages that are received and elaborated at the HM. As shown in Figure 5.12(b), similar results can be appreciated considering $T_{elab-RX}$ during the trasmission from HM to CM.

Encryption activities introduced in order to privide security features such as confidentiality, authenticity and non-repudiation of data, proposed in this scientific work cause delays. However, considering all the management activities of the Cloud middleware such delays are still acceptable.

5.8 Conclusions and Remarks

Nowadays, with the advent of Cloud-assisted Internet of Things (IoT) or Cloud-of-Things (CoT) paradigm the need of manage in a secure and flexible fashion VMs and containers in both Cloud and Edge layers is becoming more and more compelling. In this chapter, we investigated several of the major concerns described by the Cloud Security Alliance (CSA) guidance, i.e., 1)Governance and Enterprise Risk Management; 2) Information Management and Data Security; 3)Data Center Operation; 4)Incident Response, Notification and Remediation; 5)Traditional Security, Business Continuity and Disaster Recovery; 6)Encryption and Key Management.

In order, to study the process required to enable Cloud/Edge MOM to address the aforementioned concerns, we extended security features in CLEVER, that is one of the reference implementation of the MOM4Cloud architectural model. More specifically, we highlight the involved issues and discussed how to make its communication system secure.

Different modules of the CLEVER middleware communicate each other using the XMPP. Even though, the XMPP presents several interesting capabilities in the context of Cloud computing, it does not natively support the security features required to address the concerns described in the CSA guidance. Thence, considering the CLEVER case of study, we specifically focused on how to make its XMPP communication system secure in order to meet the security requirements of Cloud/Edge environment for he management of CoT services deployed in virtualized environments.

The performance analysis on the developed security extensions shows the effectiveness of the proposed approach. The proposed work can be considered a landmark for software architects who want to make their Cloud/Edge systems compliant with the CSA guidance. For future works we plan to move to the innovative Osmotic Computing paradigm[81].

CHAPTER 6

Enabling the secure store of files by using Secret Share techniques

A common practice to store data is to use remote Cloud-based storage systems. However, storing files in remote services can arise privacy and security issues, for example, they can be attacked or even discontinued. A possible solution to solve this problem is to split files into chunks and add redundancy by means of Secret Share techniques. When it comes to Internet of Things (IoT), Edge and Cloud environments, these techniques have not been evaluated for the purpose of storing files. This work aims to address this issue by evaluating two of the most common Secret Share algorithms in order to identify their suitability for different environments, while considering the size of the file and the availability of resources. In particular, we analysed Shamir's Secret Share schema and the Redundant Residue Number System (RRNS) to gauge their efficiency regarding storage requirement and execution time. We made our experiments for different file sizes (from 1kB up to 500MB), number of parallel threads (1 to 4) and data redundancy (0 to 7) in all aforementioned environments. Results were promising and showed that, for example, to have seven degrees of redundancy, Shamir uses eight times more storage than RRNS; or, Shamir is faster than RRNS for small files (up to 20 kB). We also discovered that the environment on which the computation should be performed depends on both file size and algorithm. For instance, when employing RRNS, files up to 500kB can be processed on the IoT, up to 50MB on the Edge, and beyond that on the Cloud; whereas, in Shamir's schema, the threshold to move the computation from the IoT to the Edge is about 50kB, and from the Edge to the Cloud is about 500kB.

6.1 Introduction

Nowadays, more often users adopt remote services such as Clouds (public or private) to store files. However they can arise several security threats including the following cases. Firstly, hackers could violate the privacy of people by attacking these servers and publish their private images on Internet. In fact, this has been a real concern during the last years; for example, Google revealed that more than 1 million of Google Accounts were hacked by Gooligan in three months [82]; Yahoo disclosed that hackers reached between 1 to 1.5 billions accounts [83]; and, Dropbox denounced that more than 68 million account details were disclosed [84]. Secondly, Cloud service providers can be discontinued as it happened to Copy-Cloud [85], Ubuntu One [86], DELL Data Safe [87], etc. Thirdly, users can be affected by the vendor lock-in, and thus their data cannot be easily migrated among Cloud service providers [88]. Fourthly, servers can be affected by the ransomware (e.g., WannaCry [89]) and become the subject of ransom payments. A possible solution to alleviate the chance of occurring any of the aforementioned sources of threat is to employ techniques for providing secure storage systems to encrypt data, and consequently protect its content from unauthorised people. Furthermore, by adding redundancy, they can recover original data even if services are discontinued or attacked by viruses.

In literature, several techniques have been discussed, among which, 'Secret Share' algorithms to split a secret in shares and distribute them among parties have drawn the highest amount of attention to themselves. These techniques, however, have not been used and evaluated for vastly heterogeneous environments such as Edge-Computing platforms that consist of devices in all layers of Internet of Things (IoT), Edge and Cloud.

To address this issue and make this study, our contribution in this work can be summarised as the followings. Firstly, we analyse and compare Secret Share algorithms; and secondly, we identify the most suitable environment to run each algorithm considering the size of the file and the computational capacity of the node. To be more specific, we discuss about Shamir's Secret Share schema and the Redundant Residue Number System (RRNS). For each technique, we will consider IoT, Edge and Cloud environments, and investigate both storage required and execution time. We performed experiments for different file sizes (from 1kB up to 500MB), number of parallel threads (1 to 4), and degree of redundancy (0 to 7).

The rest of the chapter is organised as follows. Section 6.2 describes related works. Section 6.3 discusses mathematical foundation of Secret Sharing techniques. Experiments are discussed in Section 6.4. Conclusions and our future directions are summarised in Section 6.5.

6.2 Related Work

Secret Share is a very hot topic within the scientific community; yet, there are not so many works available in literature focusing on these techniques for IoT, Edge and Cloud devices.

In [90], authors compared storage requirements and computational complexity of different Shamir's Secret Share implementation. In particular, they compared the schema proposed by Chien with Yang's and their schemes. The Chien's implementation for recomposing costs $(n + p - t)$ simultaneous equations (where n is the total number of parts which the secret is divided, t is the minimum number of needed chunks for recomposing each secret, and p is the number of secrets). The authors' schema and Yang's ones are able to recompose the original data by using Lagrange interpolation polynomial that is lighter than solve multiple simultaneous equations [91].

In [92], authors discussed visual Secret Share techniques for storing images. In particular, they proposed a boolean multi secret sharing scheme applied to grey scale images. This system is able to encrypt k images by using $k + 1$ cover images. In order to recover original images, a boolean XOR operation is performed. Although the proposed system fulfils the security criteria of Secret Share algorithms, it modifies the information contained in images; for example, secret image pixel values equal to 119, 34, 153 will result in reconstructed image pixel values of 124, 44, 156.

In [93], authors surveyed and compared a few Secure Share techniques for embedding shares in cover images. In particular, they considered 7 aspects: (1) secret sharing threshold, (2) embedding technique, (3) payload, (4) strength, (5) meaningful shadow image, (6) lossless secret image, and (7) lossless cover image. They show that some characteristics (5)-(7) are independent of implementation, whereas others strongly depend on the implementation.

In [94], authors proposed a Visual Secret Share (VSS) for colour QR codes. Their idea is to divide a colour QR code into three different QR codes, and to reconstruct the original information by using the human visual system (HVS). The proposed approach is based on a (n, n) schema. This means that the original information can be reconstructed, if-and-only-if, all chunks are recovered.

In [95], authors analysed Secret Share algorithms for the SM2 digital signature algorithm. In particular, they demonstrated that ordinary Secret Share techniques cannot be applied to

SM2. Therefore they proposed the "product-based Secret Share (PBSS)" and the "summation-based Secret Share (SBSS)". The basic idea of these techniques is to use several devices in order to sign with the SM2 algorithm. The PBSS acts as (n, n) Secret Share technique; whereas the SBSS acts as (k, n) Secret Share technique –it means that the re-composition can be made if k devices over n are available.

In [96], authors discussed generalised random grid for implementing the Secret Share. In particular, with reference to binary images, they proposed a lossless schema. However, it is based on the (n, n) Secret Share algorithm.

In [97], authors discussed a parallel implementation of the Shamir's Secret Share scheme. In particular, they designed and developed an algorithm that is able to run on GPUs. They compared performances of sequential versus parallel version of the algorithm for two different schemes: (k, n) and (n, n). In both cases the parallelised version ran faster than their sequential implementation.

In [98], authors propose a technique that can be applied to Secret Share algorithms in order to identify cheaters or verify the integrity of shares. The basic idea is to distribute, within each share, a list containing the hash values of all other shares. In order to verify the integrity of the information in each share, it is necessary to compare the hash value of each share with values contained in all lists.

6.3 Secret Share Algorithms

Secret Share, a.k.a. Secret Split, is a cryptographic technique that allows dividing data S into p shares/fragments ($S_1, S_2, \ldots, S_p$), where $p > 1$. It splits S in such a way that each of $\binom{p}{p-1}$ combinations composed of $p - 1$ chunks does not contain enough information to reconstruct the original data S. p also indirectly reflect the level of "security" in the system, because, the larger the p, the greater number of chunks is required to reconstruct S, and thus harder to infiltrate the system or steal information from it. Without losing generality, it is possible to split the data S into n fragments, where $n \geq p$, in such a way that the data S can be reconstructed by using only p chunks. This would lead to $\binom{n}{p}$ different combinations of chunks that can reconstruct the secret S. The difference between n and p ($r = n - p$ where $r \geq 0$) is called the "redundancy" level; it represents the maximum number of fragments that can be lost. In other words, to have r degrees of redundancy, the secret S must be split into n chunks, out of which, any p numbers of them is sufficient to reconstruct S. The main advantage of using Secret Share algorithms, instead of encrypt the whole data and then split it in chunks and transmit them among several storage provider (to be discussed in Section 6.4), is related to the capacity of storage required. That is, in order to guarantee r degrees of redundancy with secret share algorithms, we have to split the file into $n = p + r$ chunks, as opposed to $r + 1$ times that we require is the simple data encryption and splitting approach.

6.3.1 Shamir's Secret Share

The Shamir's Secret Share schema [99] is based on the interpolation of polynomials based on the fact that k distinct 2D-points (x, y) are sufficient to represent a polynomial with the degree of $k - 1$, for example, two 2D-points to represent a line, or three 2D-points to represent a parabola.

In Shamir's algorithm, a generic polynomial function is able to divide a secret S, which is a number, into $p = k - 1$ parts using the following formulation:

$$f(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + ... + \alpha_\mathrm{p} x^\mathrm{p} \tag{6.3.1}$$

where $S = \alpha_0$.

Hence, to divide S into p fragments, it is sufficient to calculate $f(x)$ into p distinct points; that is, $\{(1, f(1)), (2, f(2)), \ldots, (p, f(p))\}$. If we want to add r degrees of redundancy, $f(x)$ needs to be computed for $n = p + r$ distinct points; that is, $\{(1, f(1)), \ldots, (p, f(p)), \ldots, (n, f(n))\}$. In both cases, to recover the original secret S, it is sufficient to interpolate a subset of p distinct points to evaluate the resulting function $f(x)$ for $x = 0$.

6.3.2 Blakley's Secret Share

The Blakley's Secret Share schema [100], such as the Shamir's one, is based on the interpolation of geometric objects. In particular, it is based on interpolation of planes. That is, two non-parallel 2D-plans intersect in a line, and three of them in a point (x_0, y_0). Considering 3D-plans, two of them intersect in a 2D-plan, three in a line, and four in a point (x_0, y_0, z_0). In general, considering $k - 1$-dimensional hyper-plans, at least k non-parallel hyper-plans are needed to find a common (intersect) point $(x_1, x_2, \ldots x_\mathrm{k\text{-}1})$. In this scheme, a secret S can be stored using a generic point Q that contains the secret; that is, $(x_1, x_\mathrm{i}, \ldots x_\mathrm{p})$, where $x_1 = S$, $p = k - 1$ and $x_\mathrm{i} \forall i \in [2, p - 1]$ are random values. The equation of a generic hyper-plane that contains this point is the following:

$$y = \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_\mathrm{p} x_\mathrm{p} + \alpha_\mathrm{k} \tag{6.3.2}$$

In order to share the secret S with p parts, it is necessary to create p different equations:

$$\begin{aligned} y_1 &= \alpha_{1,1}x_1 + \alpha_{1,2}x_2 + ... + \alpha_{1,\mathrm{p}}x_\mathrm{p} + \alpha_{1,\mathrm{k}} \\ \dots \\ y_\mathrm{p} &= \alpha_{\mathrm{p},1}x_1 + \alpha_{\mathrm{p},2}x_2 + ... + \alpha_{\mathrm{p,p}}x_\mathrm{p} + \alpha_{\mathrm{p,k}} \end{aligned} \tag{6.3.3}$$

with $i \in [2, p-1]$. It can be summarised as a linear system $CX = Y$, where matrix $C_{p,p}$ and the vector $Y_{p,1}$ correspond to hyperplane equations [101]. The solution of the system is the point $Q(x_1, x_2, \dots x_\mathrm{p})$. As we discussed, x_1 is the secret S. In order to add redundancy to the system, it is sufficient to create $n = p + r$ different equations. In order to retrieve the secret, it is sufficient to compose the matrix C by using only a subset of equations.

From the space point of view, the Blakley's schema is less efficient than Shamir's one [100], because shares are not points but hyper-plan equations. However, under particular conditions, it can be re-conducted to the Shamir's schema.

6.3.3 Proactive Secret Share

The Proactive Secret Share (PSS) [102] is a technique based on the Shamir's Secret Share. The goal of PSS is to overcome the main security issue of Shamir's schema, where attackers can retrieve enough fragments to recompose secret S after many attempts. The basic idea of the PSS is to periodically renew shares. In order to do that, the owner of each share has to periodically execute the following steps: (1) calculate new coefficients for itself and all other parties, (2) share new coefficients with all other parties, (3) sum received coefficients to original shares, (4) delete old shares, and (5) save new shares.

As can be inferred, PSS is more secure than Shamir, but almost impractical for widely distributed systems (such as IoT-Edge-Cloud), because, it requires many synchronisation rounds among parties. From a computational point of view, it is less expensive to adopt the Shamir's schema with new coefficients.

6.3.4 Redundant Residue Number System: RRNS

The Redundant Residue Number System (RRNS) [103] is based on the Residue Number System (RNS). RNS choose p prime moduli called primaries $(m_1, \dots, m_p)$ such that $m_i > m_{i-1}$ $\forall i \in [1, p]$. RNS can represent numbers up to M that can be defined as:

$$M = \prod_{i=1}^{p} m_\mathrm{i} \tag{6.3.4}$$

Therefore, for any secret $S \in [0, M]$, it is possible to calculate the residue of $s_i = S$ modulus $m_i \forall i \in [1, p]$ and store it into a p-tuple $((s_1, s_2, \cdots, s_p))$ called the *Residue Representation* of S, in which, s_i is the i^{th} residue digit of the representation. For every p-tuple $(s_1, s_2, \cdots, s_p)$, the corresponding S can be reconstructed by means of the Chinese Remainder Theorem; that is:

$$S = \left(\sum_{i=1}^{p} s_i \frac{M}{m_i} b_i \right) \mod M \tag{6.3.5}$$

where b_i, $i \in [1, p]$ is such a way that $\left(b_i \frac{M}{m_i} \right) \mod m_i = 1$.

The RRNS is defined as the RNS, with the difference in the number of modules; that is, RRNS defines $n = p + r$ moduli $(m_1 ... m_p)$ such that $m_i > m_{i-1}$ $\forall i \in [1, n]$. The range (the largest number) of the RRNS, M is:

$$M = \prod_{i=1}^{p+r} m_{\mathrm{i}} \tag{6.3.6}$$

that includes r degrees of redundancy.

As demonstrated above, both RNS and RRNS using Chinese Remainder Theorem are able to reconstruct all numbers within $[0, M]$. The only difference is in the Residue Representation. RNS has only a Residue Representation, whereas the number of Residue Representations of RRNS is equal to $\binom{n}{p}$. The RNS and RRNS are fundamentally different from Shamir and Blakley, because, the number of primaries they consider is not arbitrary. Furthermore, RNS and RRNS require much less storage than Shamir and Blakley.

6.4 Performance assessment

As we discussed in previous sections, we are going to use Secret Share algorithms to encrypt files, split them into chunks and store them on different storages so that the original file could be recomposed even if several chunks are lost, or their hosting storages are attacked or decommissioned. As shown in Figure 6.1, there are two procedures (split and merge) in using Secret Share algorithms. The 'split' algorithm is to divide the input file into chunks and stores them on different services. The 'merge' algorithm is to reconstruct the original file using several available chunks (3 out of 4 in this figure).

'Split' and 'merge' algorithms can be divided in two parts: sequential and parallel. With reference to Shamir split, the sequential part: (1) reads the input file, (2) converts the datatype, and (3) define the equation 6.3.1. Each thread of the parallel part of the Shamir split simply calculates the equation 6.3.1 in a different point and save the result in a file. The sequential

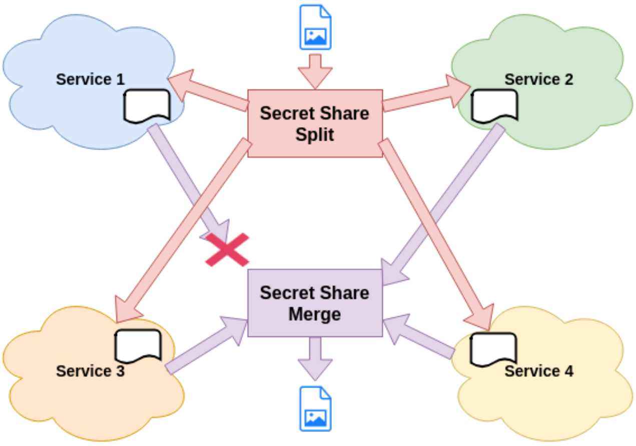


Figure 6.1: Secret Share Split and Recomposition

part of the RRNS split, instead: (1) reads the input file, and (2) converts the datatype. Each thread of the parallel part of the RRNS split make the modulo operation between each secret and a specific 'primary' and save the result in a file. Also the 'merge' algorithm for both techniques can be divided in parallel and sequential parts. The parallel part simply read information contained in each chunk, and then recompose the original file by using Lagrange interpolation function (Shamir) or Chinese Reminder Theorem (RRNS).

As we discussed in Section 6.3, Blakely's schema and PSS can be re-conduct to Shamir. Therefore, in this Section, we evaluated only Shamir and RRNS techniques from two different points of view: storage utilisation and execution time of their 'split' and 'merge' operations. To this end, as summarised in Table 6.1, we make scalability analysis for different file sizes, levels of redundancy and number of parallel threads.

We executed our tests in three different test-beds: IoT, Edge and Cloud. The IoT device is a Raspberry Pi 3+, the Edge device is a mid-range physical server, and Cloud is a powerful Virtual Machine (VM). The Hardware (HW) and Software (SW) configurations of these environments are shown in Table 6.2.

6.4.1 Storage Analysis

The amount of storage required for saving files is not a trivial aspect, especially for IoT devices with limited resources. In this Section, we analysed and compared the behaviour of RRNS and Shamir techniques for different degrees of redundancy using only one thread for splitting a file (a JPG image or a small video in our case). With $p = 5$ (number of primaries), we will have 5 chunks for zero level of redundancy (r), and 12 chunks to achieve seven degrees of redundancy.

Table 6.1: Summary of experiments performed.

Parameter	Values
Environments	IoT, Edge, Cloud (table 6.2)
Secret Share algorithms	RRNS, Shamir
Functions	Split, Merge
Primaries (P)	5
Redundancy (R, only Split)	0, 1, 4, 7
Parallel Thread	1, 2, 4, 8
File size [B]	1k, 2k, 5k, 10k, 20k, 50k, 100k, 200k, 500k, 1M, 2M, 5M, 10M, 20M, 50M, 100M, 200M, 500M
Number of Iterations	30
JDK/JRE	1.8.0_201

Table 6.2: Hardware and Software characteristics of testbeds

Environment	RAM	CPU	Storage	OS
IoT (Raspberry Pi 3)	1 GB LPDDR2-900 SDRAM	1.2 GHZ quad-core ARM Cortex A53	MicroSD 8GB (Write throughput: 10.9 MB/s; Read throughput: 22.6 MB/s)	Raspbian
Edge (Server)	8 GB DDR3-1600 SODIMM	2.3 GHz quad-core Intel(R) Core(TM) i5-6200U	SSD 500GB (Write throughput: 869 MB/s; Read throughput: 528 MB/s)	Ubuntu 16.04 LTS
Cloud (VM)	16 GB DDR3-1600 SODIMM	3.4 GHz eight-core Intel(R) Core(TM) i7-3770 CPU	SSD 100GB (Write throughput: 998 MB/s; Read throughput: 367 MB/s)	Ubuntu 16.04 LTS

The Shamir's algorithm needs more storage compared to the other techniques; it starts from ∼13 times the size of the input file and increases up to 32 time of it. This is because, as we discussed in Section 6.3 and shown in Equation 6.3.1, several non-negative numbers need to be added for each secret. To avoid computational overflows, we read secrets as 'shorts (16 bits)' and store them as 'integers (32 bits)' in our Java implementations of these algorithms. Therefore, the size of each chunk is twice the size of the original file plus 33% more because files are encoded as Base64. The mathematical formula to describe Shamir's storage requirement is:

$$Size_{chunks}^{shamir} = (p + r) \times 2 \times (1 + \frac{1}{3}) \times Size_{in} \tag{6.4.1}$$

where $Size_{chunks}^{shamir}$ is total size of all chunks, and $Size_{in}$ is the size of the input file.

The RRNS algorithm requires less amount of storage; in fact, for $r \leq 2$ (level of redun-

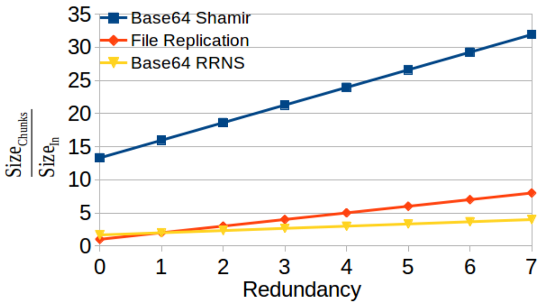


Figure 6.2: Storage requirement for different degrees of redundancy

dancy), the storage requirement is comparable to a simple file replication. For higher degrees of redundancy, RRNS requires ∼50% and ∼12.5% of storage as compared to Shamir and simple-replication, respectively. This is due to the fact that each share in RRNS is the remainder of the division operation between the secret and a modulo, and thus its value is always lower than the modulo. As a result, shares will be smaller sizes than the secret as well. In our implementation, we used 'integers (32 bits)' as primitive type of the secret and 'bytes (8 bits)' as primitive type of modules and shares. Hence, the size of each chunk becomes four times smaller than the input file, plus 33% because chunks are encoded in Base64. The RRNS storage requirement can be computes as:

$$Size^{RRNS}_{chunks} = \frac{p+r}{4} \times (1 + \frac{1}{3}) \times Size_{in} \tag{6.4.2}$$

For the simple-replication strategy, in which, the content of a file (or an encrypted version of it) is simply copied several times, the storage requirement can be described as:

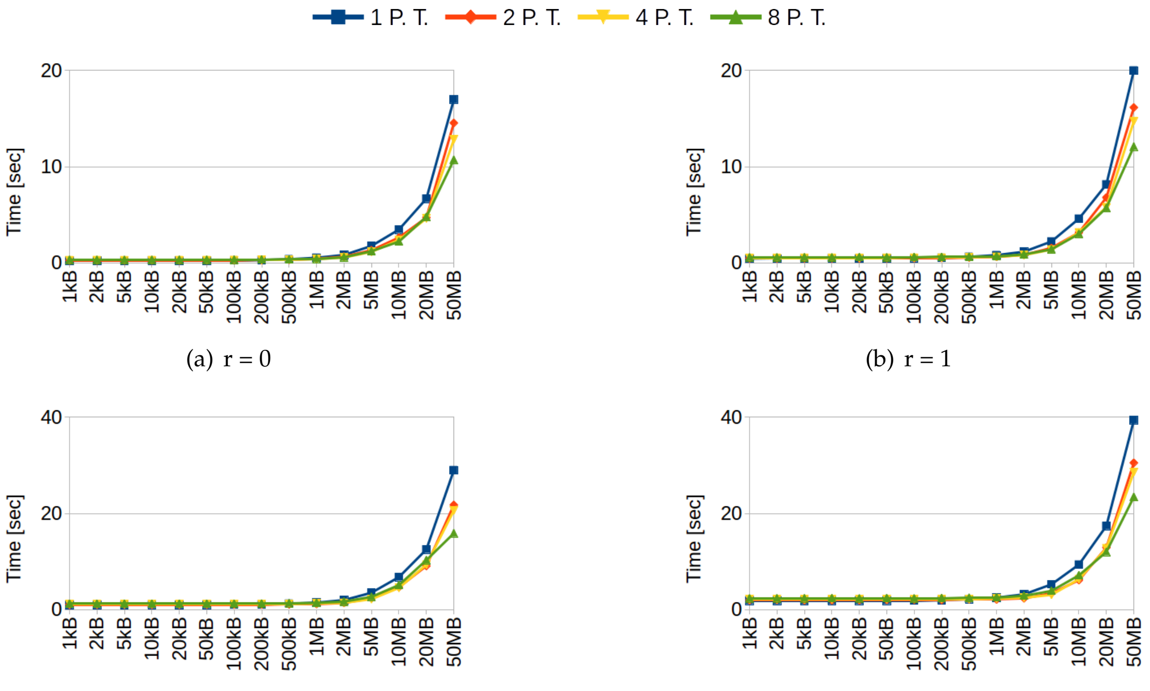
$$Size^{simple}_{out} = (r+1) \times Size_{in} \tag{6.4.3}$$

where r is the number of replicas plus '1' for the original file.

6.4.2 Time Analysis

The time required for the 'split' and 'merge' procedures on data files (images/videos in our experiments) is a fundamental factor for the applicability of these techniques in real scenarios. In this Section, we analyse them in all discussed scenarios (Table 6.1) and test-beds (Table 6.2).

Figures 6.3–6.5 show the behaviour of the RRNS split. In these figures, sub-figures (a),(b),(c),(d) refer to $r = 0, 1, 4, 9$ (redundancy) with $5, 6, 9, 12$ total number of chunks, respectively. In all sub-figures, the X axis represents the size of the file in Byte, and the Y axis represents the execution time in seconds. Each experiment was repeated 30 times and its average value is shown in these sub-figures.



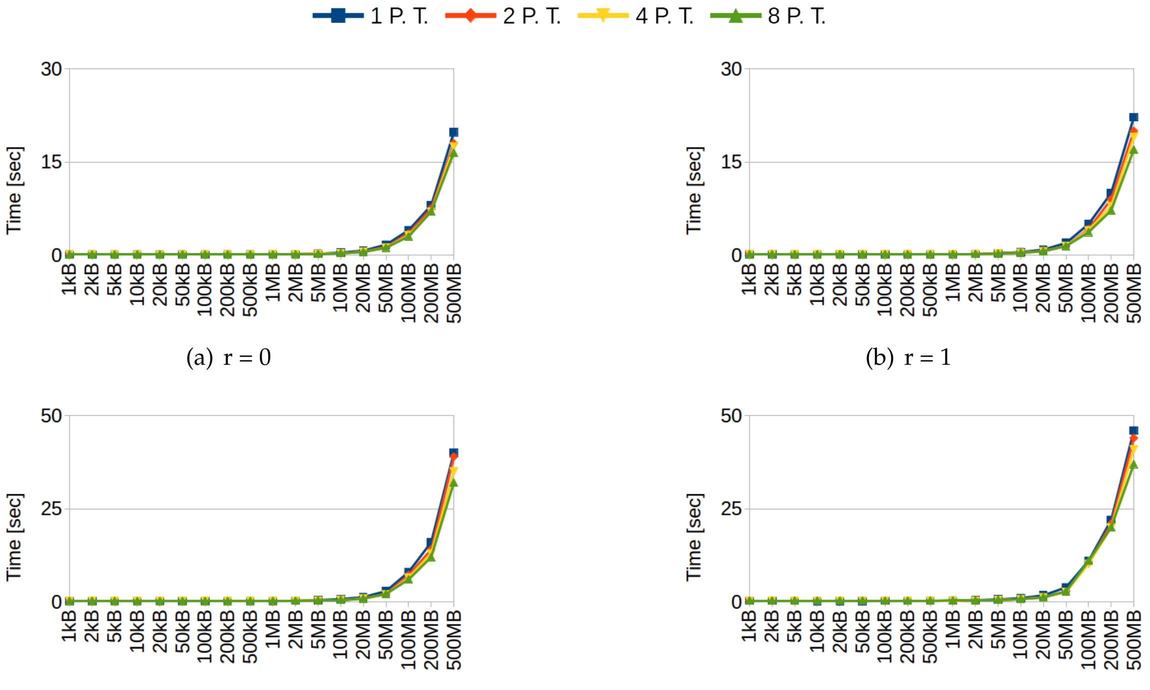
(a) r = 0

(b) r = 1

(c) r = 4

(d) r = 7

Figure 6.3: RRNS split on IoT



(a) r = 0

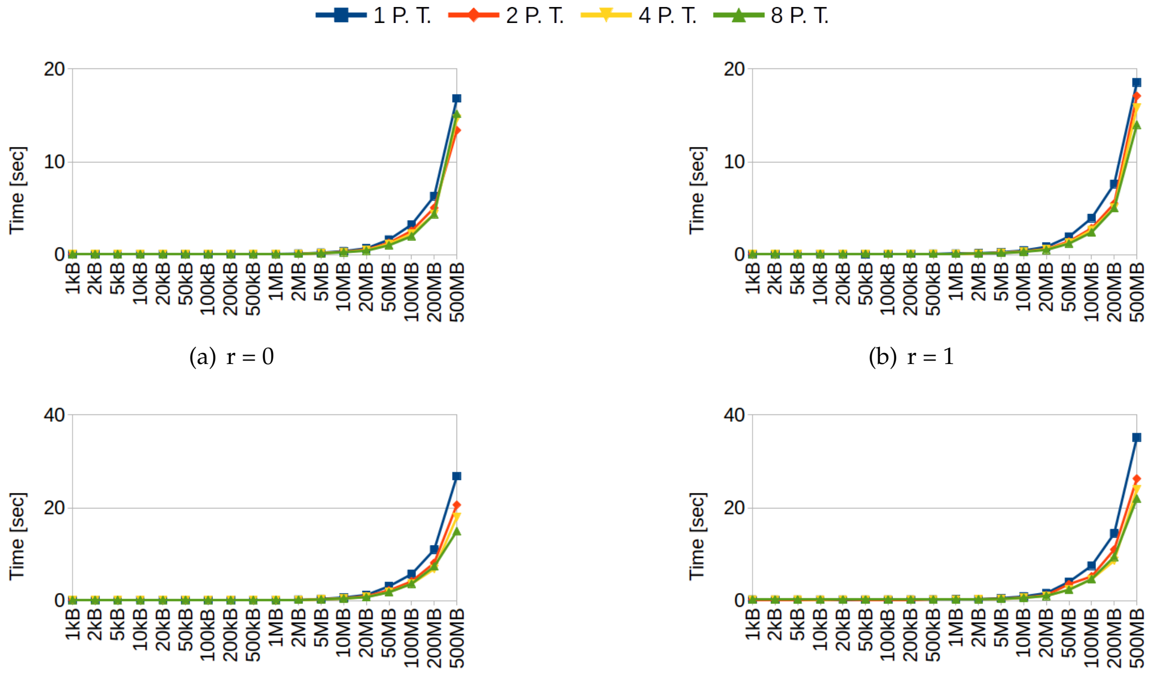
(b) r = 1

(c) r = 4

(d) r = 7

Figure 6.4: RRNS split on Edge

As can be observed, the execution time increases with both the redundancy level and the



(a) r = 0

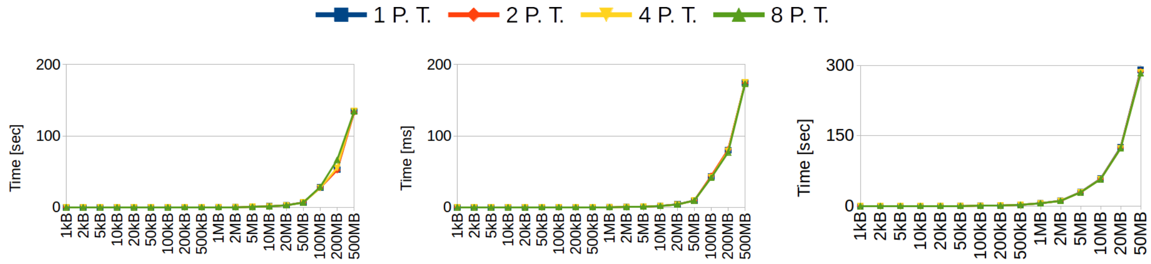
(b) r = 1

(c) r = 4

(d) r = 7

Figure 6.5: RRNS split on Cloud

file size. In particular, the time required for the computation in configuration (d) is double than that of (b); mainly because, the number of chunks that have to be calculated is double as well. For each environment, we stopped our analyses when we saw noticeable differences in the behaviour of the algorithm; that is, for file sizes up to 50MB for IoT, and up to 500MB for Edge and Cloud.



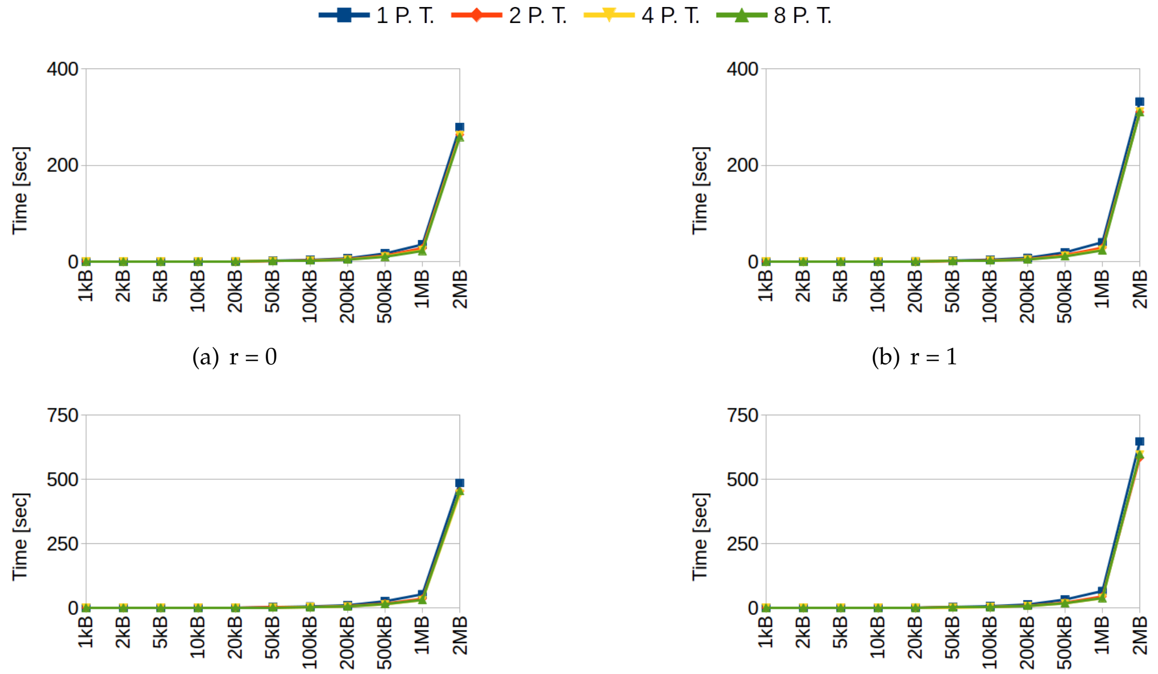
(a) Cloud

(b) Edge

(c) IoT

Figure 6.6: RRNS merge on Cloud, Edge and IoT

Figure 6.6 shows the behaviour of the RRNS merge, where sub-figures (a),(b) and (c) relate to Cloud, Edge and IoT, respectively. Unlike the 'split' case, we did not consider the value of the redundancy for 'merge', because RRNS reads only the minimum number of chunks (i.e., p) in parallel and performs the heavier task of merging using only one thread. Thus, as can also be observed in this figure, increasing the number of parallel threads does not improve its execution time.



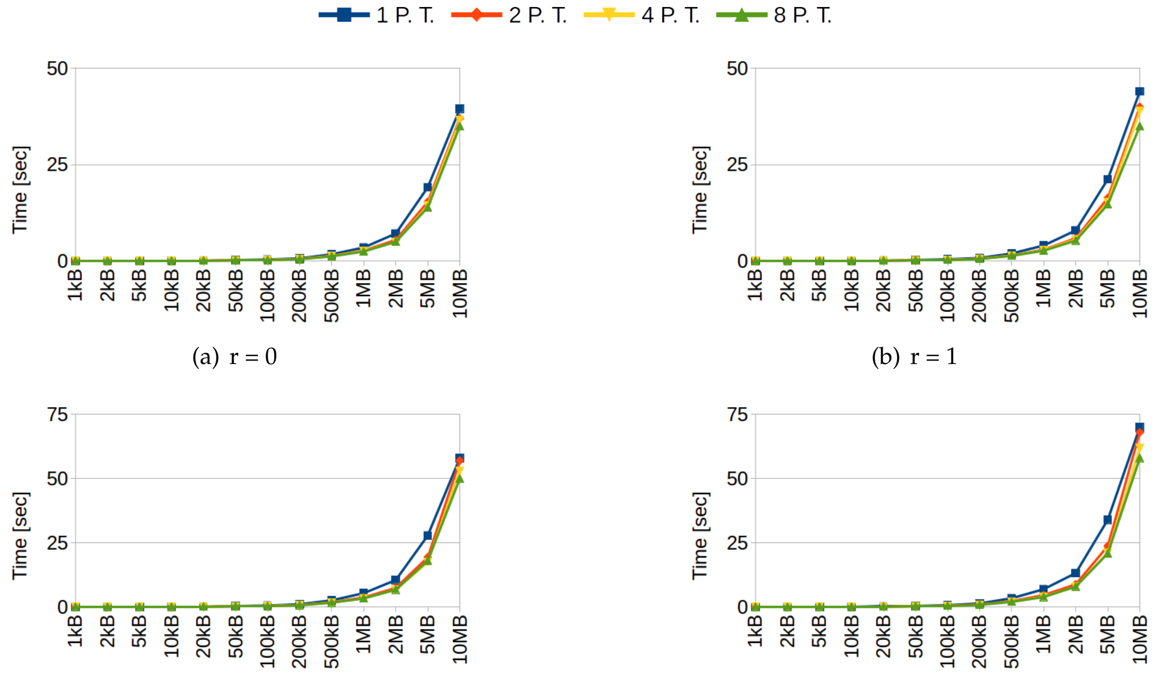
(a) r = 0

(b) r = 1

(c) r = 4

(d) r = 7

Figure 6.7: Shamir split on IoT



(a) r = 0

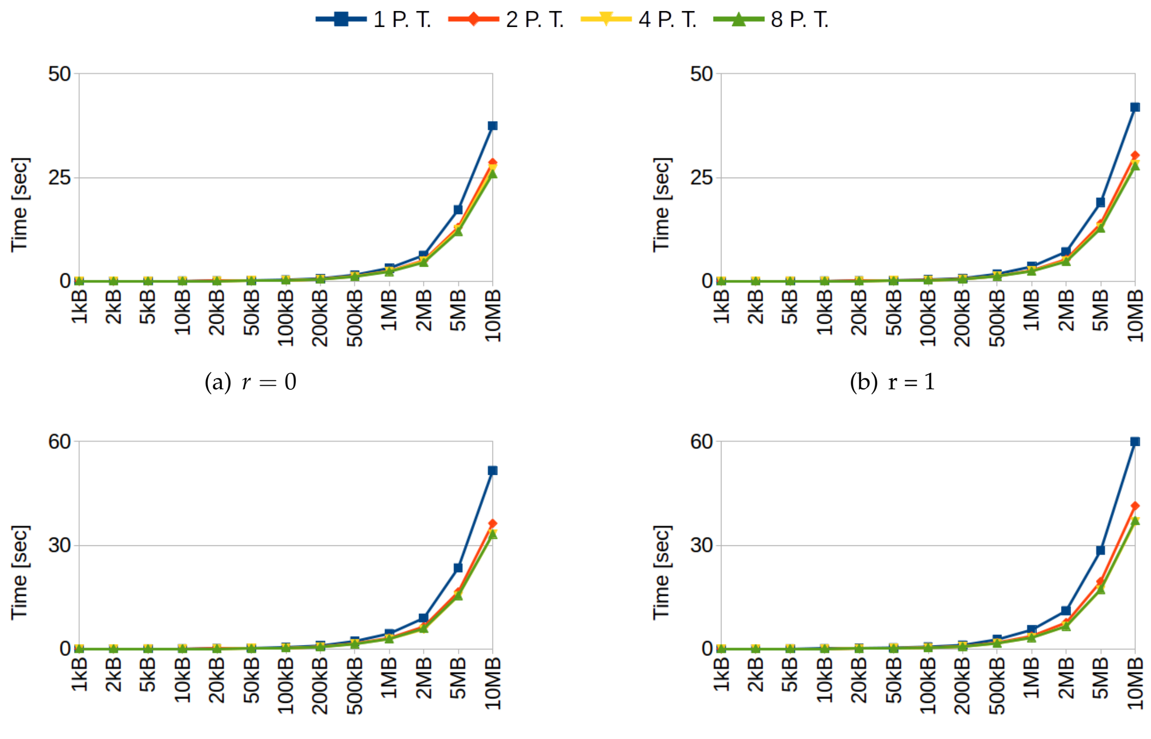
(b) r = 1

(c) r = 4

(d) r = 7

Figure 6.8: Shamir split on Edge

Figures 6.7–6.9 show the behaviour of the Shamir split. Similar to Figures 6.3–6.5, Figures 6.7–6.9(a),(b),(c),(d) show results for $r = 1, 2, 4, 7$, respectively. With reference to the IoT scenario, we can see that the improvement gained by increasing of the number of parallel threads is negligible, because, the sequential part of the algorithm is very slow compared to the parallel one. For Shamir, similar to RRNS, we also stopped our tests when we saw noticeable differences in the execution time; that is, to process files up to 2MB for IoT, and up to 10MB for Edge and Cloud.



(a) $r = 0$

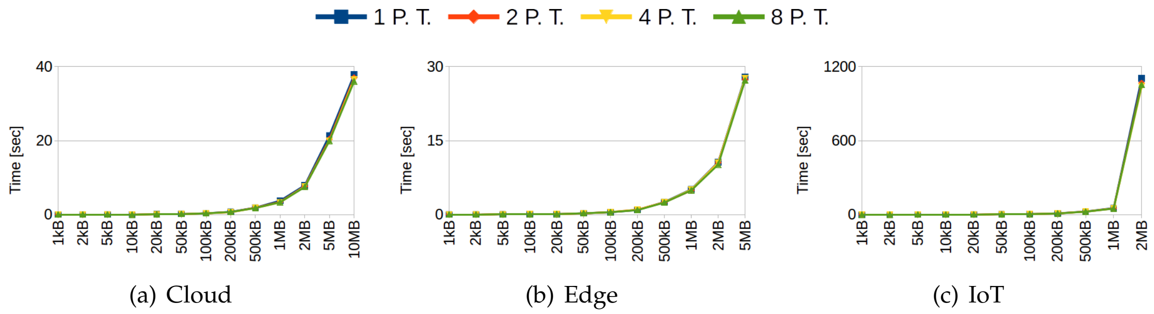
(b) r = 1

(c) r = 4

(d) r = 7

Figure 6.9: Shamir split on Cloud

Figure 6.10, shows the behaviour of Shamir merge in (a):Cloud, (b):Edge and (c):IoT. Similar to RRNS merge, Shamir also read the minimum number of chunks in parallel and made the re-composition using a single thread; and therefore, running more threads in parallel does not provide any improvement.



(a) Cloud

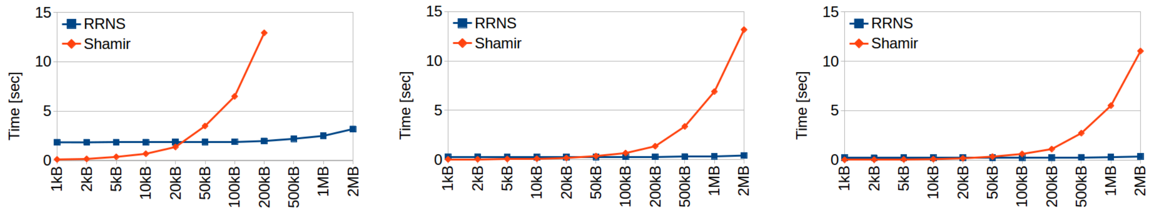
(b) Edge

(c) IoT

Figure 6.10: Shamir merge on Cloud, Edge and IoT

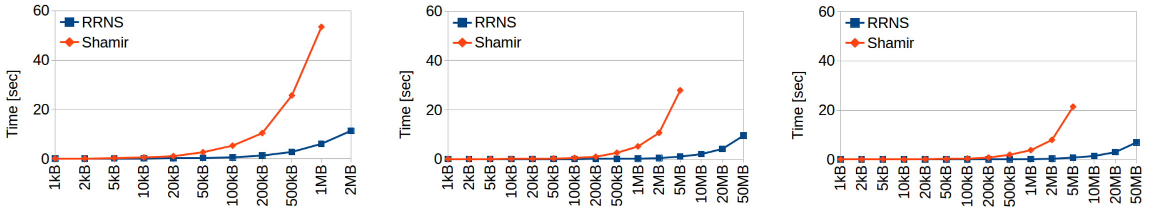
6.4.3 Discussion

In this Section, we will summarise our findings of neck-to-neck comparison of RRNS and Shamir with regards to their storage requirements and execution time in relation to the file-size, execution environment, redundancy degree, etc. For IoT devices, we saw that the RRNS uses less storage than Shamir, especially for high values of redundancy. For execution time on IoT devices, Figure 6.11 shows the behaviour of their 'split' functions in IoT (a), Edge (b) and Cloud (c) environments. This figure shows that, although RRNS is slower than Shamir for smaller files, it scales better; that is, its execution time stays almost constants for file sizes up to 2MB, whereas increases exponentially for Shamir's. Figure 6.12 shows their behaviour for the 'merge' function, also in this case Figures (a),(b) and (c) refer to IoT, Edge and Cloud respectively. As can be seen, execution time of RRNS and Shamir are close for smaller files, while Shamir algorithm becomes notably slower than RRNS for files larger than 50kB (IoT) and 500kB (Edge and Cloud).



(a) IoT (b) Edge (c) Cloud

Figure 6.11: Execution time of RRNS vs. Shamir split on IoT, Edge and Cloud devices



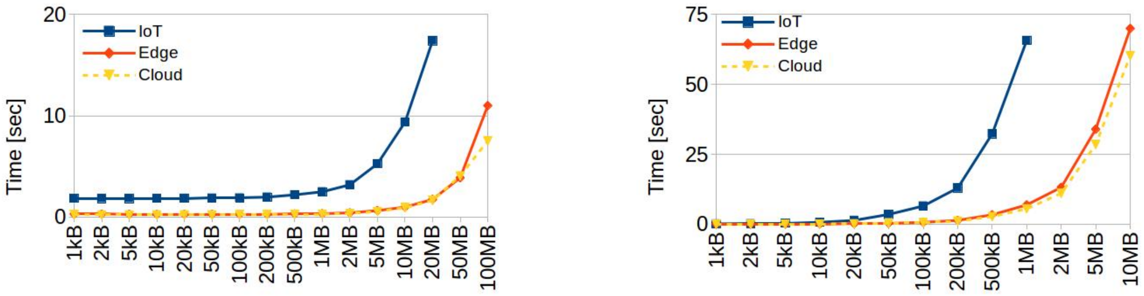
(a) IoT (b) Edge (c) Cloud

Figure 6.12: Execution time of RRNS vs. Shamir merge on IoT, Edge and Cloud devices

All experiments show that the execution time of different tasks (RRNS split, Shamir split, RRNS merge and Shamir merge) heavily depends on the file size, the number of threads and the location of execution. Therefore, to avoid wasting computing resources, it is necessary to identify where each task must be executed.

For RRNS, Figure 6.13(a) hints that execution should be performed on IoT devices for files smaller than 1MB, on Edge devices for files between 1MB and 100MB, and on Clouds for files larger than 100MB.

For Shamir, Figure 6.13(b) hints that execution must be performed on IoT devices for files smaller than 100kB, on Edge for files between 100kB and 500kB, and on Cloud for files larger than 500kB.



(a) RRNS

(b) Shamir

Figure 6.13: Comparison of the execution time of the split in IoT, Edge and Cloud considering a single thread and redundancy equal to 7

We remark that in Figures 6.13(a) and 6.13(b), we fixed the redundancy (*r*) to 7, and the number of parallel threads to 1. Same behaviour can be seen for different values of redundancy and parallel threads. Based on our results, we composed Table 6.3 to hint the most suitable environment for executing split/merge operations.

Table 6.3: The most suitable environment to split/merge

	1k-50k	**100k-500k**	**1M-50M**	**>100M**
RRNS	IoT	IoT	Edge	Cloud
Shamir	IoT	Edge	Cloud	Cloud

6.5 Conclusions and Future Work

In this chapter, we discussed the use of two Secret Share techniques (RRNS and Shamir) for storing files on IoT, Edge and Cloud devices. In order to evaluate these techniques, we made two different types of analyses: storage required and execution time. Experiments have shown that RRNS requires less storage capacity than Shamir. With regards to the execution time, we have seen that Shamir is faster than RRNS for smaller files, however, its behaviour exponential grows for larger files. Therefore, RRNS performs better for larger files. We also observed that the environment on which the computation should be performed depends on both file size and the algorithm. For instance, RRNS can be used for files up to 500kB on IoT, up to 50MB on the Edge, and bigger files on the Cloud. For Shamir, on the other hand, the threshold to move from the IoT to the Edge is 50kB, and from the Edge to the Cloud is 500kB. In future works, we plan to design and implement a scheduler to place 'split' and 'merge' operations based on our finding in this chapter. We will consider the Osmotic Computing paradigm (a system where microservices can migrate from IoT to Edge and Cloud) to deploy our scheduler. We also plan to implement nested RRNS systems in order to further increase its security level, while reducing its computational time and storage requirements.

CHAPTER 7

An innovative Big-Data visualization tool for telemedicine.

With the explosion of Big Data, visualizing statistical data became a challenging topic that has involved many research efforts over the last years. Interpreting Big Data and efficiently showing information for good understanding are hard tasks, especially in healthcare scenarios, where different types of data have to been managed and cross-related. Some models and techniques for health data visualization have been presented in literature. However, they do not satisfy the visualization needs of physicians and medical personnel. In this chapter, we present a new graphical tool for the visualization of health data, that can be easily used for monitoring health status of patients remotely. The tool is very user friendly, and allows physician to quickly understand the current status of a person by looking at coloured circles. From a technical point of view, the proposed solution adopts the geoJSON standard to classify data into different circles.

7.1 Introduction

Nowadays, we are participating to a great diffusion of Internet of Things (IoT) devices in many application domains that have been affected by Information Technology (IT). Statista [104] predicted that the number of connected devices will grow up to about 75 billion in 2025, thanks to its small size and a competitive price. Among these, almost a billion of smart connected devices will be the wearable[105] by 2021. Therefore, as the number of data sources grows, data volume will also grow. Thus, the IT community started talking about a new digital trend known as Big Data. In addition to the characteristics of large data volumes, velocity for fast processing, variety of generated data format and veracity (i.e. consistency), Big Data includes data analysis and data visualization functionalities for extracting insights and understanding data.

Focusing on the data visualization task, its main target [106] is to clearly communicate data or information via visual objects, plots and information graphics, stimulating viewer engagement and attention. Indeed, as reported in [107], data visualization is a human need from origins of life and it is currently used from geographers, engineers, military, economists, statisticians, biology, etc... In this context, the healthcare industry is looking at modern visualization solutions working on Big Data storage, processing, and analytics technologies, as reported by the McKinsey Global Institute in [108]. In particular, telemedicine could really benefiting a lot from new visualization and Big Data technologies. More specifically, telemedicine aims to provide clinical healthcare at distance, helping doctors or medical operators to overcome issues due to distance barriers, such as reaching rural communities [109], and moving elderly [110, 111] or disabled [112] patients. For example, such patients can be constantly monitored in a familiar home environment through the usage of IoT and wearable devices [113], and the large amount of data they generate are made available to doctors and physicians [114]. However, usually, doctors and physicians are not able to understand raw data, and need support in understanding what they represent.

Many scientific works on this topic are available in literature, such as [115, 116] and [117]. However, to address issues related to the huge amount of collected data in telemonitoring scenarios, we need to move towards the problem of storing, analysing and visualizing Big Data. Some works such as [118] and [119] propose solutions for manage efficiently Big Data among sites geographically distributed. More specifically, the visualization of these data have to be smart enough to deal a fast comprehension of disease and dangerous situations for patients.

In this work, we propose an innovative method for visualizing the patients health status according with measurements coming from different data sources, such as wearable sensors, bloody analysis IoT devices and medical instruments. This method is based on data visualization techniques able to treat distant patients monitoring problem. After having selected a treatment for a specific patient, the doctor or the medical operator is able to follow the evolution of his/her care. Moreover, selecting a zone of interest simply drawing a polygon on the map, it is possible to select one of the included markers for visualizing a patient health status by means of a circular view.

The rest of the chapter is organized as follows. In Section 7.2, we present other works related to telemedicine and BigData visualization. In Section 7.3 we focus on our motivations for this work. In Section 7.4, we describe the architecture designed for deploying the visualization service. Two innovative BigData visualization methods are described in Section 7.5; whereas the experiments of our scientific work are presented in the Section 7.6. Finally, the Section 7.7 concludes this work with lights on the future.

7.2 Related Works

Information and communication technologies in Telemedicine are research topic started many years ago. Several studies discussed benefits and drawbacks of Telemedicine [120]. Indeed, it improves access to remote information and services, provides care in rural environment, guarantees quality control of patients' treatments, and reduces health-care costs. On the contrary, it reduces communications both among professionals and between professionals and patients. This raises doubts on the quality of health services and presents organizational and bureaucratic difficulties.

Data Visualization is only a recent aspect associated to Telemedicine, coming from the emergence of Big Data analysis. This has interested different health diseases and topics, as reported in the following.

In [121], the authors focused about analysis and comparison of electromyography (EMG) results on different patients. They proposed a data model to follow the evolution of amyotrophic lateral sclerosis (ALS) by means of Web APIs that interface dynamic and interactive charts.

Three algorithms for processing and analyzing Big electroencephalography data have been proposed in [122]. The first one aims to transform the standard European Data Format (EDF) into the standard JavaScript Object Notation (JSON), in order to facilitate human readability of data, and to compress data for decreasing size and transfer time. The second algorithm is focused on how gathering the compressed files. Finally, the third algorithm performs real-time interactions with signal data by users' perspective. More specifically, single or multiple signal channels visualization through a smartphone were used as example.

Instead, the MyGeneFriends Web platform has been proposed in [123]. It was inspired by social networks for following the evolution of genetic disease. More specifically, the authors organized the social network around three agents (genes, humans and genetic disease), providing data interactions by means of evaluation and visualization of connections among these. Thanks to this platform, the behavior of users generated insights very useful for discovering connections among agents.

In [124], authors proposed A Data Visualization Approach for Supporting Heterogeneous Data. In particular, their aim is to show the relationship among heterogeneous data stored into relational databases, their approach is based on three data structures: i) Graph Structure; ii) Tree Structure and iii) Graph-Tree Structure: the graph is used in order to show the relationship among tables belonging to the same database; trees instead are used in order to represent tables; finally, by means of the combination of the above presented data structures, authors are able to show the relationship among collected data. In [125], authors discussed a visualization tool based on web and mobile platforms for physicians. In particular, they presented 4 visualization tools: i) "Continuous Month" that represents each month like a calendar; the color of each day depends on the average value of the measured parameter; ii) "Continuous Day", that shows per each hour vertical bars representing measured values, the color of the bars depends on the parameter, the height depends on the average measured value; iii) the "Circular Day", by means of a pie chart, that provides a representation of the whole day. The chart is divided into 24 segments, and each radial segment represents a different hour. In this view, is it possible to analyse parameters one by one, and the color of each segment (like in "Continuous Month" view) depends on the means value; iv) "Multi-Circular Day", that is used to analyse the behavior of a chosen parameter over several days.

The authors in [126] proposed a mobile health program for expanding the coverage and improving the effectiveness of healthcare. More specifically, they developed the GeoHealth application, deploying that in a metropolitan Brazil area. The solution includes a database with family health conditions data inside, a Web Browser for running visualization tools, and a smartphone for gathering data and georeferencing the family health data. We aim to generalize such concept of georeferencing, providing a data visualization tool able to track patients health status according with measurements coming from different geolocated sources.

7.3 Motivation

Data visualization is the branch of Big Data analytics that allows end users to analyse, understand and extract insights from data simply looking at its representation. In the field of healtcare, end users include personnel and practitioners of healthcare centers, clinicians and researchers in charge to evaluate the health status of a patient and, in case, take decisions on treatments. Often, these persons are not very confident with information and communication technologies. Thus, they need easy to use software solution, and visualization tools able to provide a clear picture of the patient health status. This is a very important task especially in the telemedicine application scenario, where medical operators can not have a direct contact with their patients, but relays on video communications and/or remote IoT devices for measuring bio-physical parameters. In telemedicine, an operator has the responsibility to evaluate the effective need of a therapy or other heath related issues only considering interviews and data collected at the patient home. To improve the quality of the decision support system for medical operators, many data at the patient home should be collected (generating Big Data), and then opportunely shown in aggregate/statistical way to the medical operators.

Following the above considerations, we aim to provide an easy-to-use visualization tool for by using innovative web and mobile-based interfaces. The proposed solution will provide a dashboard showing a clear picture of Big data related to the patient health, implementing user-centric approach to simplify the digital experience of end users.

The design of our Big Data visualization solution for telemedicine should fulfill the following requirements:

1. highly scalability: capacity to adapt the execution of different services to the effective availability of resources (hardware ad software);
2. ability to manipulate and cross relate both historical data and real time acquisitions;
3. user friendliness in using the new tools;

7.4 Architectural Overview

Starting from requirements described in the section above, the design of the reference architecture of our visualization solution (in Figure 7.1) is based on microservices. Each microservice performs a specific task and can be easily deployed in heterogeneous infrastructures accordingly to the specific needs of applications and users. This make our solution highly scalable in terms of computing resource. About scalability on data management, usually a Database Management that is not able to work efficiently with huge amount of data. For this reason, in our system we adopted as central storage MongoDB, it is a Big Data document oriented database with features of data replication (for improving the availability) and sharding (for improving the scalability).

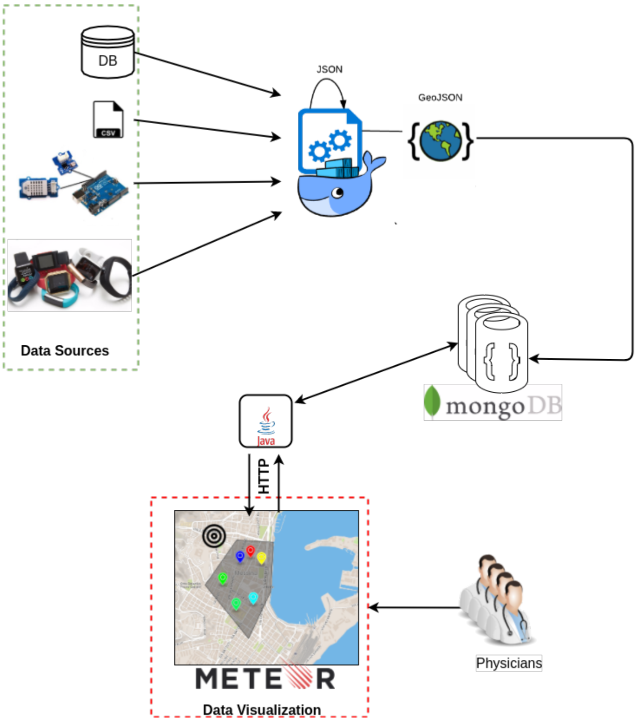


Figure 7.1: Overview of the deployed System

The second requirement satisfied by our system is the ability to manipulate both historical data (stored into DBs or CSV files such as bloody analysis) and real time physical measurement acquisitions coming from wearable sensors or medical devices at home. This requirement is very important and, in order to manage data coming at different locations (e.g. from medical devices, such as the Lokomat in a care center and physical parameters measurements at home), a Docker container microservice needs to convert incoming data by means of specific adapters to GeoJSON, which is a human and machine readable format for encoding geographic data structures. The choice of this format is justified by the widespread use of OpenStreetMap, a collaborative project aimed at creating maps with free content in the world. Furthermore, the format is natively stored in the selected database for geolocalization.

In order to make simple and user friendly the interaction with the system (third requirement), a data visualization layer was deployed using Meteor, a web framework that allows to generate cross-platform code even for Android and iOS. It is able to show data for physicians-defined geographical zones in two modalities: general overview or patient-specific, as we will discuss in Section 7.5.

Finally, in order to interconnect the data visualization layer with backend system, a specific microservice was required. It exposes its functionalities through REST APIs and interacts with the database system thanks to the MongoDB Java drivers.

7.4.1 MongoDB's data structure overview

MongoDB represents the core of our system. Indeed it stores all data. In particular, we created four collections for storing respectively: i) personal data of each user; ii) acquisition coming from different data sources; iii) corrective factor for representing markers and circles; iv) circles produced by physicians. The collection related to personal data contains: name, surname, date of birth, fiscal code, residence address(mapped as GeoJSON), and a list of contacts such as email, mobile and so on. The acquisition collection contains all measurements as raw data represented by the following parameters: value, description, UOM and timestamp. The collection related to the corrective factor contains, for each patient, multiple documents, each of them related to a specific measurement. The structure of the document is the following: patientId, name of the parameter and value. The collection related to circles contains, mapped as GeoJSON, the coordinates both of markers that represent measurement, in aggregate form, and circles produced.

7.5 Flexible visualization

The architecture previously described allows us to develop an innovative approach for data visualization with both general overview and patient-specific observations.

Considering a physician that want to monitor if his/her patients are following the prescribed treatment in terms of rehabilitative activities or medicines consumption, he/she can search some patients by filtering data with personal information or specifying the geographical zone. In the second case, the physician is able to select a zone of interest simply drawing a polygon on the map that is shown on our Meteor application. Then, the system sends the coordinates of the polygon to the backend of our system, by means of a POST request. There, a specific microservice creates the GeoJSON query and makes the request on MongoDB, that returns back to the microservice all the GeoJSON acquisitions related to the specific area that match physicians' requests. Starting from retrieved values, the backend of our system creates markers and sends them to Meteor application. At the end of this process, the markers appear in the map, as shown in the Figure 7.2.

Considering that the position of each marker depends on the real residence address of the patient, the symbol into the marker is specific for each requested parameter and the color of each marker depends on the value of the acquisition. Figure 7.2 shows the patients' activity, in terms of daily steps through a stroke circle. Moreover, in order to make our system more user friendly, we introduced the innovative color scale that is shown in the Figure 7.3 for

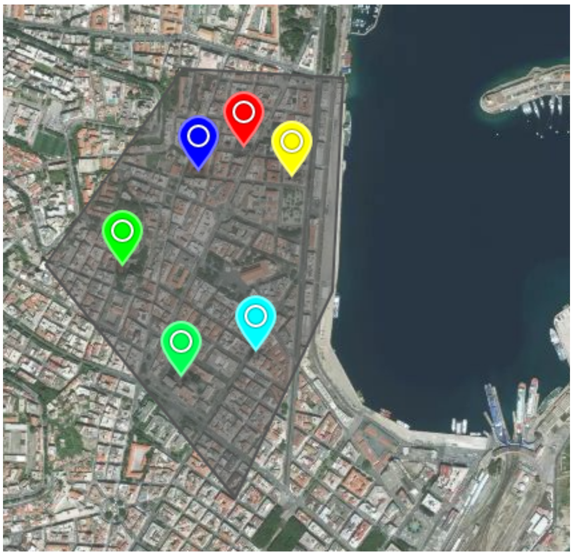


Figure 7.2: General Overview visualization mode.

filling markers. In particular, we divided the scale in three zone respectively:

1. **Hypo zone.** Zone in which parameters are too low.
2. **Good zone.** Zone in which parameters are good, the mean value represents the optimal value.
3. **Hyper zone.** Zone in which parameters are too high.

With this approach, for physicians is very simple to know which patients are following treatments in recommended manner or not simply looking at colors. Indeed, when the measured parameter is equal to 0, the associate color is blue; when it is equal to the optimal recommended value, the color is green; when the parameter is equal to 2 × *optimal value* the associated color will be red. Of course, each intermediate value has a different color associated.

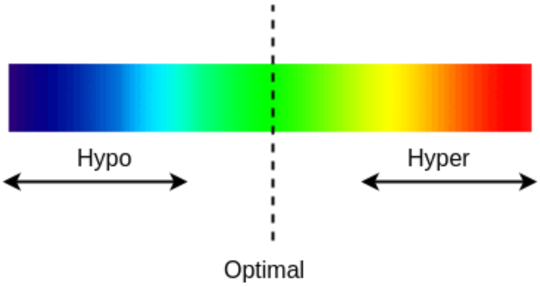


Figure 7.3: Colors scale with reference to defined zones.

Unfortunately, it is not possible to make direct associations among measured parameter and colors. For this reason, we introduced a normalized parameter by means of the following formula:

$$normalized\_param = \frac{measured\_param \times color\_range}{param\_range}; \tag{7.5.1}$$

where param_range is equal to:

$$param\_range = 2 \times optimal\_value; \tag{7.5.2}$$

and color_range is equal to:

$$color\_range = lastcolor - firstcolor; \tag{7.5.3}$$

In order to represent colors, we adopted the r,g,b notation. We considered 1020 colors from blue (0,0,255) to red (255,0,0).

Table 7.1: R,G,B component behavior

Interval	Red	Green	Blue
0-255	=	+	=
256-510	=	=	-
511-765	+	=	=
766-1020	=	-	=

In order to make association among colors and r,g,b components, we used the normalized parameter calculated above. In particular, we noticed that the behavior of r,g,b components depends on the normalized parameter, Table 7.1 resumes the behavior. As we can observe, we divided the color range into four sub intervals:

1. $[0 - 255]$. In this interval, the first color is blue (0,0,255), the last color is cyan (0,255,255), red and blue components are fixed respectively to 0 and 255, green increase linearly in according to the mathematic law:

$$y(x) = x \tag{7.5.4}$$

Equation 7.5.5 resumes the behavior of r,g,b components:

$$\begin{cases} r = 0; \\ g = normalized\_param; \\ b = 255; \end{cases} \tag{7.5.5}$$

2. $[256 - 510]$. In this interval, the first color is cyan (0,255,254), the last color is green (0,255,0), red and green components are fixed respectively to 0 and 255, blue decrease linearly in according to the mathematic law:

$$y(x) = 510 - x \tag{7.5.6}$$

Equation 7.5.7 resumes the behavior of r,g,b components:

$$\begin{cases} r = 0; \\ g = 255; \\ b = 510 - normalized\_param; \end{cases} \tag{7.5.7}$$

3. $[511 - 765]$. In this interval, the first color is green (1,255,0), the last color is yellow (255,255,0). Green and blue components are fixed respectively to 255 and 0, red increase linearly in according to the mathematic law:

$$y(x) = x - 510 \tag{7.5.8}$$

Equation 7.5.9 resumes the behavior of r,g,b components:

$$\begin{cases} r = normalized\_param - 510; \\ g = 255; \\ b = 0; \end{cases} \tag{7.5.9}$$

4. $[766 - 1020]$. In this interval, the first color is yellow (255,254,0), the last color is red (255,0,0). Red and blue components are fixed respectively to 255 and 0, green decrease

linearly in according to the mathematic law:

$$y(x) = 1020 - x \tag{7.5.10}$$

Equation 7.5.11 resumes the behavior of r,g,b components:

$$\begin{cases} r = 255; \\ g = 1020 - normalized\_param; \\ b = 0; \end{cases} \tag{7.5.11}$$

However, the general overview does not show a clear view of each patient. Indeed, it is not able to represent the health state. For this reason we introduced a patient-specific visualization that consists in multiple concentric circles with specific colors. Figure 7.4 shows an example. In this visualization mode we maintained fixed to three the number of zones. Instead, physicians can decide the number of sub levels per each zone and the zoom level, color are calculated by means of formulas above described. We remark that each calculated circle is stored into the system database.

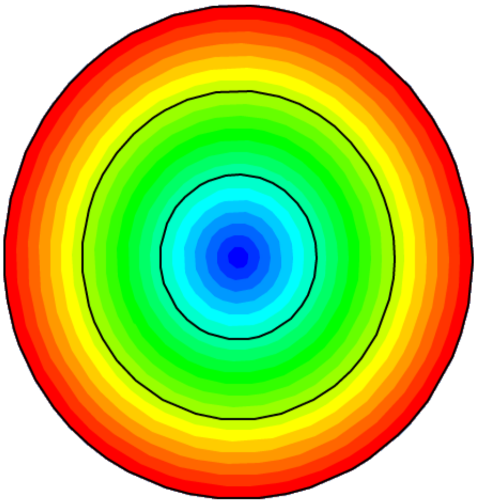


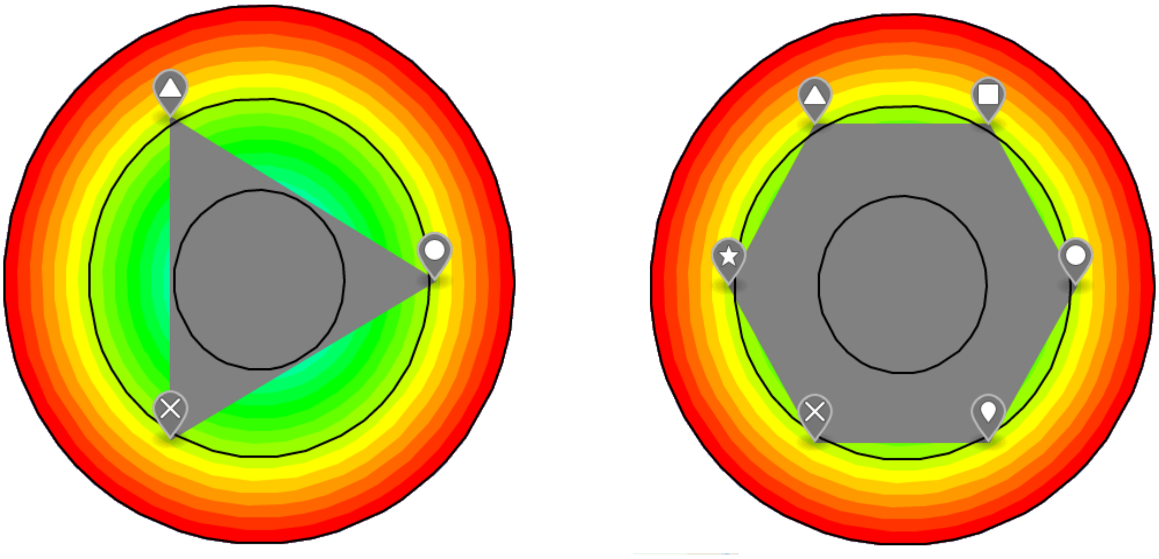
Figure 7.4: Skeleton of the Patient specific visualization

The system database, MongoDB, is very powerful with GeoJSON, indeed it allows to make queries on hundreds thousand of documents very fast. For this reason we decided to adopt this standard for the representation of circles.

Due to the fact that GeoJSON does not allow to draw circles simply using radius and coordinates of the center, in our application, starting from the equation explained in 7.5.12, we created a specific microservice able to calculate coordinates of circles.

$$\begin{aligned} x &= c_x + radius \times \cos\alpha; \\ y &= c_y + radius \times \sin\alpha; \end{aligned} \tag{7.5.12}$$

Physicians, in order to see this kind of visualization in our software, have to click on the patient that would like to investigate and to select parameters of interest. The system, after a request to the backend will create the chart. Figures 7.5(a) and 7.5(b) show circles with three and six parameters respectively.



(a) Three parameters

(b) Six parameter

Figure 7.5: Patient specific visualization.

Each parameter, represented by a specific marker, is referring to a specific test such as oxygenation of the blood, heartbeats etc. Due to the fact that different parameters have different range, in order to represent markers in the same charts, we introduced a corrective factor for both dimensions that depend on the parameter itself even of the patient. The new equation for calculating the coordinates of markers are shown in listing 7.5.13.

$$\begin{aligned} p_x &= c_x + \frac{measured\_param \times \cos\alpha}{factor_x}; \\ p_y &= c_y + \frac{measured\_param \times \sin\alpha}{factor_y}; \end{aligned} \tag{7.5.13}$$

7.6 Experimental Results

In order to validate our method, we made specific analysis for both the visualization techniques we presented. Our analysis is based on experimental measurements (time performances) and on comments of physicians that acted as beta users (comments were collected by means of a survey).

Our testbed is composed of two workstations, one that acts as central storage in which we installed MongoDB v.3.4 in standalone configuration and another one for computation.

Hardware/software configuration of the **storage workstation** is the following: 2 CPU Intel(R) Core(TM) Xeon E7-8860V3 CPU @ 3.2 GHz with 2 cores and 2 threads, RAM 4GB, GFLOPS 39, OS: Ubuntu server 16.04 LTS 64 BIT. Bandwidth: uplink 76.78 Mbps, downlink 90.18 Mbps.

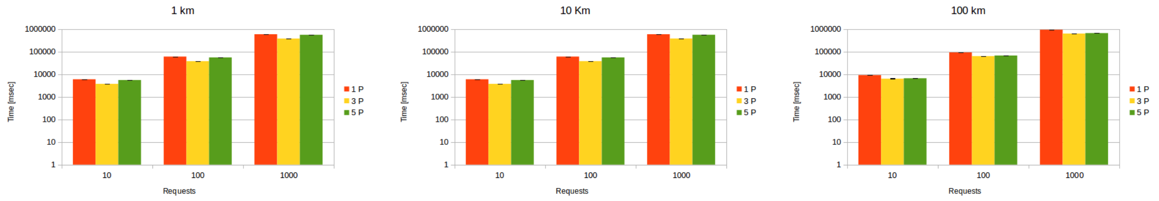
Computation workstation in which is running the module that calculates circles has the following characteristics: CPU Intel(R) Core(TM) i5-5200U CPU @ 2.8 GHz with 2 cores and 2 threads, RAM 8GB, GFLOPS 66, OS: Ubuntu server 16.04 LTS 64 BIT. Bandwidth: uplink 0.68 Mbps, downlink 12.73 Mbps.

Experiments were repeated 30 times, in order to consider confidence intervals, at 95% and mean values.

Unfortunately, we did not found any similar solution to compare performances of systems. We made specific analysis for each visualization technique.

7.6.1 General Overview

Time performances



(a) radius = 1 kilometer (b) radius = 10 kilometer (c) radius = 100 kilometer

Figure 7.6: Time performance of the general overview considering different dimension concentric radius and 1, 3 and 5 parameters.

In order to analyse performance of the general overview feature, we inserted into MongoDB collection of 400k on random patients. We conducted our analysis in different scenarios.

More specifically, we realized three different scenarios varying the geographic shape of interest (we considered increasing concentric circles with different radius, starting from 1 kilometer up to 100 kilometers), in three different configurations (varying the number of requested patient's parameters). In order to perform scalability analysis, we made 10, 100 and 1000 subsequent requests. Figures 7.6(a), 7.6(b), 7.6(c), show the behavior of the deployed system. Time performances of three presented scenario are very close. As we expect, time performances increase linearly with the number of subsequent requests that we performed.

Requests with a single patient's parameter are the simplest but time performance are slower. This behavior is due to the huge amount of data that flows from the database system to the physician. Requests with five patient's parameters have intermediate performance, indeed they present the more complex query but return back less results. From a numerical point of view, the better trade off is implemented by the request with three parameters, indeed it presents computation time lower than other scenarios.

Physicians' comment

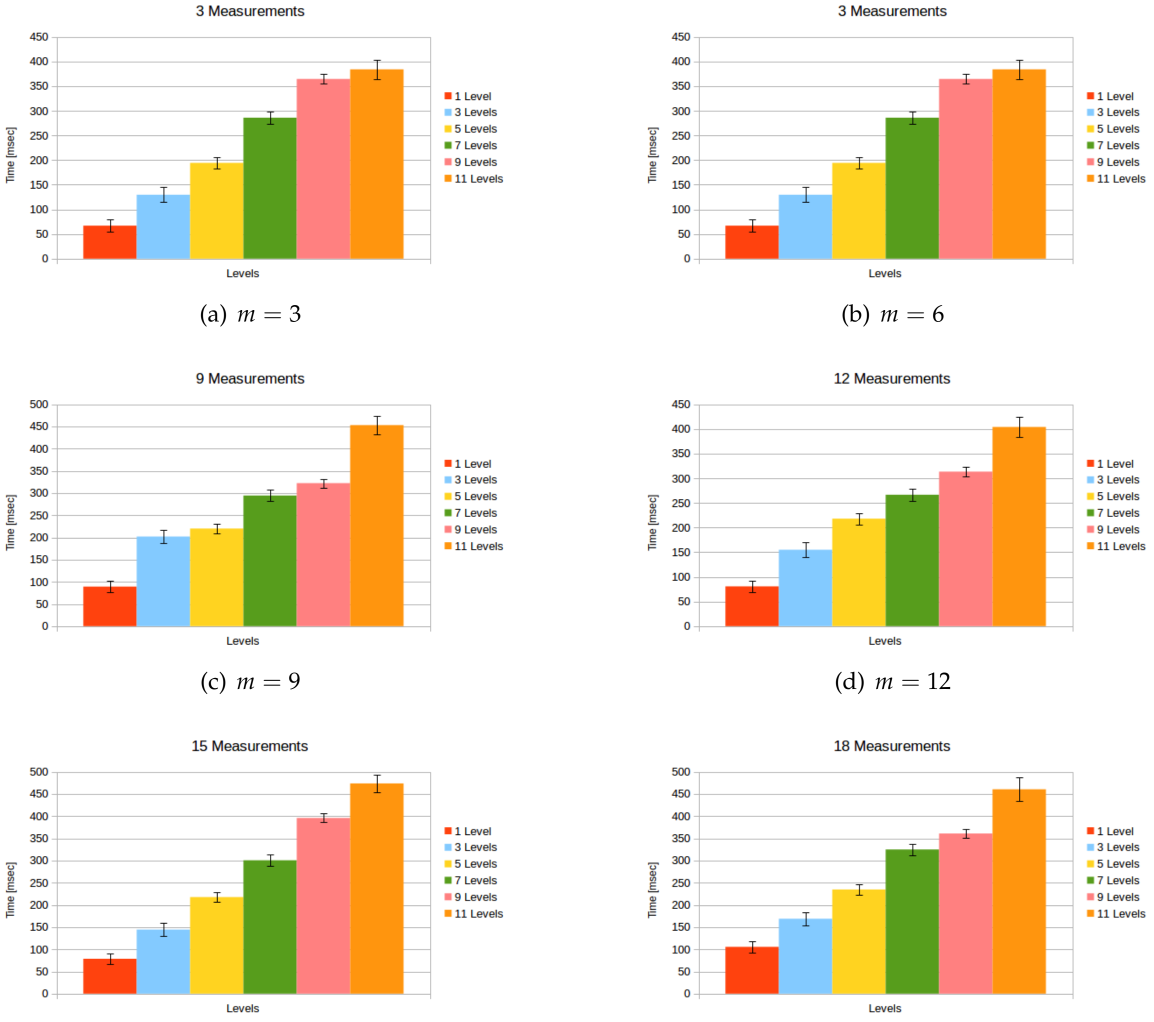
In order to collect comments for the general overview function we conducted a survey to physicians that act as our beta tester of the system. In particular, we asked for three aspects: i) user-friendliness; ii) time performance; iii) improvement.

The comment of the users was: i) the system is very simple to be used, markers are well understandable even the colors representation describe very well the user status; ii) time performances are good; iii) in order to improve the system users suggest to add more shapes for drawing polygon.

7.6.2 Patient Specific Observation

In order to analyse the Patient Specific Observation performance, we made different experiments varying both number of levels that compose each zone and quantity of measured parameters that we showed in the chart. In particular, we considered six different configurations (3, 6, 9, 12, 15 and 18 measured parameters) in six different scenarios: 1, 3, 5, 7, 9 and 11 levels per each zone.

Time performances



(a) $m = 3$

(b) $m = 6$

(c) $m = 9$

(d) $m = 12$

(e) $m = 15$

(f) $m = 18$

Figure 7.7: Time performance of the patient specific observation considering different measurements (m).

In the charts we presented the execution time expressed in milliseconds on the y-axis, and he number of levels on the x-axis.

Figure 7.7(a) shows the time performance in order to create circles with three measured parameters. As we expect, execution times increase with the increasing of the number of levels. Execution time varies approximately form 70 to 400 milliseconds.

Figures 7.7(b), 7.7(c) and 7.7(d) show the behavior of our system in configurations respectively with 6, 9 and 12 measurements. Time performances are very similar to the previous experiment.

Figures 7.7(e) and 7.7(f) show the behavior of our system when 15 and 18 measurement are displayed respectively. From a numerical point of view the behavior was very similar respect to cases previously presented.

Physicians' comment

As a well as for the general overview, also for the specific overview we conducted the same survey to beta tester user.

With reference to the case represented in Figure 7.7(a) beta users commented that time performance was good in all presented configurations, even the number of measurement was adequate. However they noticed that configurations with 1, 3 and 5 levels presented too low resolution.

With reference to cases represented in Figures 7.7(b), 7.7(c) and 7.7(d) physicians commented that the number of measurement displayed on the chart was good, indeed each parameter was well displayed without any overlapping.

With reference to cases represented in Figures 7.7(e) and 7.7(f) for the physicians' point of view the experience was very similar to already presented cases, but, in order to see measurements without any marker overlap they had to use the zoom function to make bigger circles.

The execution times, as well as the user experience in all presented configurations were very similar. In order to use this tool even in small devices such as tablets and smartphones, the best solution is represented by the configurations with less than 12 measurements.

7.7 Conclusion and future works

In this scientific work, we focused about the Data Visualization aspect among the Big Data steps. More specifically, we investigated a data visualization technique for monitor health status of patients. First of all, we presented the architecture designed and deployed for running the visualization services.

From the features point of view, the physicians can monitor treatment of patients drawing a polygon on a showed map, visualizing the markers included in that specific area. Thus, each marker can run a circular view that shows different parameters in order to monitor the status of the health of patients. The experiments performed for different scenarios highlighted great performance of both visualization mode in terms of time as a well as user experience.

In future works, for each measurement, we plan to implement a non-linear regression algorithm able to to quantify the missing value with high precision. Furthermore, the development of a Machine Learning techniques would help us to create a model capable of predicting the relative measurement values in order to send alarms to physicians when risky conditions occur.

CHAPTER 8

An innovative way for managing Electronic Health Records (EHRs) and avoid mistakes

In last years, achievements in Information and Communication Technologies (ICTs), such as the Electronic Health Record (EHR), have been improved the health care system. However to be effective, paramedics and doctors have to consult the most recent version of EHRs anytime and anywhere. A possible solution is to store EHRs on remote storage services. However the EU General Data Protection Regulation (GDPR) does not allow to store plain files containing personal data in services accessible remotely. To solve this challenge, a possible solution is to use Edge computing devices running Secret Shares algorithms to split and merge EHRs on demand, however these techniques have not been evaluated before for these purposes. To address this issue, in this work we analyse the Redundant Residue Number System (RRNS). In particular, considering different EHR sizes (from 10kB to 1MB), we evaluated computation time (split and recomposition), transfer time (upload and download) from/to public Cloud storage providers (Google Drive, Mega and Dropbox) and storage requirement. Results showed that, in configuration with 7 level of redundancy, the RRNS uses only 50% of the storage required for the simple file replication. We also discovered that Google Drive, due to synchronization overhead, is slower than other Cloud service providers for the upload of chunks but faster for the download.

8.1 Introduction

Nowadays computer assisted tools are becoming much more popular in all fields from cities to farm, to industries etc. Even hospitals, in last years are interested from this phenomena. That is, old medical devices are replaced by new computer assisted ones or new achievements in Information and Communication Technologies (ICTs) are adopted. Among these the Electronic Health Record (EHR) is becoming much more popular. It contains the whole clinical life of patients. Therefore, if manged properly, it can help to reduce clinical errors inside and outside the hospital. The management of EHRs is a very hard task, because paramedics and doctors have to consult the most recent version of EHRs anytime and anywhere. A possible solution is to store EHRs on remote public or private Cloud storage services. However they can arise several privacy and security threats such as:

1. hackers could violate the privacy of people by attacking these servers. In fact, this has been a real concern during the last years; for example, Google revealed that more than 1 million of Google Accounts were hacked by Gooligan in three months [82]; Yahoo disclosed that hackers beached between 1 billion and 1.5 billions accounts [83]; and, Dropbox denounced that more than 68 million account details were disclosed [84].

2. Cloud Service Providers can be discontinued as it happened to Copy-Cloud [85], Ubuntu One [86], DELL Data Safe [87], etc.

3. hospitals can be affected by the vendor lock-in, and thus their data cannot be easily migrated among Cloud Service providers [88].

4. Servers can be affected by the ransomware (e.g., WannaCry [89]) and become the subject of ransom payments. In fact, during the last years several cases have been announced; for example, in UK roughly 200000 computer have been affected costing for the National Health Service (NHS) about 92 millions of pounds [127].

Furthermore, the new EU General Data Protection Regulation (GDPR)[128] came into force on 25 May 2018. It does not allow that plain files containing personal information can be stored in public or private services accessible remotely. A possible solution is to use Edge computing devices for splitting and add redundancy to EHRs. In literature several approaches have been proposed, among them algorithms belonging to Secret Shares family fit better these requirements. However, these techniques have not been evaluated before for splitting EHRs. To address this issue, in this work we analyse the Redundant Residue Number System

(RRNS). In particular, considering different EHR sizes (from 10kB to 1MB), we evaluated computation time (split and recomposition), transfer time (upload and download) from/to public Cloud storage providers (Google Drive, Mega and Dropbox) and storage requirement. Results showed that, in configuration with 7 level of redundancy, the RRNS uses only 50% of the storage required for the simple file replication. We also discovered that Google Drive, due to synchronization overhead, is slower than other Cloud service providers for uploading chunks of EHR but faster in downloading.

The chapter is organized as follows: Section 8.2 analyses the state of the art. In Section 8.3 we discuss the EHR. Motivation and Use-Cases are described in Section 8.4. The proposed approach is discussed in Section 8.5. The performance assessment of the proposed approach is discussed in Section 8.6. Open issues and challenges are described in Section 8.7. Section 8.8 concludes the work and provides lights of the future.

8.2 State of the Art

Edge computing is a very hot topic within the scientific community. Recently, the advent of Internet of Things and industry 4.0 is increasing the interest on such a topic. However, currently there are not so many works available in literature focusing on Edge-based services for hospitals.

In [129], authors present RASPRO (Rapid Active Summarization for effective PROgnosis) an IoT-Edge based system for telemedicine. The proposed system is composed of wearable sensors that send data to RASPRO which summaries row data into severity symbols called PHMs (Personalized Health Motifs).

A similar approach is discussed in [6]. Author presented a telemedicine visualization tool for patients' clinical Big Data. The system is composed of Edge devices that acquire and transmit data the Cloud engine that create markers in colored circles representing the health status of patients.

In [130], authors proposed an architecture based on Narrowband IoT devices to interconnect objects inside ed outside hospitals. The proposed system, by using Edge devices, is able to fulfill latency requirements of medical processes.

In [131], authors focused their attention on DICOM (Digital Imaging and Communications in Medicine), a standard for storing and managing medical imaging. They proposed a hybrid model for Cloud-based HIS focusing about public and private Cloud. The last one was able to manage and store computer-based patients records and other important information, while public Cloud was able to handle management and business data. In order to share information among different hospitals, authors adopted VPN (Virtual Private Network).

In [132], authors analysed the possibility to adopt the electronic Patient Clinical Record (ePCR) in pre-hospital emergencies. In particular, they survey different UK ambulance services. More than 50% of them 7 over 13 were using ePCR. Instead 4 of the 6 remaining companies in the past adopted the ePCR but reverted to the paper records.

In [133], authors proposed an Edge Computing based gateway for the interoperability of health-care systems. In particular, they mapped heterogeneous and geographically distributed medical centers and pharmacies as nodes of their system. Each node is able to interact with others by means of restful APIs.

In [134], authors analysed the indoor localization problem for visitors of hospitals. In order to solve this challenge, they proposed a navigation and localization system based on low cost Edge devices. The devices composing the system, by using Bluetooth Low Energy and Ultra-wideband(UWB) sensors are able to identify the position of visitors and to send them back localization information.

In [135], author discuss about the placement problem in the era of single board computers. In particular, they analysed two use cases related to smart train and smart hospital respectively. They proposed solutions that rely on Edge devices. However, the proposed solutions are suitable for small and medium environments (such as a train or a small hospital). Considering a big hospital with high number of rooms the proposed approach does not scale properly, therefore it is necessary to realize a hybrid approach based on Edge and Cloud devices.

8.3 The Electronic Health Record

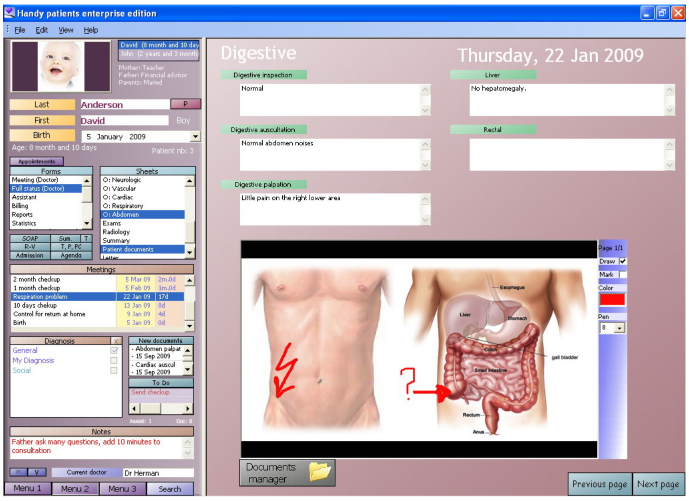
The Electronic Health Record (EHR), encapsulates and represents patients life in a digital way. It can be seen as an extended collection of Electronic Medical Records (EMR, that we will discuss in Section 8.3.1), indeed it could include the following types of data:

- medical history;
- demographics;
- allergies;
- laboratory test results;

- radiology images;
- medication;
- personal statistics such as the weight and the age;
- immunization status;
- vital signs;
- billing information.

Furthermore, the EHR allows to create specific private views managed by patients: the Electronic Patient Record, that we will discuss in the Section 8.3.2. The EHR is not a new concept, indeed, some works such as [136] in the early 2000s discussed about the need of a National Electronic Health Record Architecture. In particular, authors described the Australian and the American models. The Australian model, called HealthConnect, extracts and aggregates patients' medical data in order to create centralized HealthConnect records. These records can be shared among authorized actors. The HealthConnect is based on a "push" approach, this means that patients decide which part of the health record can be pushed to the HealthConnect record. In USA, instead, local governments launched several local initiatives. The result was a proliferation of different EHR. In order to harmonize EHR information, an US national model have been developed. This model is based on a "pull" approach, this means that semantically similar data are imported from local EHRs into the national one. In the last years, each European country developed its own EHR, based on local rules and laws, the objective is to have a centralized European health record system before the 2020. The EHR from a clinical point of view is very useful, indeed, it allows to have a clear vision of each treatment that the patient followed, allergies and medicines that s/he took. As the reader can see, in Figure 8.1 is shown an example of EHR. It can be divided in two parts: on the left side personal and historical data such as appointments, diagnoses and meetings are shown, on the right part is shown an EMR.

Furthermore, the EHR allows to reduce duplication of effort and clinical errors. By means of HealthConnect, the Australian government estimated to save AUD $300 million per year [137]. However, EHR presented security issues in terms of data privacy protection that blocked its proliferation. In Europe, in order to protect the processing of personal data, including that for purposes of health care, the European Parliament in 2016 approved the General Data Protection Regulation (GDPR) [128].



David (8 month and 10 day
John (2 years and 3 month)
Mother: Teacher
Father: Financial advisor
Parents: Maried

Last: Anderson P
First: David Boy
Birth: 5 January 2009
Age: 8 month and 10 days
Patient nb: 3

Forms: Meeting (Doctor), Full status (Doctor), Assistant, Billing, Reports, Statistics

Sheets: O: Neurologic, O: Vascular, O: Cardiac, O: Respiratory, O: Abdomen, Exams, Radiology, Summary, Patient documents, Letter

Meetings

2 month checkup	5 Mar 09	2m.0d
1 month checkup	5 Feb 09	1m.0d
Respiration problem	22 Jan 09	17d
10 days chekup	13 Jan 09	8d
Control for return at home	9 Jan 09	4d
Birth	5 Jan 09	0d

Diagnosis: General ☑, My Diagnosis □, Social □

New documents: - Abdomen palpat - 15 Sep 2009; - Cardiac auscul - 15 Sep 2009

To Do: Send checkup

Assist: 1 Doc: 0

Notes: Father ask many questions, add 10 minutes to consultation

Current doctor: Dr Herman

Digestive

Thursday, 22 Jan 2009

Digestive inspection: Normal

Liver: No hepatomegaly.

Digestive auscultation: Normal abdomen noises

Rectal:

Digestive palpation: Little pain on the right lower area

Figure 8.1: Electronic Health Record
Source: Oguntoye patients electronic medical record (free open source version)

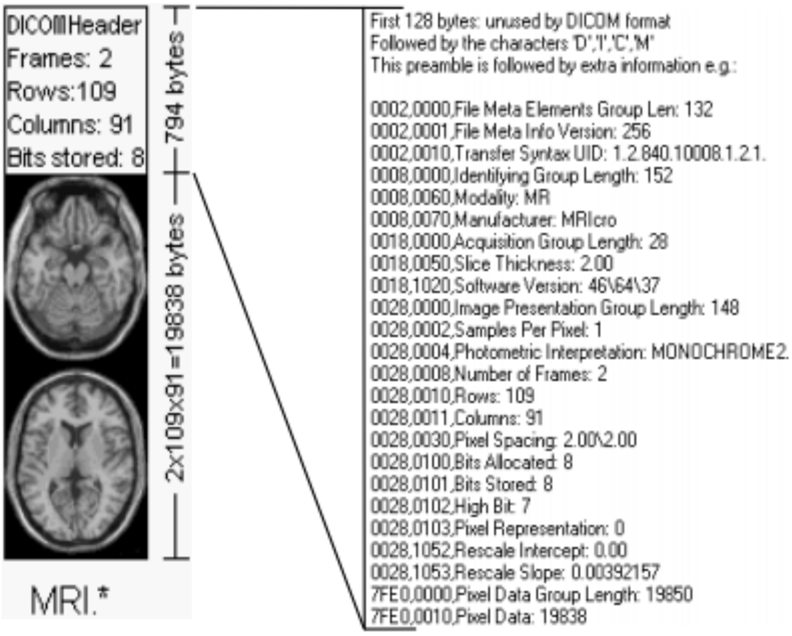
A well known family of standards for exchanging EHR is the Health Level Seven (HL7) [138]. It define how information must be packaged and transmitted among parties. The HL7 provides seven reference categories:

1. **Primary.** Standards for compliance and inter-operability;
2. **Foundational.** Standards that define infrastructure and fundamental blocks;
3. **Clinical and Administrative Domains.** Standards that define messaging and document.
4. **EHR Profiles.** Standard for the definition of models and profiles of the EHR.
5. **Implementation Guides.** Supplemental material of parent standards. This section contains the documentation for the interaction with existing standards.
6. **Rules and References.** This section contains developer manuals and technical guides.
7. **Education & Awareness.** Projects and Standards for Trial Use.

8.3.1 Electronic Medical Record

The Electronic Medical Record (EMR), instead, represents the digital version of the clinical paper documents. It allows to eliminate handwritten medical records, reducing interpretation errors. The EMR can contain: notes, treatments, medical tests and diagnoses.

In Figure 8.2, an example of EMR is shown. In particular is shown a DICOM. DICOM, Digital Imaging and COmmunications in Medicine, is the standard used in medicine in order to represent medical images. The extension of DICOM files is dcm. It is composed of the header that contains metadata (such as personal patient information, software version, number of frames etc.) and the image dataset acquired as single image, cine loop or multiple frames of a study.



```
First 128 bytes: unused by DICOM format
Followed by the characters 'D','I','C','M'
This preamble is followed by extra information e.g.:

0002,0000,File Meta Elements Group Len: 132
0002,0001,File Meta Info Version: 256
0002,0010,Transfer Syntax UID: 1.2.840.10008.1.2.1.
0008,0000,Identifying Group Length: 152
0008,0060,Modality: MR
0008,0070,Manufacturer: MRIcro
0018,0000,Acquisition Group Length: 28
0018,0050,Slice Thickness: 2.00
0018,1020,Software Version: 46\64\37
0028,0000,Image Presentation Group Length: 148
0028,0002,Samples Per Pixel: 1
0028,0004,Photometric Interpretation: MONOCHROME2.
0028,0008,Number of Frames: 2
0028,0010,Rows: 109
0028,0011,Columns: 91
0028,0030,Pixel Spacing: 2.00\2.00
0028,0100,Bits Allocated: 8
0028,0101,Bits Stored: 8
0028,0102,High Bit: 7
0028,0103,Pixel Representation: 0
0028,1052,Rescale Intercept: 0.00
0028,1053,Rescale Slope: 0.00392157
7FE0,0000,Pixel Data Group Length: 19850
7FE0,0010,Pixel Data: 19838
```

Figure 8.2: DICOM,
Source: Image segmentation methods and edge detection: An application to knee joint articular cartilage edge detection - Scientific Figure on ResearchGate.

8.3.2 Electronic Patient Record

The Electronic Patient Record (EPR), like the EHR, contains information about medical history, demographics, allergies etc. The difference between EHR and EPR is in the management of the medical records. Indeed, in the first case are managed by hospital employees such as doctors, practitioners and nurses, instead, the EPR is managed by the patient that can decide which kind his/her health information have to stay in a private zone [139].

Patients' access and management rights on EPRs, depend on the country legislations. Authors, in [140], considering ten countries (United States, New Zealand, Australia, Netherlands, France, Estonia, Sweden, Denmark, Norway and Finland) all around the world and made a comparison in terms of: login procedures, management of the underage's EPR and available datasets. In particular, they considered two kinds of regulations: "soft" and "hard". Soft laws refer to voluntary rules, instead hard to mandatory ones. The study revealed that access and management rights depend on the country in which patients live. Furthermore, they highlight the absence of a cross-national system that can give the same access and management rights in order to solve common problems.

8.4 Motivation and Use Case definition

As we discussed in previous Sections the EHR can help to reduce clinical errors inside and outside the hospital.

For example, considering the first aid in the emergency room of the hospital or in ambulance, knowing the patient's allergies can avoid the administration of wrong drugs. Instead, considering a patient hospitalized, an exchange of medical records could lead to unnecessary surgery [141] or at the patient's death [142] and [143].

Considering these as use cases, in this Chapter we define a possible system based on Edge devices able to manage efficiently the EHR.

8.4.1 UC 1: Hospitalized Patients

This Use Case is focused on hospitalized patients. In Italy, in last years, the rate of hospitalization was 132,5 out of 1000 citizens [144]. As we discussed in Section 8.4, exchange medical records of two patients cloud be very dangerous. Also in this case, the use of the EHR can help to avoid such problems. In particular, we propose a system that allows patients to be moved among ambulatories with their EHR. The proposed system, exploits the same technologies of the use case 2. That is, an Edge device, installed on the bed of the patient, can split and recompose EHRs stored as chunks on the public Cloud.

8.4.2 UC 2: Smart Ambulances

This Use Case is focused on first aid done by ambulances. In Italy, in the last years, roughly the 47% of medical emergency calls required the intervention of ambulances [145]. During the first aid, one of the possible medical problems concerns the administration of drugs to which the patient is allergic. The use of the EHR can avoid this because it contains allergies and the whole clinical history of the patient. However, to be effective, the last recent version of EHRs has to be accessible from ambulances by nurses and paramedics. Storing EHRs on local Hard Disk Drives (HDDs) or Network Attached Storages (NASs) installed on ambulances could make difficult the synchronization among ambulances and hospitals. A possible solution is represented by the use of remote Cloud storage services for storing EHRs and Edge devices installed on ambulances for retrieving and showing EHRs to practitioners.

8.5 The Edge based Hospital Information System

In this Section, we discuss about the design of the proposed system. The enabling technologies are Edge devices. That is, the computation of split and recomposition algorithms is done by these devices. Figure 8.3 show an high level representation of the architectural diagram. As the reader can observe, for both use cases the architecture is the same. It is composed of:

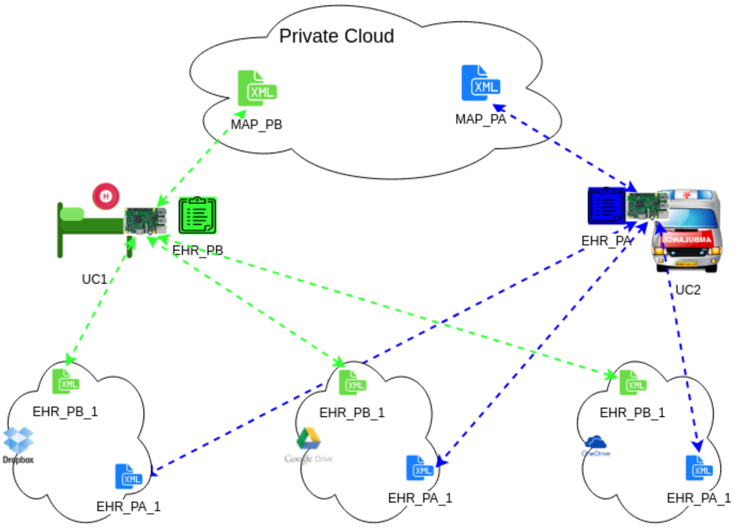


Figure 8.3: Overall Architecture.

1. the private Cloud storage service of the hospital for storing "map files" (special files containing information about the location of chunks);
2. public Cloud Services for storing chunks of EHRs;
3. Edge devices running the RRNS installed on the bed (UC 1) or ambulances (UC 2).

8.5.1 The Redundant Residue Number System

The Redundant Residue Number System (RRNS) is a cryptographic technique belonging to Secret Share algorithms. The basic idea of Secret Share techniques is to split a secret in chunks called shares and recompose it by using all chunks, (n, n) schema, or a subset, (k, n) schema. The Redundant Residue Number System (RRNS) is based on the Residue Number System (RNS). RNS choose p prime, positive integers $m_1, m_2, \cdots, m_p$ called *moduli* such as $M = \prod_{i=1}^{p} m_i$ and $m_i > m_{i-1}$ for each $i \in [2, p]$. Given $W \geq 0$, we can define $w_i = W$ mod m_i the residue of W modulus m_i. The p-tuple $(w_1, w_2, \cdots, w_p)$ is named the *Residue Representation* of W with the given moduli and each tuple element w_i is known as the i^{th}

residue digit of the representation. For every p-tuple $(w_1, w_2, \cdots, w_p)$, the corresponding W can be reconstructed by means of the Chinese Remainder Theorem:

$$W = \left(\sum_{i=1}^{p} w_i \frac{M}{m_i} b_i \right) \mod M \tag{8.5.1}$$

where b_i, $i \in [1, p]$ is such that $\left(b_i \frac{M}{m_i} \right) \mod m_i = 1$ (i.e. the multiplicative inverse of $\frac{M}{m_i}$ modulus m_i). We call *Residue Number System (RNS)*, with residue moduli $m_1, m_2, \cdots, m_p$, the number system representing integers in $[0, M)$ through the p-tuple $(w_1, w_2, \cdots, w_p)$. Considering $p + r$ modules $m_1, \cdots, m_p, m_{p+1}, \cdots, m_{p+r}$ we have:

$$M = \prod_{i=1}^{p} m_i \tag{8.5.2}$$

and

$$M_R = \prod_{i=p+1}^{r} m_i \tag{8.5.3}$$

without loss of generality $m_i > m_{i-1}$ for each $i \in [2, p + r]$. We define *Redundant Residue Number System (RRNS)* of moduli $m_1, \cdots, m_{p+r}$, range M and redundancy M_R, the number system representing integers in $[0, M)$ by means of the $(p + r)$-tuple of their residues modulus $m_1, \cdots, m_{p+r}$. Although the above mentioned *RRNS* can provide representations to all integers in the range $[0, M \cdot M_R)$, the legitimate range of representation is limited to $[0, M)$, and the corresponding $(p + r)$-tuples are called *legitimate*. Integers in $[M, M \cdot M_R)$ together with the corresponding $(p + r)$-tuples are instead called *illegitimate*. Let now consider an *RRNS* whose range is M and redundancy M_R, where $(m_1, m_2, \cdots, m_p, m_{p+1}, \cdots, m_{p+r})$ is the $(p + r)$-tuple of moduli and $(w_1, w_2, \cdots, w_p, w_{p+1}, \cdots, w_{p+r})$ is the legitimate representation on a W integer in $[0, M)$. If an event making unavailable d arbitrary digits in the representation occurs, we have two new sets of elements $\{w'_1, w'_2, \cdots, w'_{p+r-d}\} \subseteq \{w_1, \cdots, w_{p+r}\}$ with the corresponding moduli $\{m'_1, m'_2, \cdots, m'_{p+r-d}\} \subseteq \{m_1, \cdots, m_{p+r}\}$. This status is also known as *erasures* of multiplicity d. If the condition $d \leq r$ in true, the *RNS* of modules $\{m'_1, m'_2, \cdots, m'_{p+r-d}\}$ has range:

$$M' = \prod_{i=1}^{p+r-d} m'_i \leq M \tag{8.5.4}$$

since $W < M$, $(w_1, w_2, \cdots, w_p, w_{p+1}, \cdots, w_{p+r})$ is the unique representation of W in the latter *RNS*. Integer W can be reconstructed from the $p + r - d$-tuple $(w'_1, w'_2, \cdots, w'_p, w'_{p+1}, \cdots, w'_{p+r-d})$ by means of the Chinese Remainder Theorem (as in the case of equation

8.5.1):

$$W = \left(\sum_{i=1}^{p+r-d} w'_i \frac{M'}{m'_i} b'_i \right) \mod M' \tag{8.5.5}$$

where b_i is such that $\left(b'_i \frac{M'}{m'_i} \right) \mod m'_i = 1$ and $i \in [1, p+r-d]$. As a consequence, the above mentioned *RRNS* can tolerate erasures up to multiplicity r.

Fig. 8.4 shows a practical example on how RRNS can be applied for data replication. Let us assume W is a file, with $p = 5$ and $r = 3$. W is split in eight residue segments, i.e., $W_1, W_2, W_3, W_4, W_5, W_6, W_7, W_8$. Supposing that a fail occurs and that W_3, W_5, W_6 are not available anymore, W can be reconstructed considering residue-segments W_1, W_2, W_4, W_7, W_8 and corresponding moduli using the Chinese remainder theorem.

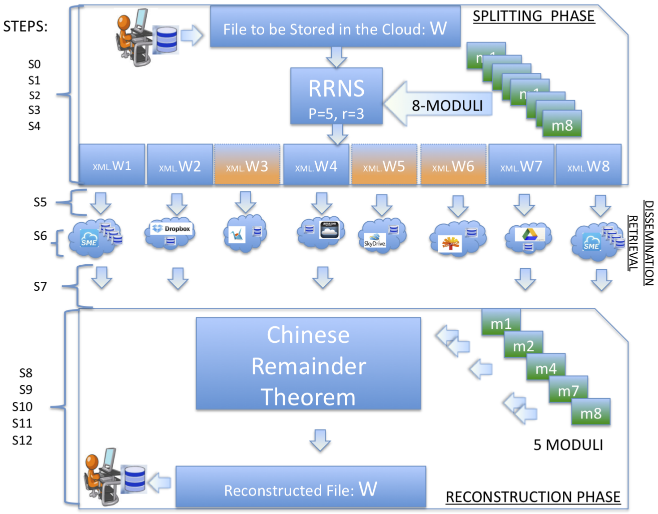


Figure 8.4: Redundant Residue Number System split and recomposition.

8.5.2 Map Files

The Map file is a special XML file that contains both metadata of the EHR and the mapping of chunks. Its structure can be divided in two parts: the "header" section (containing metadata such as the hospital and the patient) and the "file" section containing URLs of chunks. A graphical representation of the structure is shown in Figure 8.5, an example is provided in listing 8.1.

To each EHR multiple map files can be associated. In order to have a valid map file, the "file" section has to fulfill the following requirements:

- it has to contain the minimum number of chunks needed for the recomposition
- any two different sections belonging to different files have to differ at least for one

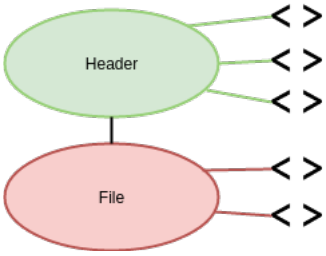


Figure 8.5: Map-file structure

element.

Listing 8.1: An example of map file considering 5 chunks

```
<id>
  <Header>
    <Hospital>Hospital X</Hospital>
    <Patient>Patient Y</Patient>
    <SizeType>noBig</SizeType>
    <Padding>4</Padding>
  </Header>
  <File>
    <Chunk num="2">url_chunk_2</Chunk>
    <Chunk num="10">url_chunk_10</Chunk>
    <Chunk num="9">url_chunk_9</Chunk>
    <Chunk num="4">url_chunk_4</Chunk>
    <Chunk num="6">url_chunk_6</Chunk>
  </File>
</id>
```

In order to better understand how the map-file is made, for simplicity we consider a (3,2) RRNS schema. This means that RRNS create three shares (S_1, S_2, S_3) but, for the recomposition, needs only two of them. In this case, it is possible to build three valid map files: mf_1 which contains in the "file" section the couple (S_1, S_2); mf_2 which contains in the "file" section the couple (S_2, S_3); and mf_3 containing in the "file" section the couple (S_1, S_3). The order of chunks is not significant. That is; the couple (S_1, S_2) is equal to (S_2, S_1).

Considering the (4,2) RRNS schema, six valid map files can be built: (S_1,S_2), (S_1,S_3), (S_1,S_4), (S_3,S_2), (S_4,S_2) and (S_3,S_4).

In general, considering the (k,n) RRNS schema is it possible to build

$$C(k,n) = \frac{k \times k - 1 \times \cdots \times k - n + 1}{n!} \tag{8.5.6}$$

different valid map files, where C(k,n) is the combination of k elements by n.

8.6 Performance Assessment

In this section we discuss about the performances of the proposed system. In particular, we analyzed three different aspects:

1. EHR Split and Recomposition;

2. Upload and Download;

3. Storage required.

We made a scalability analysis in several considering three different EHR sizes: 10kB, 100kB and 1MB. and increasing levels of redundancy: 1, 4, 7. In order to have accurate results, we executed the same operations 30 times and considered the mean time.

Table 8.1 summarises, the performed experiments.

Table 8.1: Summary of experiments performed.

Parameter	Values
Analyses	execution time, transfer time, storage required
Environments	IoT (table 8.2)
Secret Share algorithms	RRNS
Functions	Split, Merge
Primaries (p)	5
Redundancy (r for Split)	1, 4, 7
EHR size [B]	10k, 100k, 1M
Number of Iterations	30
Public Cloud providers	Google Drive, Mega, Dropbox

For storing chunks we considered three public Cloud storage providers (Google Drive, Mega and Dropbox). Instead, the Edge device, in which the RRNS was executed, is a Raspberry pi which hardware (HW) and software (SW) characteristics are shown in Table 8.2.

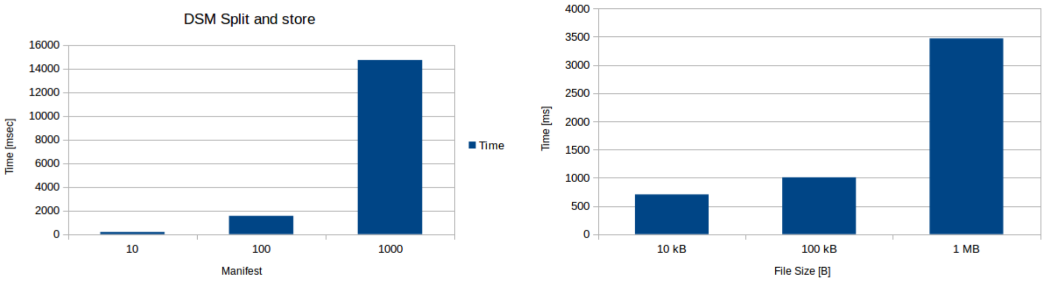
8.6.1 EHR Split and Recomposition

In this Section, we discuss about the RRNS split and recomposition performance.

As the reader can observe in Figure 8.6(a), the time needed for the split of EHRs grows up with the increasing of both EHR size and redundancy. In particular, the behavior is linear, the time spent for the split starts from 1.4 seconds considering EHRs of 10kB in configuration with a single level of redundancy growing up until roughly 3 seconds to split EHRs of 1MB.

Table 8.2: Hardware and Software characteristics of testbeds

Parameter	Values
Model	Raspberry Pi 3
RAM	1 GB LPDDR2-900 SDRAM
CPU	1.2 GHZ quad-core ARM Cortex A53
Storage	MicroSD 8GB ((write throughput: 12.7 MB/s; read throughput: 20.8 MB/s)
Networking	10/100 Ethernet, 2.4GHz 802.11n wireless
Operating System (OS)	Raspbian
GFLOPS	5



(a) RRNS split

(b) RRNS merge

Figure 8.6: RRNS split on IoT

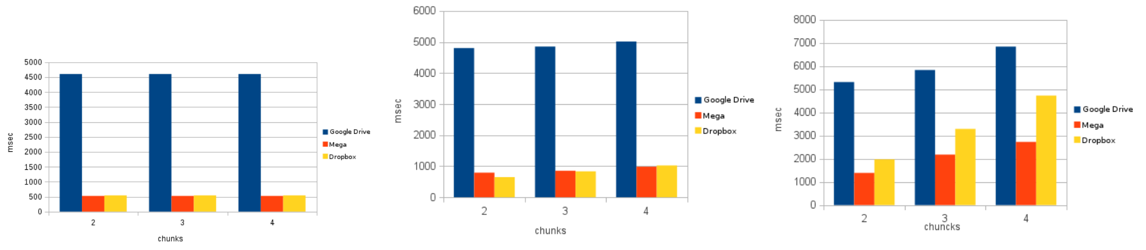
In Figure 8.6(b) the behavior of the recomposition task it is shown. In this case, in the analysis, it is not necessary to consider different degree of redundancy. This because, as we discuss in Section 8.5 the RRNS for the EHR recomposition needs a fixed number of chunks.

As the reader can see, also in this case the behavior is linear. The time needed for the recomposition is 0.7 seconds for EHRs of 10kB, it grows up until 3.5 seconds for EHRs of 1MB.

8.6.2 EHR Upload e download

In this section, we discuss about the time needed for the upload and download of EHR chunks into the public Cloud storage providers.

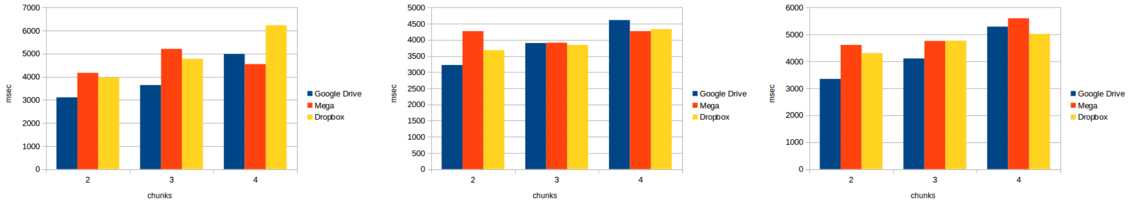
The upload behavior is shown in Figures 8.7(a)-(c). In particular, Figure (a) refers to 10kb EHR, Figure (b) to 100kB whereas Figure (c) to 1MB. The y axis of the chart represent the time needed for the upload expressed in milliseconds, whereas, the x axis the number of chunks stored in each Cloud service provider. We remark that in our performance evaluation we considered the degree of redundancy equal to 1 (2 chunks per each Cloud service provider), 4 (3 chunks per each Cloud service provider) and 7 (4 chunks per each Cloud service provider).



(a) Size = 10 kB (b) Size = 100 kB (c) Size = 1 MB

Figure 8.7: Upload of chunks of EHR in public Clouds

As the reader can observe in Figures 8.7 (a) and (b), the time required for the upload of chunks is constant at the increasing of the number of chunks. This behavior is justified by the fact that chunks are very small, therefore the time required for internal synchronization is greater than the effective time needed for the upload. A different behavior can be seen in Figure 8.7 (c). In this case, the time required for the upload is not negligible therefore the total time required for the upload depends on the number of chunks. As the reader can observe, in all presented scenarios Google Drive (the blue bar) presents the worst performance this because it require more time to start the synchronization. Furthermore, in Figure 8.7 (c) we can see that the time required from Dropbox (the yellow bar) increase faster than Mega (red bar) this because Dropbox, in order to optimize the storage divide files in chunks of 100 kB before the transmission.



(a) Size = 10 kB (b) Size = 100 kB (c) Size = 1 MB

Figure 8.8: Download of chunks of EHR from public Clouds

Figures 8.8 (a)-(c), show the behavior for the download of HERs of 10 kB, 100 kB and 1Mb respectively. In these Figures the y axis represents the time needed for downloading chunks expressed in milliseconds, whereas, the x axis the number of chunks stored in each Cloud service provider. As a well as for the upload, also in this case we considered 2, 3 and 4 chunks per each Cloud storage provider. As the reader can observe, in all scenarios, differently from the upload phase, the time required from all Cloud service providers are comparable and Google Drive has better performances that others.

8.6.3 Storage required

In this Section we discuss about the Storage required for the proposed approach considering different degree of redundancy. In particular, we compared the RRNS with the file replication. For seven levels of redundancy, the RRNS requires roughly the 50% of the capacity required from file replication. This because each share in RRNS is the result of the modulo operation between the secret and a modulo. This means that each share will be smaller of both the secret and the modulo. In our implementation, we codified as 'integers (32 bits)' the secret and 'bytes (8 bits)' modules and shares. Therefore, the size of each share is $\frac{1}{4}$ of the size of the EHR. For compatibility reasons, we codified each chunks as base64 that introduces the 33% of overhead. The mathematical formula that expresses the storage required is the following:

$$S = \frac{p + r}{4} \times 1.33 S_{\text{EHR}} \tag{8.6.1}$$

Where p is the number of primaries (fixed to 5 in our case), r the level of redundancy and S_{EHR} the size of the EHR. The Figure 8.9 shows the behavior of two approaches.

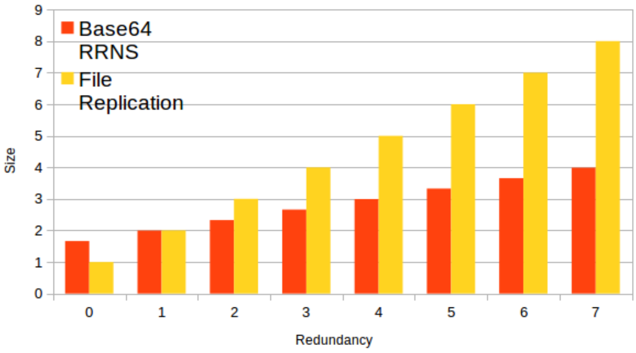


Figure 8.9: Storage Required for Base64 RRNS and replication approaches.

8.7 Open Issues and Challenges

In this section, we discuss the security issues related to the system we proposed in Section 8.5 in both use cases presented in Section 8.4.

Generally, in the health sector, the main obstacle to effective EHR systems to ensure ongoing continuity of healthcare is lack of interoperability of EHR systems used by patients, healthcare providers and security of patient's data. The lack of data integration makes it difficult to offer accurate and timely treatments. Therefore, the proposed EHR system requires the collaborations outside the organizational boundaries in order to promote effective delivery of health care for individuals and companies, as well as allowing savings and efficiencies. During the medical assistance service, a large amount of data is created which must be securely stored.

Data transfer and consent management in use cases presented in Section 8.4 can become complex and inconvenient: the patient may need to get in touch with the caregiver and sign consent in the hospital from which he is not receiving care any longer. Transfer of data can take time, and on receiving the hard copy of the patient data, a clinician will have to introduce them into the system again. Furthermore, with this approach, it is very difficult for the patient to maintain any access control or log of his data and to have a complete view of the data. Furthermore, the lack of integration makes it difficult to maintain the privacy and security of the patient's confidential information. Patient's EHR security is a cause for concern especially when patient data is transmitted over a network. On the other hand, however, sharing EHR can benefit patients and medical organizations in various ways. Firstly, data sharing can facilitate medical research, such as bringing together multiple medical studies for better knowledge and scientific discoveries. Secondly, collaboration between different health organizations (including cross-border cases) will be easier, such as doctors accessing patient records, reimbursement of medical care in a foreign country and so on. Thirdly, regulations and standards will be developed and strengthened to facilitate secure sharing of data centers, which in turn will increase trust between different medical organizations and can therefore offer patients better service and further improve medical research. the main security issues related to EHR sharing there are confidentiality, privacy, access control, integrity, data authenticity, user authentication, audibility and transparency. It is urgent to resolve these problems to enable secure sharing of EHRs.

During recent years, blockchain technology has been advocated as a promising solution to solving EHR interoperability and security issues. A blockchain is a decentralized, trustless protocol that combines transparency, immutability, and consensus properties to enable secure, pseudo-anonymous transactions stored in a digital ledger. Blockchain enables a potentially evolving and open set of parties to maintain a safe, permanent, and tamper-proof digital ledger of transactions, without a central authority. It consists of consecutive chained blocks, replicated and stored by the nodes of a peer-to-peer network (p2p), where blocks are created in a distributed fashion by means of a consensus algorithm. Such algorithm, together with the use of crypto mechanisms, provides two properties of Blockchain: 1) decentralisation and 2) democratic control of data. This ensures that data on the chain cannot be tampered maliciously, operations on the chain are non repudiable and their origin is fully tracked.

Blockchains are then typically classified in two main categories:

1. Permissionless: a block can be added to the blockchain by any process, but only if a cryptographic puzzle is solved (Proof-of-Work). Examples are Bitcoin, Zerocash and Ethereum.

2. Permissioned: closed group of processes execute a (traditional) Byzantine consensus algorithm to add a block to the Blockchain. Consensus decisions are either taken unilaterally by this central entity, or by a preselected group (so-called "consortium blockchains"). Permissioned Blockchains can be further categorized into public and private. Examples include Hyperledger.

Unlike Bitcoin, new types of Blockchains such as Ethereum, Hyperledger Fabric (HLF) have recently appeared with featuring smart contracts: programs deployed and executed on blockchain. Smart contract allows the creation of so-called decentralised applications, i.e applications that operate autonomously and without any control by a system entity and whose logic is immutably stored on a blockchain. We discuss each of the above mentioned security issues and how blockchain technology can solve each one in the following:

Confidentiality is one of the major concerns is sharing patient's EHRs between multiple organizations over the network. Using the blockchain, the patient can have full access to the data and control how the data is shared maintaining privacy and security. For example, patients who are part of the blockchain would be able to approve or deny any sharing or modification of their data, helping to ensure a greater level of privacy and greater consumer control. Furthermore, the blockchain is based on consolidated cryptographic techniques to allow each participant in a network to interact without the pre-existing trust between the parties. Due to the encryption of blockchain information, patient EHRs confidentiality is preserved when it is shared between the parties involved.

Privacy Ensuring the safety of the EHR and the underlying components of the ecosystem is crucial but difficult due to the interaction and complexity between systems and components. Furthermore, the privacy and integrity of the extremely sensitive healthcare data must be ensured not only from outside attackers, but also from unauthorized access attempts within the network (e.g health service provider employee, or cloud service provider) [146]. Attacks intentional and unintentional (such as data loss or data modification) can penalize or held criminally liable for such incidents organizations, for example under the Health Insurance

Portability and Accountability Act. The protection and traceability a fundamental principle of the chain of trust. Smart contracts allow to encrypt patient data, and permissions are sometimes taken to access this content. The data will be digitally signed, and caregivers will need access grants to access it.

Access control There are mainly two aspects related to EHR access, namely a patient accessing their EHRs and a medical personnel accessing the patients EHRs. For instance, the patient should be able to grant who can access his EHRs and also should maintain and track any change of his medical history. By applying blockchain for the permissions management, we enable the exchange of patient-controlled data between medical organizations and an interoperable content management system for physicians who supervise these records. Moreover, in the future, this verifiable history of medical interactions between patients and suppliers technology can relevant for regulators and payers (eg insurance).

Integrity Another concern in patient's EHRs sharing is the integrity; EHRs integrity is crucial in prescribing the correct treatment. Any modification can produce potential patient's death. Furthermore, data integrity is guaranteed by blockchain. This is mainly achieved by using the consent protocol and cryptographic primitives such as hashing and digital signatures. The use of hashing and digital signatures makes the manipulation, verification and accessibility of patient data or data with public and private keys almost impossible, and in essence, security is fully guaranteed. In fact, it is computationally difficult to tamper with the contents of a block without changing the hash value stored in the next block. Furthermore, the distributed blockchain architecture is integrated into fault tolerance and disaster recovery.

Data authenticity Each entity (patients and medical personnel) who generated an EHR should be able to ensure that that one was created by itself. In order to ensure this, patients and medical personnel must sign their EHR before being are added to the blockchain. With these signatures, blockchain provides the authenticity of the data.

Audibility and Transparency Given that a patient can be treated from different doctors belonging to different healthcare organizations and therefore each one generates a different EHR for each treatment, patients should be able to track his medical history. Blockchain technology is the most suitable solution. As all changes in patient data will be recorded and then added to the blockchain, an integrated, clear and transparent audit trail of EHR changes can be maintained, and patients can have access to these paths as long as they have access to the blockchain network.

Interoperability The patients can be treated by different medical organizations also belonging to different jurisdictions. The current medical system consists of a complex network of healtcare organizations; each organization uses its own methods for storing and accessing medical data (e.g off-line architecture with centralized and local databases); cooperation and sharing among the various organizations is almost inexistent. In general, data exchange in a coherent manner requires a shared network, as well as a solid platform for sharing and cooperation, enabling easy understanding and operation. Unlike a centralized network where different healthcare organizations have their own systems, the decentralization of the blockchain is designed to be convenient for each entity to communicate with each other which means that all the blockchain entities will use the same standard for EHRs sharing and management.

8.8 Conclusions and Future Works

In this Chapter we discussed the use of Edge devices for the management of EHR. In particular, we considered two use cases: hospitalized patients and ambulances. We described a possible architecture consisting of private and public Cloud storage services and Edge devices running the RRNS. In order to evaluate the proposed approach we made three different type of analyses: i) execution time for the EHR split and recomposition; ii) time required for upload and download of EHR chunks and iii) storage required. We considered three different EHR sizes from 10kB to 1MB and three different public Cloud storage providers (Google Drive, Mega and Dropbox). With regards to EHR split and recomposition we made the scalability analysis. We have seen that, contrary to the RRNS merge, the execution time of the RRNS split depends on the level of redundancy. Whereas, regarding upload and download of chunks we have seen that Google Drive, due to synchronization overhead, is slower than other Cloud service providers for the upload of chunks but faster for the download. In future works, in order to monitor the accesses to EHRs and discover attackers, we plan to design and implement a logger system based on blockchain.

CHAPTER 9

Adding long term availability and security to drones by using Nested Secret Share techniques

Military drones are not only used for fight wars, they are also used for surveillance and for spying enemies. More often they are destroyed or captured, therefore photos contained inside are lost or revealed. A possible solution to solve such a problem is to use techniques of Secret Share (SS) to split photos among fleet of drones.

These techniques assure a good level of redundancy, however their security level is fixed. Therefore, after a long time, attackers can get enough information to recover the secret. A possible solution to improve the security is to nest Secret Share techniques. However, Nested Secret Share (NSS) have not been evaluated before. This work aims to address this issue by comparing SS with 1 Level Nested Secret Share (1LNSS) techniques. In particular, we considered the redundancy, the retrieval rate, the execution time, the power consumption and the storage required. We analysed Shamir's Secret Share schema and the Redundant Residue Number System (RRNS). We made our experiments for different file sizes (from 500kB up to 50MB), data redundancy (0 to 7) and fleet sizes in both Single Level and 1 Level Nested. Results showed that Shamir's schema, due its complexity, is not suitable for drones especially considering high degrees of redundancy and small fleet sizes.

9.1 Introduction

Drones are used for both military and civil purposes including surveillance, spying enemies, photography, rescue operations, infrastructure monitoring, farming and aerial mapping [147].

More often, they are destroyed [148] or captured by enemies [149], therefore photos contained inside are lost or revealed.

A possible solution to partially solve such a problem is to use techniques of file share to split photos and distribute them among a fleet of drones. Among these techniques, Secret Share (SS) is becoming of particular interest within the scientific community.

SS allows to split a secret into fragments and to share them among parties in such a way that a subset of fragments is enough to recompose the original information. These techniques assure a good level of redundancy, however the security level is fixed. Therefore, enemies capturing a fixed number of drones can get enough information to recover photos inside drones. A possible solution to improve the security is to nest Secret Share techniques. However, for the best of our knowledge, Nested Secret Share (NSS) techniques have not been proposed before.

To address this issue and make this study, in this chapter we discuss about SS and then we propose NSS. In particular, considering an use-case with fleets of drones that go in mission to take photos of enemies, we analyse Shamir's Secret Share schema and the Redundant Residue Number System (RRNS). Our contribution in this work can be summarised as the followings. Firstly, we analyse and compare SS algorithms; then we propose NSS techniques; and finally we compare SS and NSS discussing the the suitability for drones. To be more specific, for each technique, we will consider redundancy, retrieval rate, execution time, power consumption and storage requirements. We performed experiments for different file sizes (from 500kB up to 50MB), data redundancy (0 to 7) and fleet sizes in both SS and 1 Level Nested Secret Share (1LNSS). Results were promising and showed that, for example, the number of primaries of Nested Secret Share techniques depends on the number of nesting levels; or, to have 119 degrees of redundancy, Nested RRNS uses $\sim$ 39% of the storage required from RRNS, and only $\sim$ 1.5% of Nested Shamir; or, RRNS uses less energy and drones need smaller batteries for the computation. We also discovered that by using lower values of redundancy the recomposition of photos is more difficult.

The rest of the chapter is organised as follows. Section 9.2 describes related works. Section 9.3 discusses NSS. Experiments, are discussed in Section 9.4. Conclusions and our future directions are summarised in Section 9.5.

9.2 Related Work

9.2.1 Secret Share

Secret Share is a very hot topic within the scientific community; yet, there are not so many works available in literature focusing on Nested Secret Share techniques.

In [150], authors presented a VSS for medical images. Their system is based on two modules: the sender, that generates shares using the Block-based Progressive Visual Secret Sharing (BPVSS) scheme and embed into them low-resolution images, and the receiver that extracts data from shares, recovers secret images and increases the contrast by using Super resolution techniques.

In [151], authors discuss Physically unclonable functions (PUFs) based on Secret Share. Their idea is to create a lightweight system suitable for energy constrained platforms such as IoT devices and smart cards.

In [152], authors discuss a multi-cloud system based on Secret Share. In particular, they used the RRNS to spread files among public Clouds.

Several works employ Secret Sharing techniques in health domain: in [153] and [154], authors adopted RRNS to store and retrieve Magnetic Resonance Images (MRI) from public Clouds; in [155], authors proposed a Remote Monitoring System that stores patients' clinical data such as temperature, heartbeat, ECG etc. in different servers by using Shamir's schema.

In [156], authors proposed a verifiable SS for images by using the XOR operator. In particular, they proposed an (n,n) schema which $n - 1$ shares are obtained splitting the secret image, the n^{th} share is constructed by using the XOR operation between secrets and a random image.

A similar approach is presented in [157]. In their work, authors proposed a (t,n) schema-based verifiable SS for images able not only to reconstruct the original image in presence of fake data but also to identify cheaters.

In [158], we analysed and compared Shamir and RRNS in order to verify their applicability on IoT, Edge and Cloud. We discovered that the environment on which the computation should be performed depends on both file size and algorithm. For instance, when employing RRNS, files up to 500kB can be processed on the IoT, up to 50MB on the Edge, and beyond that on the Cloud; whereas, in Shamir's schema, the threshold to move the computation from the IoT to the Edge is about 50kB, and from the Edge to the Cloud is about 500kB.

9.2.2 Image processing on drones

In [159], authors proposed an encoding and communication technique of images for drones based on Reed Solomon algorithm. In particular, they proposed a 4-bits encoder able to correct 2 errors per each block. In order to verify the correctness of the decoded image, they create a new image starting from checksum values. If the new image is full black then there are no errors.

In [160], authors proposed a novel video surveillance system based on drones. In particular, by using drones features (such as autonomous flight capabilities) and processing images (by using Probability Hypothesis Density algorithms), they send swarm of drones able to track humans interactions.

In [161], authors proposed a surface water monitoring system by using drones. In particular, the proposed drone implements several functions: it is able to i) collect samples of water; ii) send real time video to the base station; and iii) study the effect of wind & water parameters.

In [162], another video surveillance system based on drones has been proposed. In particular, authors incorporates motion features to commercial drones.

In [163], authors proposed a compression algorithm that, by using the predicted drones trajectory, it is able to send to the base station only important frames. The goal of this algorithm is to minimize the usage of bandwidth allowing the real-time processing of images on the base station.

9.3 Nested Secret Share

As we discussed in Section 6.3, p represents the security degree of SS algorithms, for further details about SS, please refer to Section 6.3. Although this value theoretically is unlimited for Shamir, Blakley and Proactive, it is fixed for RRNS. Therefore, in order to increase the security level of Secret Share algorithms, a solution could be to nest recursively two or more Secret Share algorithms.

The basic idea of the Nested Secret Share techniques is to use as input of the following Secret Share level the output of the previous one.

Let us consider $p_1 = x$ and $p_2 = y$ with $x, y > 0$ and $r_1 = r_2 = 0$, where p_1 is the number of primaries of the first level of Secret Share, p_2 the number of primaries of the second level of Secret Share, r_1 the value of redundancy of the first level of Secret Share, and r_2 the value of redundancy of the second level of Secret Share. The output of the first level of Secret Share performed on a generic secret S is $(S_1, S_2, \ldots, S_x)$; using each S_i as input of the second level of Secret Share, the output will be $(S_{1,1}, S_{1,2}, \ldots S_{1,y}), (S_{2,1}, S_{2,2}, \ldots S_{2,y}), \ldots (S_{x,1}, S_{x,2}, \ldots S_{x,y})$.

The total number of shares, and primaries in this case, is $p_1 \times p_2$. In general, considering n level of nesting Secret Share, we can calculate the number of primaries by using the following formulation:

$$p_1 \times p_2 \times \cdots \times p_n \tag{9.3.1}$$

where p_i is the value of p at the i_{th} level. Considering $p_1 = p_2 = \cdots = p_n = p$ the total number of primaries is:

$$p^n \tag{9.3.2}$$

Without loosing of generality, if $r_1, r_2, \ldots r_n$ are greater than 0 the total number of shares can be calculated by means of the following formulation:

$$(p_1 + r_1) \times (p_2 + r_2) \times \cdots \times (p_n + r_n) \tag{9.3.3}$$

Considering $p_1 = p_2 = \cdots = p_n = p$ and $r_1 = r_2 = \cdots = r_n = r$ the number of shares is:

$$(p + r)^n \tag{9.3.4}$$

In order to calculate the redundancy level of Nested Secret Share techniques it is sufficient to subtract from the total number of chunks the number of primaries, therefore the generic formulation that represent the redundancy level is the following:

$$(p_1 + r_1) \times (p_2 + r_2) \times \cdots \times (p_n + r_n) - p_1 \times p_2 \times \ldots p_n \tag{9.3.5}$$

Considering $p_1 = p_2 = \cdots = p_n = p$ and $r_1 = r_2 = \cdots = r_n = r$ the value of redundancy is:

$$(p + r)^n - p^n \tag{9.3.6}$$

calculating the binomial power, the formula above can be written as:

$$p^n + np^{n-1}r + ... + \frac{n!}{k!(n-k)!}p^{n-k}r^k + ... + npr^{n-1} + r^n - p^n \tag{9.3.7}$$

that corresponds to:

$$np^{n-1}r + ... + \frac{n!}{k!(n-k)!}p^{n-k}r^k + ... + npr^{n-1} + r^n \tag{9.3.8}$$

Nesting levels of Secret Share introduces an overhead from computational point of view. That is, the time required for the split and recomposition of Nested Secret Share algorithms depends on p, r, number of nested levels and the nodes that can make the computation. Considering a single node scenario, the execution of Secret Share has to be sequentially, therefore the time required for splitting and merging files can be calculated with the formulations 9.3.9 and 9.3.10 respectively.

$$Ts = Ts_1 + (p_1 + r_1) \times Ts_2 + \cdots + (p_{n-1} + r_{n-1})^{n-1} \times Ts_n \tag{9.3.9}$$

$$Tm = p_{n-1} \times Tm_n + \cdots + p_1 \times Tm_2 + Tm_1 \tag{9.3.10}$$

Considering $p_1 = p_2 = \cdots = p_n = p$ and $r_1 = r_2 = \cdots = r_n = r$ the time required for splitting and merging files can be calculated with the following formulations:

$$Ts = Ts_1 + (p + r) \times Ts_2 + \cdots + (p + r)^{n-1} \times Ts_n \tag{9.3.11}$$

$$Tm = p \times Tm_n + \cdots + p \times Tm_2 + Tm_1 \tag{9.3.12}$$

Where Ts is the total time required for the split, Ts_i the time required for the split at the i_th level, p the number of primaries, r the redundancy, T_m the total time required for the merge Tm_i the time required for the merge at the i_th level and n the number of nested levels. In order to reduce the neck-to-neck computation time of NSS a possible solution is to run several Secret Share tasks in parallel. In particular, consideting a multi-node scenario able to run $(p + r)^{n-1}$ parallel Secret Shares, the time required for splitting and merging can be calculated by means of the formulations 9.3.13 and 9.3.14 respectively.

$$Ts = Ts_1 + Ts_2 + \cdots + Ts_n \tag{9.3.13}$$

$$Tm = Tm_n + \cdots + Tm_2 + Tm_1 \tag{9.3.14}$$

In Figure 9.1 an example of two levels Nested Secret Share in multi-nodes environment is shown.

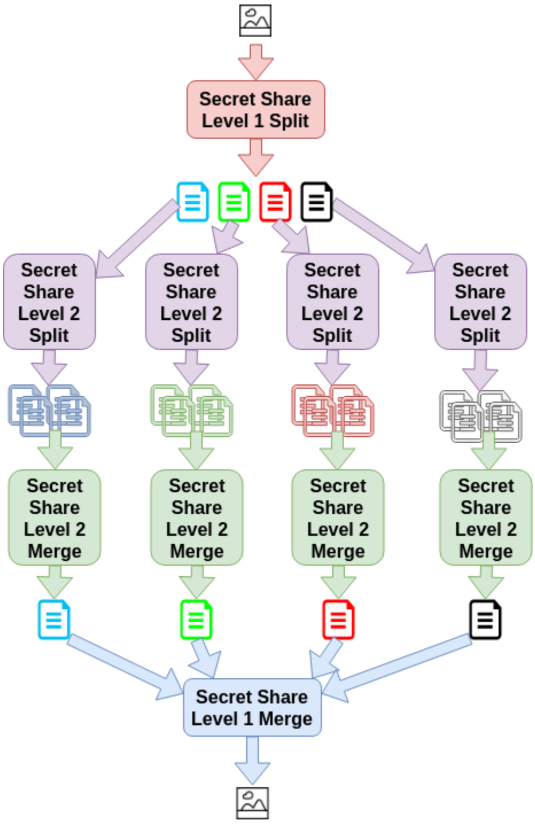

Secret Share Level 1 Split
Secret Share Level 2 Split
Secret Share Level 2 Split
Secret Share Level 2 Split
Secret Share Level 2 Split
Secret Share Level 2 Merge
Secret Share Level 2 Merge
Secret Share Level 2 Merge
Secret Share Level 2 Merge
Secret Share Level 1 Merge

Figure 9.1: Nested Secret Share Split and Recomposition in multi-node environment

9.4 Performance assessment

In this Section, we compare SS with 1LNSS. In particular, we made two types of analyses from the technical point of view (redundancy, security and storage requirements) and the practical one (execution time for split and merge functions and power consumption).

We make our analyses considering different fleet sizes (10, 100 and 1000 drones) that go in mission. Each mission finishes when the total number of photo is 1000 (100, 10, 1 photos for drone respectively). In our experiments we compared RRNS and Shamir's schema. For each level of Secret Share we considered the number of primaries p fixed to 5 (because RRNS in our implementation needs 5 primaries), whereas we varied the redundancy value for each level $r1$ and $r2$ from 0 to 7. The size of the input file (a photo or a small video in our case), varied from 0.5MB to 50 MB.

Table 9.1, summarises the performed experiments. The reference environments which these experiments were carried out was drone for the split function and p Virtual Machines (VMs) running on the drone base for the merge. Hardware (HW) and Software (SW) characteristics of these environments are shown in Table 9.2.

Table 9.1: Summary of experiments performed.

Parameter	Values
Environments	Drones, Cloud (Merge) (table 9.2)
Secret Share algorithms	Shamir, RRNS
Functions	Split, Merge
Primaries (p)	5
Redundancy ($r1, r2$)	$0, 1, 3, 5, 7$
Nested Levels	1
File size [B]	500k, 5M, 50M
Fleet size	10, 100, 1000
Missions	30
Photos for Mission	1000

Table 9.2: Hardware and Software characteristics of testbeds.

Parameter	Drone	Cloud (VM)
RAM	1 GB LPDDR2-900 SDRAM	16 GB DDR3-1600 SODIMM
CPU	1.2 GHZ quad-core ARM Cortex A53	3.4 GHz eight-core Intel(R) Core(TM) i7-3770 CPU
Storage	MicroSD 8GB W throughput: 10.9 MB/s; R throughput: 22.6 MB/s	SSD 100GB W throughput: 998 MB/s; R throughput: 367 MB/s

9.4.1 Reference Scenario

In this Section we discuss about our reference scenario. It is composed of two parts: i) drones that go in mission to take photos to the enemy and ii) the drone base for the merging of pictures. In this scenario, we are assuming that the split of pictures is done sequentially on drones, this because each drone has to elaborate the same amount of pictures therefore there are no drones "free" than could parallelise the computation. Instead the merge task can be done in parallel, this because we are making the computation inside the drone base therefore we can run several Virtual Machines (VM) in parallel. Figure 9.2 shows a high level representation of the reference scenario.

9.4.2 Redundancy

In this Section, we investigated the total redundancy degree comparing SS with 1LNSS. Its value can be calculated simply subtracting the number of primaries from the total number of chunks.

In the chart below the x axis represents the value of redundancy of the first level of Secret

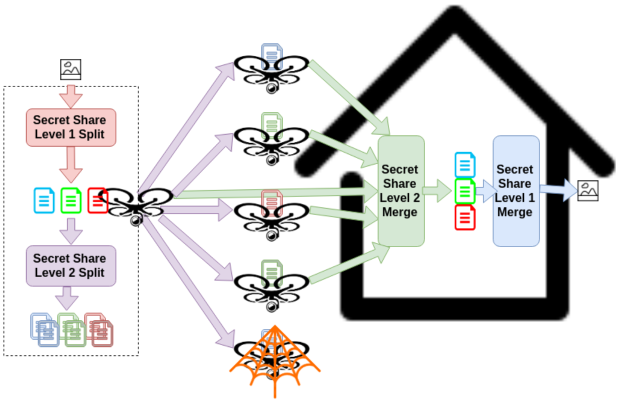

Secret Share Level 1 Split
Secret Share Level 2 Split
Secret Share Level 2 Merge
Secret Share Level 1 Merge

Figure 9.2: Reference Scenario

Share $r1$, the y axis is the total value of redundancy. The continuous line represents the SS instead the dashed lines the 1LNSS (the lightest line represents $r2 = 0$, the darkest $r2 = 7$).

Figure 9.3 shows the behavior.

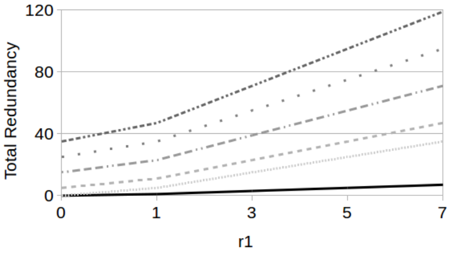
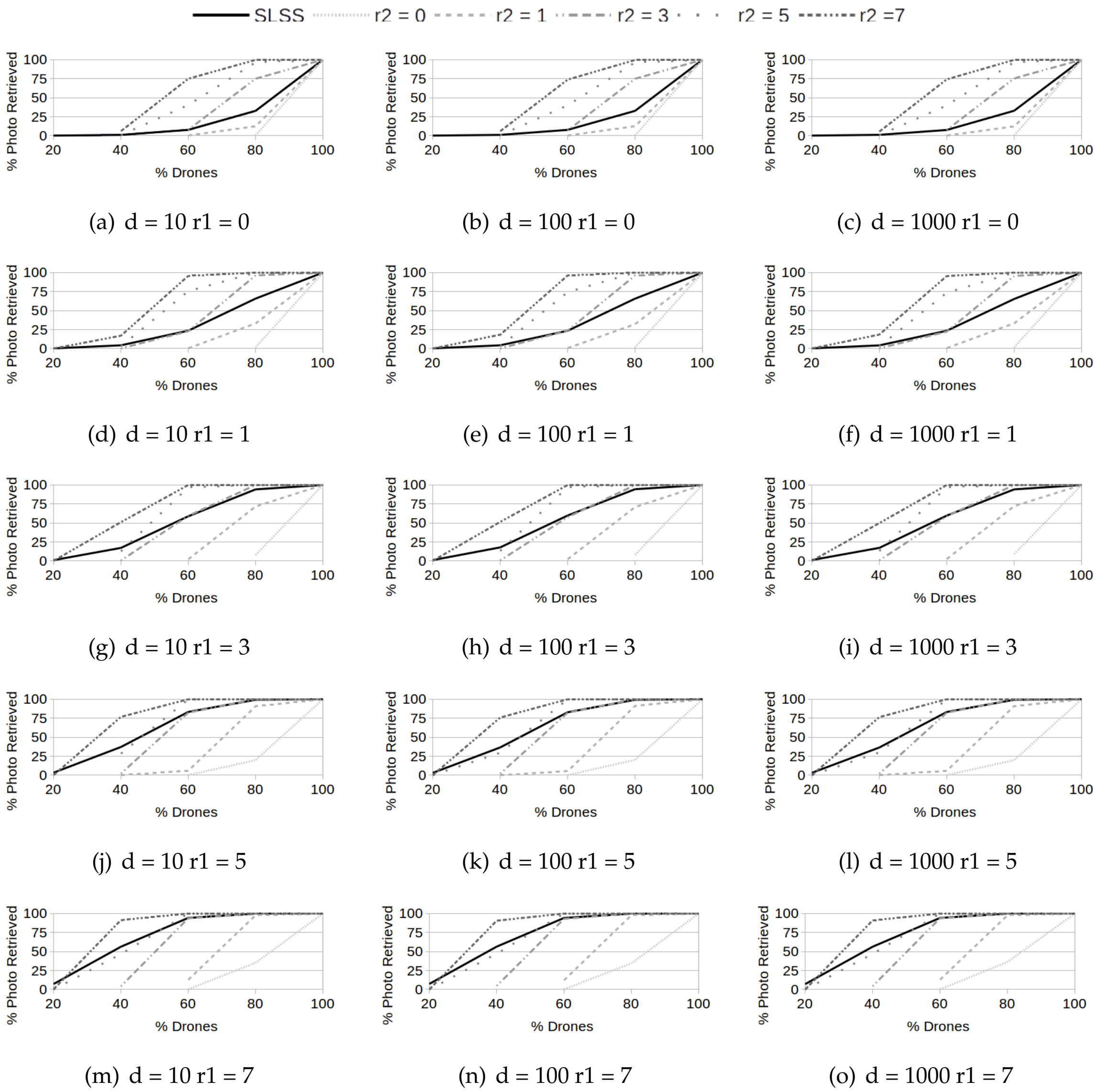


Figure 9.3: Total Redundancy considering different levels of $r1$ and $r2$

As the reader can observe, considering SS, the values of x and y are the identical. Instead, as we discussed in Section 9.3, the value of redundancy of 1LNSS algorithms depends on both p, $r1$ and $r2$ and follows the equation 9.3.5.

For simplicity, in our experiments we fixed p to 5 and considered $r1$ and 2 equal to 0, 1, 3, 5, 7. That is, when $r1$ is equal to 0, the value of the total redundancy varies from 0 ($r2 = 0$) to 35 ($r2 = 7$) instead considering $r1 = 7$ the total redundancy varies from 35 ($r2 = 0$) to 119 ($r2 = 7$); for different values of $r1$ and $r2$ we have all intermediate values. As we can see in the Figure, the redundancy level of the 1LNSS (except for $r0 = r1 = 0$) is always bigger that the redundancy degree of SS. This feature can be good in an ideal system with unlimited number of drones because guarantees better levels of availability and security of data. That is, considering that each drone contains exactly a chunk and that one or more of them are destroyed or captured by the enemy. The part of information that we lose (or the enemy gains) is not enough to influence our recomposition task or to disclose information to the enemy. This property could not be valid in real cases with fleets composed of a limited number of drones because each of them contains a bigger amount of chunks.

9.4.3 Security



(a) d = 10 r1 = 0

(b) d = 100 r1 = 0

(c) d = 1000 r1 = 0

(d) d = 10 r1 = 1

(e) d = 100 r1 = 1

(f) d = 1000 r1 = 1

(g) d = 10 r1 = 3

(h) d = 100 r1 = 3

(i) d = 1000 r1 = 3

(j) d = 10 r1 = 5

(k) d = 100 r1 = 5

(l) d = 1000 r1 = 5

(m) d = 10 r1 = 7

(n) d = 100 r1 = 7

(o) d = 1000 r1 = 7

Figure 9.4: Percentage of photo reconstructed considering different fleet sizes

In this Section, considering a real scenario with fixed fleet sizes, we investigated the retrieval rate of proposed Secret Share techniques. The retrieval rate can be seen as ‘availability’ property (in terms of number of photo/video that we are able to recompose considering the number of drones that back to the base), but also ’security’ (in terms of number of photo/video that the enemy is able to recompose capturing drones). In particular, we considered different fleets, composed of 10, 100 and 1000 drones, that go in mission to take photos of the enemy. The mission ends when the total number of taken photo is 1000 (100, 10 and 1 photo per each drone respectively). We made 30 missions and calculated the average number of photo recomposed/compromised. In Figure 9.4 the x axis of each subfigure is the percentage of drones that come back to the base (availability) or is captured (security). The y axis is the percentage of photos that it is possible to recompose.

Figures 9.4(a), 9.4(d), 9.4(g), 9.4(j) and 9.4(m) refers to fleet size equal to 10 and $r1$ equal to 0,1,3,5,7 respectively; Figures 9.4(b), 9.4(e), 9.4(h), 9.4(k) and 9.4(n) refers to fleet size equal to 100 and $r1$ equal to 0,1,3,5,7 respectively; and Figures 9.4(c), 9.4(f), 9.4(i), 9.4(l) and 9.4(o) refers to fleet size equal to 1000 and $r1$ equal to 0,1,3,5,7 respectively. In the Figures, as we discussed for the redundancy, the continuous lines represents the SS whereas dashed lines 1LNSS. In our experiment considering different values of $r1$ and $r2$, we compared the proposed approaches. As we can see, in all configurations the lightest dashed line ($r2 = 0$) is on the right and under the SS (our reference), this means that in order to recompose photos we need (or the enemy needs) a larger portion of the fleet. That is, loosing more than 20% of the fleet size does not allow to reconstruct any photo. This configuration can be useful when the goal is not to reveal any information to the enemy. If more than 20% of the fleet size is compromised we make the mission again. Considering $r2 = 7$ we have the opposite behavior; that is, the darkest dashed line is on the left and above the SS line, this means that we are (or the enemy is) able to recompose almost all photos with a small portion of the fleet. This kind of configuration is the optimal when retrieve information is more important than disclose. Considering other values of $r1$ and $r2$ we have intermediate conditions. In our experiments, we considered a random distribution of chunks among drones, in future works we plan to apply mathematical models in order to have better results. As we can see in the Figure 9.4, the 'retrieval rate' does not depends on the fleet size. However, it influences the total execution time, as we will discuss in Section 9.4.5.

9.4.4 Storage Analysis

The amount of storage required for saving files is not a trivial aspect, especially for drones with limited resources. In this Section, starting from results of Section 9.4.2, we analysed and compared the behaviour of RRNS and Shamir techniques.

Figure 9.5 shows the storage required from RRNS and Nested RRNS compared to simple file replica.

For the simple-replication strategy, in which, the content of a file is simply copied several times, the storage requirement can be described as:

$$Size_{out}^{simple} = (r+1) \times Size_{in} \tag{9.4.1}$$

where r is the number of replicas plus '1' for the original file.

The RRNS algorithm requires less amount of storage; in fact, for the redundancy value equal to 119 it requires ∼33% of storage as compared with simple-replication. This is due to the fact that each share in RRNS is the remainder of the division operation between the secret and a modulo, and thus its value is always lower than the modulo. As a result, shares will be smaller than the secret as well. In our implementation, we used 'integers (32 bits)' as primitive type of the secret and 'byte (8 bits)' as primitive type of modules and shares. Hence, the size of each chunk becomes four times smaller than the input file, plus 33% because chunks are encoded in base64. The RRNS storage requirement can be calculated as:

$$Size_{chunks}^{RRNS} = \frac{p+r}{4} \times (1+\frac{1}{3}) \times Size_{in} \tag{9.4.2}$$

The storage required from Nested RRNS strategy can be calculated by means of the formulation 9.4.2 considering as $Size_{in}$ of the level n the $Size_{chunks}^{RRNS}$ calculated at the step $n-1$. The following formulation describe the behaviour:

$$Size_{chunks}^{NRRNS} = \frac{p_1+r_1}{4} \times (1+\frac{1}{3}) \times \cdots \times \frac{p_n+r_n}{4} \times (1+\frac{1}{3}) \times Size_{in} \tag{9.4.3}$$

that is equivalent to the following one:

$$Size_{chunks}^{NRRNS} = \frac{p_1+r_1}{4} \times \cdots \times \frac{p_n+r_n}{4} \times (1+\frac{1}{3})^n \times Size_{in} \tag{9.4.4}$$

if $p_1 = \cdots = p_n = p$ and $r_1 = \cdots = r_n = r$ the previous formulation can be written as follows:

$$Size_{chunks}^{RRNS} = (\frac{p+r}{4})^n \times (1+\frac{1}{3})^n \times Size_{in} \tag{9.4.5}$$

that is equivalent to:

$$Size_{chunks}^{RRNS} = (\frac{p+r}{4} \times (1+\frac{1}{3}))^n \times Size_{in} \tag{9.4.6}$$

As we discussed in Section 9.4.2, in order to achieve the redundancy level equal to 119 we fixed $p_1 = p_2 = 5$ and $r_1 = r_2 = 7$. The total amount of storage required from Nested RRNS is $\sim 16 \times Size_{in}$, almost the 39% of the storage required from single level RRNS.

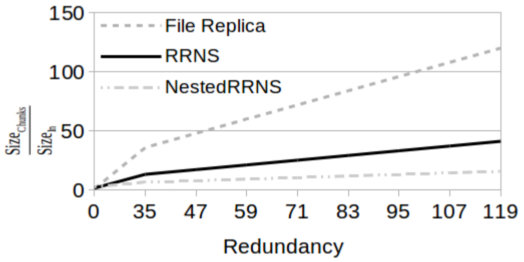


Figure 9.5: Storage requirement for RRNS considering different values of redundancy

Figure 9.6 shows the behaviour of Nested Shamir comparing it to Single level Shamir and File Replica.

The Shamir's algorithm needs more storage compared with the File replication; it starts from ~13 times the size of the input file and increases up to ~ 330 time of it. This is because, as we discussed in Section 6.3 and shown in Equation 6.3.1, several non-negative numbers need to be added for each secret. To avoid computational overflows, we read secrets as 'short (16 bits)' and store them as 'integer (32 bits)' in our Java implementations of these algorithms. Therefore, the size of each chunk is twice the size of the original file plus 33% more because files are encoded as base64. The mathematical formula to describe Shamir's storage requirement is:

$$Size^{Shamir}_{chunks} = (p+r) \times 2 \times (1+\frac{1}{3}) \times Size_{in} \tag{9.4.7}$$

where $Size^{Shamir}_{chunks}$ is total size of all chunks, and $Size_{in}$ is size of the input file. As a well as for Nested RRNS the storage required from Nested Shamir can be calculated simply multiplying the size required from each level. The following formulation shows the mathematical law:

$$Size^{Shamir}_{chunks} = (p_1+r_1) \times \cdots \times (p_n+r_n) \times (2 \times (1+\frac{1}{3}))^n Size_{in} \tag{9.4.8}$$

if $p_1 = \cdots = p_n = p$ and $r_1 = \cdots = r_n = r$ the previous formulation can be written as follows:

$$Size^{Shamir}_{chunks} = (p+r)^n \times (2 \times (1+\frac{1}{3}))^n Size_{in} \tag{9.4.9}$$

that is equivalent to:

$$Size^{Shamir}_{chunks} = ((p+r) \times 2 \times (1+\frac{1}{3}))^n Size_{in} \tag{9.4.10}$$

Considering $p_1 = p_2 = 5$ and $r_1 = r_2 = 7$. The total amount of storage required from Nested Shamir is $\sim 1018 \times Size_{in}$, $\sim$ 3 times single level Shamir and $\sim$ 8.5 times File Replication.

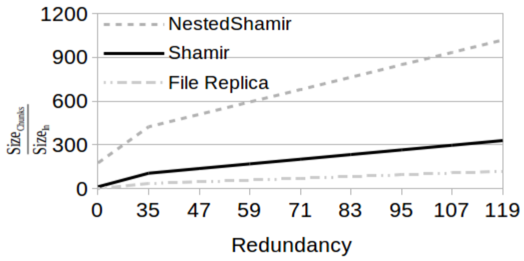


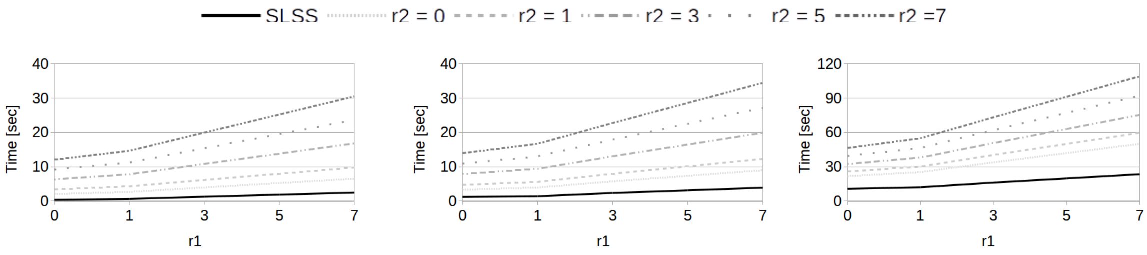
Figure 9.6: Storage requirement for different values of redundancy

As we have seen, nesting the Shamir's technique does not provide any benefit from the storage point of view, instead the Nested RRNS is able to use much less storage: $\sim$ 39% of RRNS, $\sim$ 13% of File Replication $\sim$ 5% of Shamir and only $\sim$ 1.5% of Nested Shamir.

9.4.5 Execution Time

In this Section, we investigated the execution time of the proposed solutions for the 'split' and the 'merge' functions. Differently from 'redundancy' and 'retrieval rate', the execution time depends on the Secret Share algorithm selected. As we discussed in Section 6.3, Blakely's schema and PSS can be re-conduct to Shamir. Therefore, in this Section, we evaluated only Shamir and RRNS.

As we discussed in Section 9.4.1, the split task is performed on board of drones, the formulation 9.3.11, presented in Section 9.3, represent the behavior. Therefore the heaviest case is $r1 = 7, r2 = 7$.

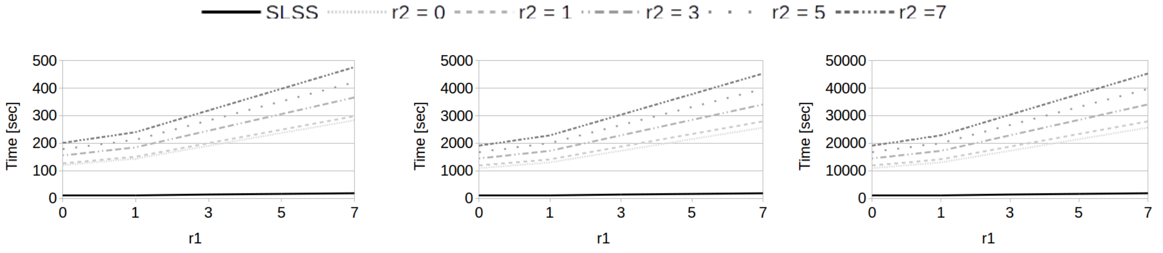


(a) File Size = 500kB (b) File Size = 5MB (c) File Size = 50MB

Figure 9.7: Execution time of RRNS slit for different file sizes. *N.B. the scale of y is different in figure c)*

Figures 9.7(a), 9.7(b) and 9.7(c) show the execution time of the RRNS for files of 500kB, 5MB and 50MB respectively. As the reader can observe, the time required from Nested RRNS is roughly 10 times the time required from the Single Level RRNS for the 500 kB file. This gap decreases at the increasing of the file size. That is, considering the 5MB file it is equal to 7, for the 50MB file it is 5. This because the execution of the second level Nested RRNS is executed on smaller files (about 0.25 times, as we discussed in Section 9.4.4).

Figures 9.8(a), 9.8(b) and 9.8(c) show the execution time of Shamir algorithm for files of 500kB, 5MB and 50MB respectively.



(a) File Size = 500kB (b) File Size = 5MB (c) File Size = 50MB

Figure 9.8: Execution time of RRNS slit for different file sizes. *N.B. the scale of y is different in all figures*

Differently from RRNS, as we discussed in Section 9.4.4, the execution of the second level of Shamir split is done on chunks that are bigger than the input file, therefore the execution time of the Nested Shamir increases exponentially.

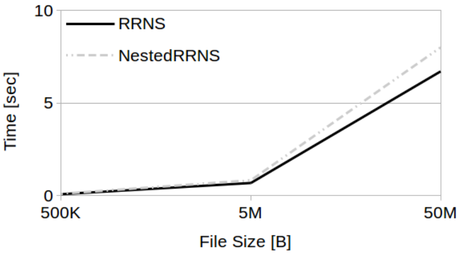


Figure 9.9: Time execution for merge function of RRNS and Nested RRNS

Figure 9.9 shows the time required for the 'merge' function of RRNS and Nested RRNS. Unlike the 'split' case, we did not considered the value of the redundancy for 'merge', because RRNS reads only the minimum number of chunks (i.e., p) and performs the CRT to recompose the original file. As we discussed at the beginning of this Section, the recomposition task is performed inside the drones base which runs p parallel instances of RRNS. Therefore, the execution time follows the formulation 9.3.14 and the overhead introduced from the nested computation is negligible for 500kB and 5MB.

Figure 9.10 shows the time required from Shamir's approaches for the 'merge' function. Also in this case the recomposition is made using the minimum number of chunks in a parallelised scenario. As a well as for the split process, the algorithm has to process chunks

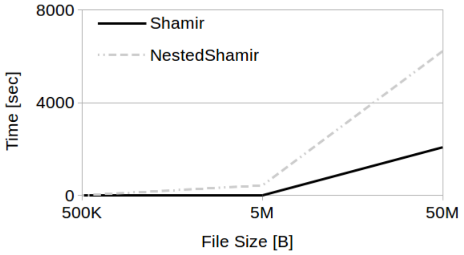


Figure 9.10: Time execution for merge function of Shamir and Nested Shamir

bigger than the input file, therefore the overhead introduced is not negligible and increases for bigger files.

9.4.6 Mission Time

The 'mission time' is the time required by the fleet to take photos and to share chunks among drones. It should be as small as possible because, during that phase, the drones are closer to enemy and the probability to be captured is higher. Bigger fleets ensure lower mission time because each drone has to process a less amount of photos. That is, considering 1000 photos per mission, if the fleet is composed of 1000 drones each of them has to process only a photo, therefore the 'mission time' of Nested RRNS and Nested Shamir are equal to times shown in Figures 9.7 and 9.8 respectively. If the fleet is composed of 100 drones the time required is 10 times bigger: roughly 17 min for Nested RRNS and ∼5 days for Nested Shamir in the slowest case (file size = 50M and $r1 = r2 = 7$), however this value is still acceptable for RRNS but too high for Shamir. Considering the fleet composed of 10 drones the mission time of Nested RRNS is still acceptable ∼2h and 47 min whereas Nested Shamir instead requires more than 7 days and 3 hours.

9.4.7 Power Consumption

The capacity of drones battery on air is limited, greater mission times require bigger batteries to power on drones. In this Section we investigated this aspect in order to evaluate the real applicability of NSS techniques. In order to make our calculation we used the benchmark from [164] considering as reference value the Raspberry pi 3 at 400% CPU load.

Figure 9.11 and 9.12 show the current needed to split each photo by using RRNS and Shamir comparing the classical version to the nested one. As a well as for the splitting time, the power consumption depends on the fleet size. The best case is the scenario with 1000 drones, which requires less than 30 mA for RRNS and almost 8 A for Shamir. Reducing the

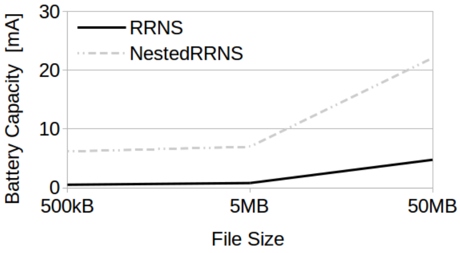


Figure 9.11: Battery capacity needed to split a photo from RRNS approaches with $p_1 = p_2 = 5$ and $r_1 = r_2 = 7$

fleet size to 100 drones increases the value of current needed to almost 300 mA for RRNS and 80 A for Shamir. Considering the fleet composed of 10 drones the current required is 3 A for RRNS and 800 A for Shamir.

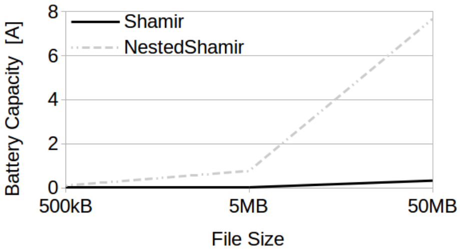


Figure 9.12: Battery capacity needed to split a photo from Shamir's approaches with $p_1 = p_2 = 5$ and $r_1 = r_2 = 7$

9.4.8 Discussion

In this Section, we will summarise our findings of neck-to-neck comparison of Nested Secret Share techniques with regards to their redundancy, retrieval rate, execution time, power consumption and storage requirements in relation to the file-size, algorithm, redundancy degree, etc.

Experiments have shown that Redundancy and Retrieval rate do not depend on the Secret Share algorithm. In particular, we have seen that the total redundancy level, contrarily to the SS that depends only on r, depends also on p of the previous level. The Retrieval Rate, of course depends on the total redundancy level, we have seen that by using lower values of r make more difficult the recomposition of photos. Execution time, power consumption and storage required, instead, depend on the Secret Share technique adopted. In this work we compared RRNS and Shamir. With regards to the execution time, we have considered the time required for split and merge of photos. We have seen that the overhead introduced from the Nested RRNS decreases at the increasing of the size of the input file, instead Shamir behaves in the opposite way. From the splitting time depends the power consumption and the capacity of batteries needed to power on drones. For each photo Nested RRNS requires a few of mA whereas Nested Shamir 8 A. This big difference makes the Nested Shamir approach not adequate for splitting and sharing pictures among drones. That is, considering to process 100 photos per drone, the capacity of the battery, for Nested Shamir, should be 800 A.

Finally, with regards to the storage required we have seen that Nested RRNS is able to use much less storage; that is, $\sim$ 39% of RRNS, $\sim$ 13% of File Replication $\sim$ 5% of Shamir and only $\sim$ 1.5% of Nested Shamir.

9.5 Conclusions and Future Work

In this chapter, we discussed the use of Nested Secret Share techniques for splitting and storing photos. We introduced the Nested Secret Share to overcome security limitations of Secret Share techniques. That is, considering SS algorithms the number of primary is fixed and depends on the implementation, by using Nested Secret Share techniques the total number of primaries can be choose arbitrarily. We evaluated these techniques considering 5 different aspects including redundancy, retrieval rate, execution time, power consumption and storage requirements.

In future works, we plan to design and implement a mathematical model for the optimal distribution of chunks among drones.

CHAPTER 10

An IoT Cloud System for Traffic Monitoring and Vehicular Accidents Prevention Based on Mobile Sensor Data Processing

The sudden traffic slowdown especially in fast scrolling roads and highways characterised by a scarce visibility is one of the major causes of accidents among motorised vehicles. It can be caused by other accidents, work-in-progress on roads, excessive motorised vehicles especially at peak times and so on. Typically, fixed traffic sensors installed on roads that interact with drivers' mobile App through the 4G network can mitigate such a problem, but unfortunately not all roads and highways are equipped with such devices. In this chapter, we discuss a possible alternative solution for addressing such an issue considering mobile traffic sensors directly installed in private and/or public transportation and volunteer vehicles. In this scenario a fast real-time processing of big traffic data is fundamental to prevent accidents. In particular, we discuss an IoT Cloud system for traffic monitoring and alert notification based on OpenGTS and MongoDB. Our IoT Cloud system, besides for private drivers, it is very useful for drivers of critical rescue vehicles such as ambulances. Experiments prove that our system provides acceptable response times that allows drivers to receive alert messages in useful time so as to avoid the risk of possible accidents.

10.1 Introduction

Motorized vehicular accidents are among the major causes of human injury or death and damage of goods with financial consequences. According to 2016 real time traffic accident statistics [165], all over the world, 853.849 was the number of people Killed in vehicular accidents, 24.100.573 was the number of injured people in traffic accidents, and $356.688.482.686 was the total cost of traffic accidents. Only in USA, the National Safety Council (NSC) [166] estimated that roughly 40.000 people died in motorized vehicles crashes in 2016, a 6% rise with respect to 2015 with a 14% increase in deaths with respect to 2014, the biggest two-year jump in more than five decades. In this panorama, all over the world, the total number of vehicles passed from roughly 921.642.000 in 2006 to 1.282.270.000 in 2015 [167] registering a meaningful growth of cars and this number is set to double by 2040 [168]. According to such a trend even the number of expected vehicular accidents is destined to increase.

In most cases, vehicular accidents are caused by avoidable human errors and improper driving practices. With recent advances in sensing technologies, self-driving, connected cars and autonomous vehicles are becoming more and more practicable. A distributed system sharing sensor data coming from vehicles can reduce the accidents by the use of direct or indirect vehicle to vehicle (V2V), vehicle to infrastructure (V2I) and infrastructure to vehicle (I2V) interactions. Sensor technology connected with cars allows drivers to improve their driving experience. This enables warnings and precautions from a network of roadside units, functioning as stationary way points. Such information is particularly beneficial for drivers in remote areas where roads cannot be equipped with traffic sensors.

The sudden traffic slowdown especially in fast scrolling roads and highways characterized by a scarce visibility is one of the major causes of accidents among vehicles. It can be caused by other accidents, work-in-progress on roads, peaks of traffic, and so on. An insufficient visibility of drivers can be due to different reasons including tight corners, fog, tunnels with scarce lighting, and so on. Typically, solutions such as Google Maps collecting traffic data coming from sensors installed on roads and sending alert messages to users' mobile apps through the 4G network can mitigate such a problem, but unfortunately not all the roads are equipped with such traffic sensors. In this context, a great progress of technologies makes Intelligent Transportation System (ITS) services more and more desirable.

In this chapter, we discuss a possible alternative ITS solution for addressing such an issue considering mobile GPS-based traffic sensors directly installed in private/public transportation and other volunteer vehicles. In this scenario, a fast real time processing of big traffic data is fundamental to prevent accidents. In particular, considering a real reference scenario of mobile GPS-based sensors installed in private/public transportation and volunteer vehicles, we discuss an IoT Cloud system for traffic monitoring and alert notification based on OpenGTS and MongoDB for the fast processing of big traffic data. Such a system, besides for private drivers, it is very useful for drivers of critical rescue vehicles such as ambulances. Assuming that the latency of the 4G network used to send traffic data from public transportation vehicles to the IoT Cloud system (V2I) and the latency required to send alert messages from the IoT Cloud to driver's mobile APP (I2V) is constant, experiments conducted on MongoDB considering datasets of variable sizes prove the goodness of our system in terms of data insertion and retrieval. In fact, the overall system allows drivers' mobile Apps to receive alert messages in a useful time in order to avoid the risk of accidents.

The IoT Cloud system is aimed at traffic monitoring and alert notification for drivers. Mobile sensors are represented by GSM/GPRS/GPS TK103 tracker based systems installed in vehicles that collect geo-localization and speed data that are sent to a OpenGTS server that stores them in a SQL database, providing a real-time OpenStreetMap visualization of the traffic scenario. In addition incoming unstructured geo-location data are forwarded to a GeoJSON parsing micro-service in order to be inserted in a MongoDB database. Another micro-service has the role of querying how the positions of vehicles change in a given area close to drivers who come from neighbour locations in order to notify their mobile Apps with alert messages related to possible sudden traffic slowdown events. The data transfer is performed by means of a 4G network connection. In order to take the advantage of a scalable Cloud-based infrastructure both OpenGTS server and micro-services are deployed by means of Docker containers so as to take the advantages of resource virtualization.

The reminder of the chapter is organized as follows. Section 10.2 presents background and related works. In Section 10.3, we motivate our work by describing a real example of accident caused by a sudden traffic slowdown in highway by considering a real report of the police of Messina (Italy). A description of our IoT Cloud system architecture is provided in Section 10.4. Experiments focusing on the MongoDB traffic data insertion and query processing are discussed in Section 10.5. Conclusion and lights to the future are summarized in Section 10.6.

10.2 Background and Related Work

Total car traffic control is the primary objective of any traffic authority. Considering an urban context, the idea of traffic control raises many application scenarios. Traffic regulation has a subsequent positive impact in terms of environment, accident prevention, speeding up vehicular circulation, optimizing public transportation, and social aspects.

Traffic congestion is a severe problem on European highways. According to a study of the European Commission, its impact will increase even up to 2% every year [169]. Since building new infrastructure is no longer an appropriate option in most (western) countries, there are many approaches towards a more effective road usage and a more intelligent way of increasing the capacity of the road network. Examples of advanced traffic control systems are, e.g., intelligent speed limits, adaptive ramp metering, or dynamic routing. These examples are based on a centralized traffic management, which controls the operation and the response according to a given traffic situation.

A local strategy based on autonomous vehicles, which are equipped with adaptive cruise control (ACC) systems is discussed in [170] and [171]. The basic idea is that a jam-avoiding driving strategy of automated vehicles might also help to increase the road capacity and thus decrease the traffic congestion. Moreover, ACC systems become commercially available to an increasing number of vehicle types. Recently some car makers have equipped, vehicles, with communication capabilities that enable inter-vehicle communication (IVC). Distance ranging principles are based on flight time. Once flight is known, the distance is computed by means of sound speed in the air. Detecting distance influences the choice of operating frequency that is over 40 kHz and less than 500 kHz to avoid absorption, using for instance beamforming that is one of most important techniques used in car traffic control and accident prevention.

Data recovery and data processing are also a big issue especially for the exponentially increasing of information to be stored and dynamically recovered to control and manage vehicular traffic [172]. However, in case of high traffic congestion and accident, the need of precise distance retrieval is required and it is a challenge for urban police authorities. Music and Esprit techniques [173] are used for this purpose but they display some limitations to more close distance to be recovered and create an overlapping of objects and obstacles. To avoid this, a specific algorithm taking inspiration from sparsity matrix is envisaged in [174]. Moreover, urban scenarios can be tailored by different passive active sensing systems, namely dedicated cameras, working in clear and infrared vision, and other sensing systems not having cameras by capable of creating microwave and/or acoustic imaging [175]. All the aforementioned outlook brings to big data and Internet issues.

The role of vehicular Cloud computing in road traffic management and internet of things (IoT) in connected vehicles is becoming more and more important as motivated in [176], [177] and [178]. Integration challenges of ITS with connected vehicles, Cloud computing, and IoT technologies is discussed in [179]. In particular, issues such as high fuel prices, high levels of CO2 emissions, increasing traffic congestion, and improved road safety are discussed. A distributed storage video Cloud (DSVC) for ITS video storage aimed at high-efficient and reliable video surveillance is discussed in [180]. Instead, a Cloud-supported gateway model for Internet access in intelligent transportation systems is discussed in [181]. According to such an approach, the heavy load of gateway government, including gateway registration, discovery, selection, dispatching, and handoff, is offloaded from the clients to the appointed Cloud servers. A platoon-based cooperative adaptive cruise control for achieving active safe driving through mobile vehicular Cloud computing is discussed in [182]. In particular, it is proposed a cooperative adaptive driving (CAD) approach that consists of three contributions: cooperative vehicle platooning (CVP), shockwave-avoidance driving (SAD), and adaptive platoon synchronization (APS). Intelligence transportation service using a vehicular Cloud network is proposed in [183], highlighting how it is necessary to provide promising solutions to prevent an accident and traffic on road. A performance analysis of vehicular ad-hoc network using different highway traffic scenarios in Cloud computing is presented in [184]. Experiments carried on two types of highway's scenarios, i.e., varying vehicles densities and simulation time, show that selected performance metrics (i.e., throughput, E2E delay and packet loss) greatly affect in both scenarios. A big vehicular traffic data mining for accident and congestion prevention is discussed in [185]. In particular, feature selection techniques are adopted in order to find the most important predictors and performed a big data analysis on a big traffic accidents dataset identifying casualties useful to develop new traffic rules and policies, in order to prevent accidents, and increase roadway safety.

In spite of the growing interest in vehicular Cloud computing, for the best of our knowledges, no scientific works are available in literature focusing on an IoT Cloud system for traffic monitoring and alert notification based on big traffic data processing coming from mobile sensors directly installed on vehicles.

10.3 Motivation

In this Section, after an overview of popular ITS technologies for vehicular traffic monitoring, we discuss a real example of accident caused by a sudden traffic slowdown happened in Messina (Italy) and reported by the local traffic police in order to motivate how it could have been avoided by an IoT Cloud GPS-based method.

10.3.1 Vehicular Traffic Monitoring Technologies Overview

Various forms of technologies have been proposed so far for vehicular traffic monitoring. UHF and VHF wireless communications have been widely used for short and/or long-range data transmission in ITS. In particular, short-range communications can be accomplished using IEEE 802.11, whereas long-range communications can be achieved through WiMAX (IEEE 802.16), Global System for Mobile Communications (GSM), 3G and 4G. In this context, a typical example of fixed traffic monitoring system includes a gantry with a dish antenna installed in the roadbed. Other fixed traffic monitoring systems are based on Radio Frequency Identification (RFID) and on the beacon sensing technology. Inductive loops can be placed in a roadbed to detect vehicles as they pass through the loop's magnetic field. Similarly, sensing devices spread along the road can detect passing vehicles through bluetooth. Fixed audio detection devices can measure the vehicular traffic density on the road analysing cumulative sounds coming from tire, engine, engine-idling, honks and air turbulence noises. Other traffic-flow measurement and automatic incident detection systems are based on fixed cameras installed in specific points of roads. Data fusion based approaches combine the road side collected acoustic, image and sensor data in order to take the advantages of different technologies. The disadvantage of the aforementioned approaches is that they require fixed sensing devices installed on the road.

Floating Car Data (FCD), also known as floating cellular data, is a method to determine the traffic speed on the road network, based on the collection of localization data time information coming from drivers' mobile smartphones. In this context, the triangulation method analyses network data using triangulation, pattern matching or cell-sector statistics, in order to deduct traffic flow information. Vehicle re-identification is an alternative method requiring a sets of fixed detectors mounted along the road collecting data coming from devices installed in vehicles with unique serial numbers. GPS-based methods include in-vehicle satnav/GPS systems sending position data used to compute vehicle speeds. Modern methods adopt smartphones using Telematics 2.0 approaches. In the end, smartphone based rich monitoring methods are based on various sensors (e.g., accelerometer, audio, and GPS) that can be used to track traffic speed, density and jams. However, the aforementioned approached do not consider emerging systems of big data storage and analytics [48] for traffic monitoring.

In this chapter, we focus on a mobile sensing GPS-based approach consisting of an IoT Cloud system collecting big traffic data coming from mobile sensors, installed on public/private transportation buses and other volunteer vehicles, and that sends warning messages to drivers' mobile apps. This approach presents several advantages such as:

- it is inexpensive to be developed because it does not need the installation of fixed traffic sensors on roads;
- it does not require a private network for sensor data exchange because it can use the

4G network;

- installing mobile traffic sensors on private and/or public transportation vehicles, it is possible to cover a wide area of the city, even the peripheral zones characterized by little traffic;
- the system is flexible and scalable because traffic sensors can be installed on other volunteer vehicles.

10.3.2 Case Study of Vehicular Accident Caused by a Sudden Slowdown

In order to motivate our scientific work, in the following, we discuss a typical car accident that could have been avoided by means of a mobile sensing GPS-based system. The accident happened in the A20 highway, Messina-Palermo direction in a mid-range corner, in uphill on the Trapani viaduct as shown in the map of Figure 10.1. The Trapani viaduct is a typical example of dangerous road with a mid-range corner in uphill with a scarce visibility for drivers. The accident involved vehicles referred as A, B, C, D and E. Figure 10.2 shows the scheme of the final disposition of all vehicles involved in the accident, whereas Figure 10.3.a, 10.3.b, and 10.3.c show real photos highlighting the damages reported by vehicles. In the following, we cite the report of the local traffic police. *"On April 3rd, 2017 at 17:10 the vehicle E driver was travelling on the A20 highway in the Messina-Palermo direction on the Trapani viaduct placed at KM 94900, in a mid-range corner, in uphill, where the highway was restricted into a single lane and where the traffic was allowed only in the overtaking lane (left lane) due to the presence of a security barrier for a work in progress correctly signalled, when due to the characteristics of the*

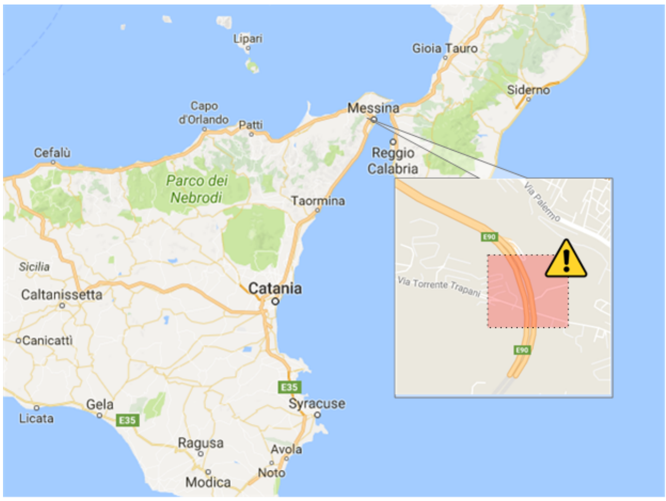


Figure 10.1: Example of dangerous tight corner in the A20 Messina-Palermo highway consisting in a mid-range corner, in uphill of the Trapani viaduct.

highway and due to a not adequate speed, it hits vehicle D that regularly stopped due to a sudden vehicular traffic slowdown. Due to the middle entity impact, vehicle E continued the drive with passiveness, hitting on the right side, vehicle C that was correctly stopped on front of it, subsequently hitting also vehicle B on the right side that was correctly stopped on front of vehicle C and in the end dabbing vehicle A that was correctly stopped due to the traffic slowdown. Thus, vehicle E after having initially rear-ended vehicle D went passively on stopping in the right lane that was closed to the traffic for work in progress. As a consequence of the accident, the driver of vehicle E was transported by ambulance to the Piemonte Hospital of Messina reporting injuries curable in 10 days, while his car was removed by a wrecker." Due to the peak of traffic many vehicles, besides the ones involved in the accident, were stopped in queue. Since it was a rush hour when many commuters travelled to go back home, it was very probable that also public and/or private transportation vehicles were also in queue. If these ones along with other volunteer vehicles had sent traffic data to an IoT Cloud service for traffic monitoring, it would have been possible to send alert messages, via a mobile app, to drivers coming in the direction of the queue.

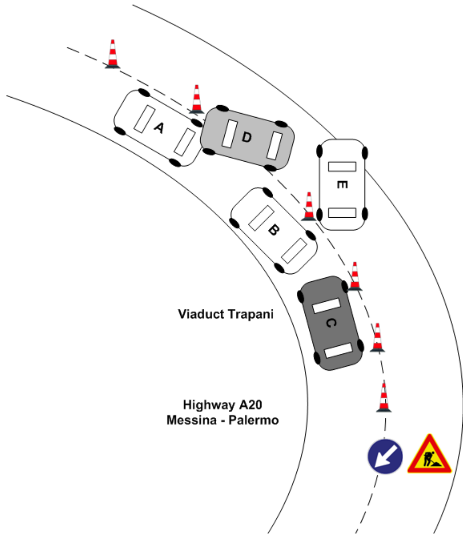
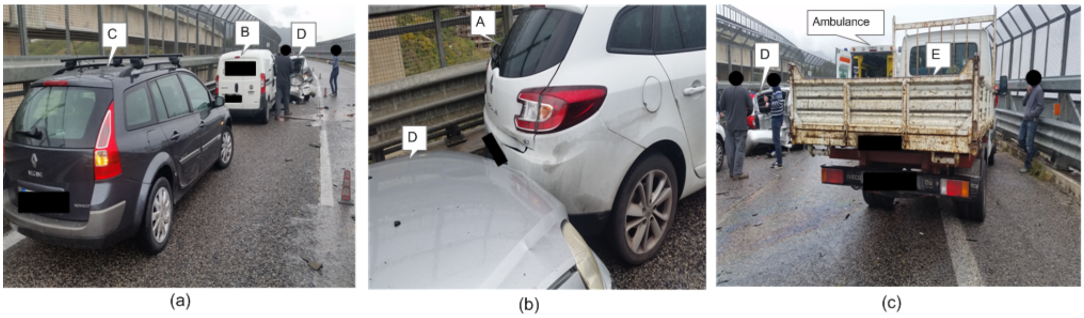


Figure 10.2: Accident scheme.



(a) (b) (c)

Figure 10.3: Accident photos.

However the success of such a system depends on response time. In fact, it is possible to prevent accidents only if the alert message arrives to the drivers in a short time. Considering that it is possible to estimate the 4G network latency, the success of the whole system depend on the capacity to process a huge amount of sensor traffic data in a short time. For this reason, a reliable and efficient big data storage and processing system is strongly required.

10.4 System Design

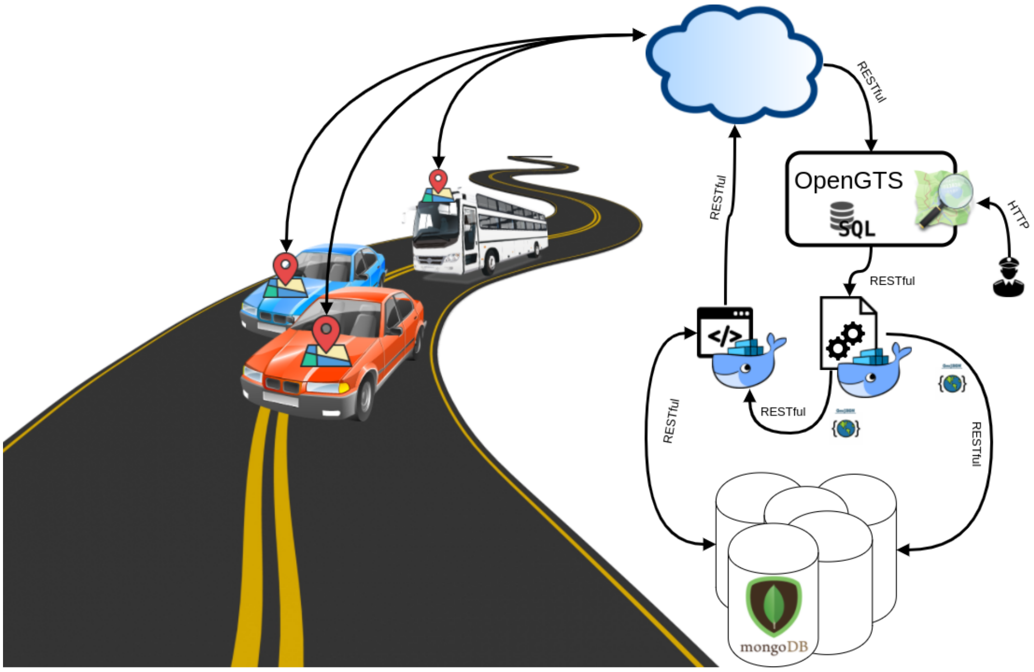
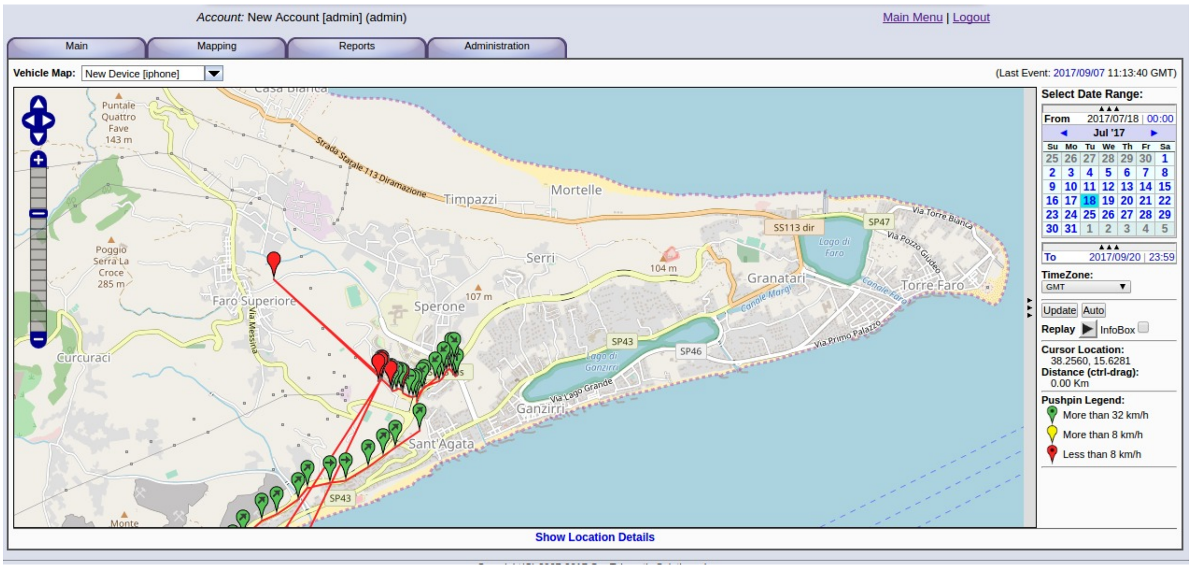


Figure 10.4: IoT Cloud system for traffic monitoring and alert notification.

In this Section, we discuss the design of the IoT Cloud system for traffic monitoring and alert notification. We define with the term "IoT Cloud" a Cloud system that integrates its storage, processing and networking capabilities with different kinds of fixed and mobile sensing assets in order to provide besides infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service levels also a new transversal type of service level called IoT as a Service (IoTaaS) [27], [18]. The adoption of the Cloud model in IoT opens toward new business opportunities [186], besides bringing different advantages in terms of service discovery [187] and security [188].

In our IoT Cloud system, mobile sensors are based on a tracker device installed in public and/or private transportation vehicles (e.g., buses, taxis, farm equipment, trucks and delivery vans) and other personal volunteer vehicles that collect data about their positions so that it is possible to calculate speed, acceleration, and other movement related data.



Account: New Account [admin] (admin)

Vehicle Map: New Device [iphone]

(Last Event: 2017/09/07 11:13:40 GMT)

Select Date Range:

From	2017/07/18 | 00:00					
Jul '17						
Su	Mo	Tu	We	Th	Fr	Sa
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

To 2017/09/20 | 23:59

TimeZone: GMT

Replay InfoBox □

Cursor Location: 38.2560, 15.6281

Distance (ctrl-drag): 0.00 Km

Pushpin Legend:

- 🟢 More than 32 km/h
- 🟡 More than 8 km/h
- 🔴 Less than 8 km/h

Figure 10.5: OpenGTS supports OpenStreetMap for visualizing a real-time traffic scenario. In this figure is shown the tracking of a single vehicle, highlighting its velocity through a green, yellow or red marker.

The IoT Cloud system requires two specific Machine-To-Machine (M2M) workflows for both gathering data from vehicles and notifying to drivers' mobile apps sudden traffic slowdown alert messages. The system explained below is Cloud-based because it is able to elastically scale up/down its internal micro-services thanks to Docker containers.

All geo-location data collected by vehicles are sent in real-time to an OpenGTS server that stores them in a SQL database, providing a real-time OpenStreetMap visualization of traffic scenario. In addition, incoming unstructured geo-location data are forwarded to a GeoJSON parsing micro-service in order to be inserted in a MongoDB distributed database. Another micro-service has the role of querying how the positions of vehicles change in a given area close to drivers who come from neighbour locations in order to notify their mobile apps with alert messages related to possible sudden traffic slowdown events. The data transfer is performed by means of a 4G network connection.

Referring to the Figure 10.4, the starting point of the data transmission are the vehicles. Each of these is equipped with a commercial satellite tracker device, such as the GSM/GPRS/GPS TK103 tracker depicted in the Figure 10.6, which measures geo-location coordinates in <latitude, longitude, and velocity> format. It supports security, positioning, monitoring, emergency alarms, and geo-localization. Its compact size and simple management is adapt for tracking vehicles or other mobile objects. Additionally, the TK103 supports SMS functions and GPRS data transmission. Data transfer functionalities allow vehicles to send such information to an OpenGTS server through a 4G connection. Such a system is currently used to track public transportation vehicles such as the ATM buses in Messina (Italy).

Figure 10.6: Example of TK103 GSM/GPRS/GPS Tracker device for vehicles.

OpenGTS is an open source project, under Apache software license, designed for providing a web-based tracking for vehicles. Specifically, the communication between the trackers and OpenGTS is implemented by means of a RESTful approach. Moreover, the spread diffusion of this technology allows to use several kinds of satellite tracking systems, proving that it is a good choice for general purpose vehicles, such as personal and public and/or private transportation vehicles.

The goal of this data gathering is twofold. From one side, OpenGTS supports the visualization of geo-located information on a Map using OpenStreetMap, a collaborative and open project. The visualized information provide a real-time picture of the traffic, allowing to monitor the high-risk areas as shown in the Figure 10.5. Such a tool is particularly useful for law enforcement and traffic operators. On the other hand, OpenStreetMap, but also our traffic notification system, needs of GeoJSON, a human and machine-readable JSON unstructured format for encoding geographic data. However, OpenGTS stores data inside a traditional structured database, such as SQL. Therefore, every time a geo-visualization is required, a SQL-GeoJSON translation is performed.

In order to avoid that, our traffic notification system needs to extract incoming data from the SQL database for performing a GeoJSON parsing task and storing data in the MongoDB database. The unstructured data guarantees a good flexibility for further data analysis and manipulation. This functionalities has been designed as a micro-service, deployed through a Docker container, for enabling elastic Cloud-based scalability. The generated GeoJSON will be also sent to another micro-service, designed for performing MongoDB query matching. Goal of this micro-service is to verify the presence of other vehicles in the same road that are at close range, notifying sudden traffic slowdown through a 4G connected the driver's mobile app.

10.5 Experimental Results

The objective of our experiments was to verify if our system prototype was able to send alert notifications in a useful time to drivers' mobile apps in order to prevent accidents. Therefore, considering constant the latency of the 4G network for sending data from vehicles to our infrastructure (V2I) and to send back the alert message from our infrastructure to vehicles (I2V) we focused our analysis on:

- data parsing and insertion;
- query time processing.

Our testbed was composed of two different types of servers on which we installed all pieces of software by means of Docker containers: one for data collection and manipulation and one for data storage. The data manipulation and collecting server, in which we run OpenGTS, was equipped with the following hardware/software configuration: Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz, RAM 32GB, OS: Ubuntu server 16.04 LTS 64 BIT. The data storage server, in which we installed MongoDB, included Intel(R) Core(TM) i3-6100 CPU @ 3.70GHz, RAM 32GB, and Ubuntu server 16.04 LTS 64 BIT. In particular, initially, MongoDB was installed with a single server configuration although it supports a distributed configuration by means of the sharding technique. The mobile APP implementation is out of the scope of this chapter.

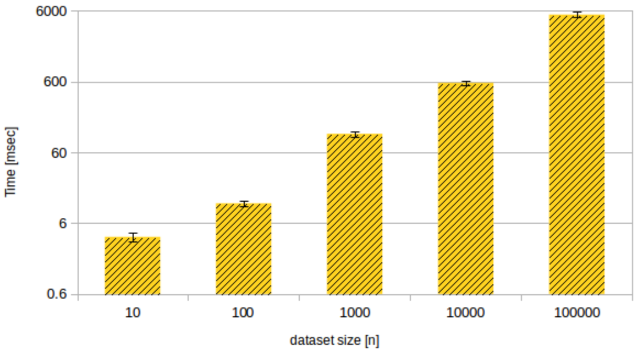


Figure 10.7: Performance parsing row to GeoJSON.

During experiments, we made scalability analysis in both scenarios. In order to have reliable results our tests were repeated 30 times. In our charts we plotted average values and confidence intervals at 95%.

Figure 10.7 shows the performance of row data to GeoJSON parsing and insertion into MongoDB. In the chart, we reported the dataset size (on the x-axis) and the response time. Values on the y-axis are expressed in milliseconds. How we can observe, response times grow up linearly as soon as we increase the size of the dataset. In particular, we highlighted that the response time for processing 100k samples is about 6 seconds.

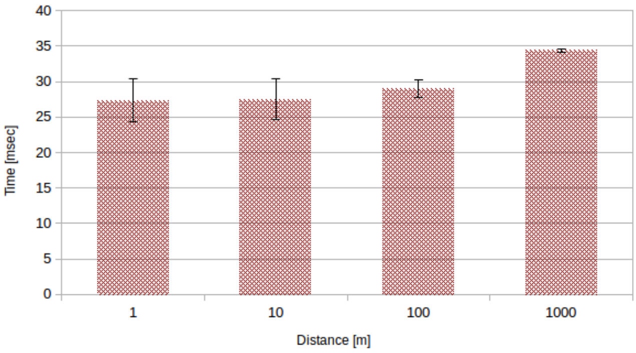


Figure 10.8: Performance query time processing.

Sampling time of GPS tracker is variable. Usually, it is fixed to 15 seconds as standard value. In our scenario, in order to have a more responsive system, we considered that each vehicle sent one piece of data every six seconds, so as to assume to have simultaneously 100000 cars in our system.

In order to assess times for retrieving data from MongoDB, we considered 300000 documents so as to have concrete values. We led our analysis in four different configurations. Specifically, we considered increasing distances from vehicles starting from 1 meter up to 1 kilometer. Figure 10.8 highlights that response times are constant being roughly 30 milliseconds.

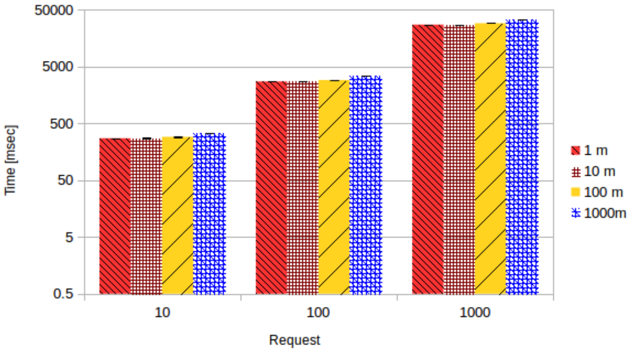


Figure 10.9: Scalability analysis of query performances.

In order to test the scalability of the deployed system, we stressed our system making increasing number of subsequent requests in in our reference scenario. In particular, we made up 1 thousand subsequent queries. Figure 10.9 shows the obtained trend. As we can observe response times grow up linearly with the number of requests.

The above discussed analysis confirms the goodness of our IoT Cloud system for traffic

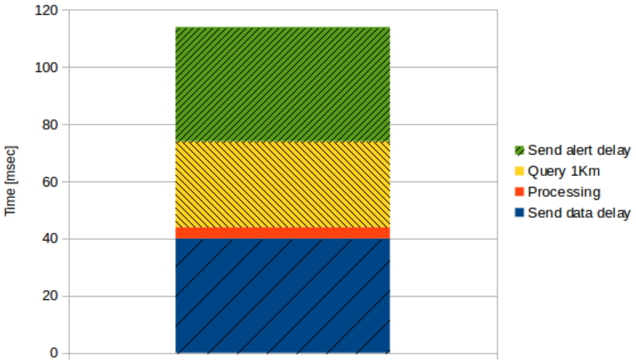


Figure 10.10: Total time.

monitoring and alert notification. In fact, considering a vehicle that travels at 120 Kmph, in order to cover 1 Km it needs 30 seconds. Our system is able to send an alert for a distance equal to 1 km in about 120 milliseconds as highlighted in 10.10. In our tests, we considered a standard 4G delay for sending data and alert equal to 40 milliseconds, indeed for processing and query time we used respectively the values reported in Figures 10.7 and 10.8. This means that our system, if used in a real scenario, can prevent accidents because it is able to notify alert messages to drivers' mobile apps in a useful time.

However, even though the aforementioned hardware/software configuration is enough to guarantee good performance, response times can be further improved by considering a distributed configuration of MongoDB in sharding mode and by scaling up the virtual hardware capabilities of Cloud-based resources according an on-demand IaaS model.

10.6 Conclusion and Future Work

Sudden traffic slowdown especially in fast scrolling roads and highways can be a serious cause of accidents among motorized vehicles. Currently, not all roads are equipped with fixed traffic sensors especially in peripherals areas of a city. In this chapter, we discussed an alternative solution consisting in an IoT Cloud system for traffic monitoring and alert notification. Geo-location and speed data are collected by a GSM/GPRS/GPS TK103 tracker based system installed in vehicles and sent to an OpenGTS server that stores them in a SQL database, providing a real-time OpenStreetMap visualization of traffic scenario. In addition, incoming unstructured geo-location data are forwarded to a GeoJSON parsing micro-service in order to be inserted in a MongoDB distributed database. Another micro-service has the role of querying how the positions of vehicles change in a given area close to drivers who come from neighbour locations in order to notify their mobile apps with alert messages related to possible sudden traffic slowdown events. The data transfer is performed by means of a 4G network connection. In order to take the advantage of a scalable Cloud-based infrastructure both OpenGTS server and micro-services were deployed by means of Docker containers. Considering that the latency of the 4G network is known, we conducted several experiments in order to compute the performance of both information insertion and retrieval in MongoDB. Obtained response times prove that our solution can be adopted in a real scenario. In future work we plan to analyse the impact of the security in our system.

CHAPTER 11

Conclusion and Future Works

In this thesis we discussed innovative systems for smart life. It is divided in two parts, in the first part we analysed basic requirements and designed the basic infrastructure; in the second part we provided solutions for specific use-cases. In particular, with reference to infrastructural requirements we analysed four aspects: in Chapter 3 we discussed the overhead introduced by containerization, we have seen that the computation overhead is acceptable considering the advantages in terms of resources partitioning; in Chapter 4 we designed and developed a broker orchestrator able to deploy distributed microserivces on multiple federated Clouds in according to constrains in terms of data export control, networking and disaster recovery defined by users; in Chapter 5 we analysed the impact of security of a Message Oriented Middleware (MOM) based on XMPP. In particular, we secured the whole communication system by using digital signature and data encryption; finally, in Chapter 6 we analysed techniques of data storage based on Secret Share. More specifically, we compared the Shamir's schema and the Redundant Residue number System (RRNS). In the second part of the thesis considering four use-cases we proposed specific Cloud-Edge based solutions. In particular, in Chapter 7 we designed and developed a Big-Data management and visualization platform for healthcare. The system allows physicians to monitors the health status of their patients simply looking at coloured circles in two modalities: generic overview and patient's specific observation. In order to validate the system we made specific scalability analyses. In Chapter 8 we proposed an innovative system for the management of Electronic Health Records (EHRs). The system, by means of Edge devices installed on ambulances and hospital beds, is able to split and merge, by using the RRNS, EHRs. The system, by storing data on public Clouds, is able to provide to physicians the most recent version of EHRs. In order to validate the system, we made specific analyses considering both RRNS (in terms of split and merge functions) and transfer time (in terms of upload and download). Performance promised and shown that execution times increase with the redundancy but they remain acceptable. In Chapter 9, considering the use-case of drones that go in mission to take pictures to enemies, we proposed the use of Nested Secret Share techniques for the management of pictures. In particular, we employed RRNS Shamir's techniques considering multiple degrees of redundancy. We have seen that the retrieval rate depends on the value of the redundancy. In oder to verify the applicability of these techniques to drones, we analysed the power consumption and discovered that Shamir's schema requires too much energy to be used on drones. Finally, in Chapter 10, we proposed an innovative solution for the vehicular traffic monitoring and accidents prevention. The system is composed of two parts: IoT devices installed on cars that send position (latitude and longitude) and the speed of vehicles and a Cloud platform that estimates traffic conditions and sends alerts when risky situations are occurring.

Open challenges in e-health and Smart Cities concern the authenticity, confidentiality and non repudiation of data. In future works we plan to fill these gaps by using both encryption, tunneling (by means of Virtual Private Networks VPNs) and BlockChain techniques performed on IoT devices.

Bibliography

[1] Lorenzo Carnevale, Antonio Celesti, Maria Di Pietro, and Antonino Galletta. How to conceive future mobility services in smart cities according to the fiware frontiercities experience. *IEEE Cloud Computing*, 5(5):25–36, 2018. (Cited at page xiii)

[2] Antonio Celesti, Davide Mulfari, Antonino Galletta, Maria Fazio, Lorenzo Carnevale, and Massimo Villari. A study on container virtualization for guarantee quality of service in cloud-of-things. *Future Generation Computer Systems*, 99:356–364, 2019. (Cited at pages xiii e 2)

[3] Antonino Galletta, Lorenzo Carnevale, Antonio Celesti, Maria Fazio, and Massimo Villari. Boss: A multitenancy ad-hoc service orchestrator for federated openstack clouds. In *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 351–357. IEEE, 2017. (Cited at pages xiii e 2)

[4] Antonio Celesti, Maria Fazio, Antonino Galletta, Lorenzo Carnevale, Jiafu Wan, and Massimo Villari. An approach for the secure management of hybrid cloud–edge environments. *Future Generation Computer Systems*, 90:1–19, 2019. (Cited at pages xiii e 3)

[5] Antonino Galletta, Javid Taheri, and Massimo Villari. On the applicability of secret share algorithms for saving data on iot, edge and cloud devices. In *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 14–21. IEEE, 2019. (Cited at pages xiii e 3)

[6] A. Galletta, L. Carnevale, A. Bramanti, and M. Fazio. An innovative methodology for big data visualization for telemedicine. *IEEE Transactions on Industrial Informatics*, 15(1):490–497, Jan 2019. (Cited at pages xiv, 3 e 102)

[7] Galletta Antonino; Buzachis Alina; Fazio Maria; Celesti Antonio; Taheri Javid; Villari Massimo. Smart hospitals enabled by edge computing. 2019. Under review. (Cited at pages xiv e 4)

[8] Galletta Antonino; Taheri Javid; Fazio Maria; Celesti Antonio; Villari Massimo;. Enhancing the security and privacy of drones by using nested secret share techniques. *ACM Transactions on Storage*, 2019. Under review. (Cited at pages xiv e 4)

[9] Antonio Celesti, Antonino Galletta, Lorenzo Carnevale, Maria Fazio, Aime Lay-Ekuakille, and Massimo Villari. An iot cloud system for traffic monitoring and vehicular accidents prevention based on mobile sensor data processing. *IEEE Sensors Journal*, 18(12):4795–4802, 2017. (Cited at pages xiv e 4)

[10] Giffinger Rudolf and Gudrun Haindlmaier. Smart cities ranking: An effective instrument for the positioning of the cities. *ACE: Architecture, City and Environment*, 4, 02 2010. (Cited at page 1)

[11] Majed Al-AZZAM and Malik Alazzam. Smart city and smart-health framework, challenges and opportunities. *International Journal of Advanced Computer Science and Applications*, 10:171–176, 01 2019. (Cited at page 1)

[12] Zhanlin Ji, Ivan Ganchev, and Mairtin O'Droma. A generic iot architecture for smart cities. volume 2014, pages 196–199, 01 2014. (Cited at page 1)

[13] B. Filocamo, A. Galletta, M. Fazio, J. A. Ruiz, M. A. Sotelo, and M. Villari. An innovative osmotic computing framework for self adapting city traffic in autonomous vehicle environment. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 01267–01270, June 2018. (Cited at page 1)

[14] Smart Government Market by Solution (Government Resource Planning System, Security, Analytics, Open Data Platform, Network Management, and Remote Monitoring), Service (Professional and Managed), Deployment, and Region - Global Forecast to 2022. `https://www.marketsandmarkets.com/Market-Reports/smart-government-market-60917358.html?gclid=CjwKCAiA9JbwBRAAEiwAnWa4Q1NRm2B228XS6OdJ9Yf8H5_`

a8nP0J-Q-tDwmnAO4d7y9HHswtXDk8hoCsMMQAvD_BwE. Last access: November 2019. (Cited at page 1)

[15] Worldwide market forecasts and analysis (20142019). https://www.marketsandmarkets.com/Market-Reports/advanced-analytics-market-58104148.html. Last access: November 2019. (Cited at page 1)

[16] Smart Cities Will Include 10 Billion Things by 2020 - Start Now to Plan, Engage and Position Offerings. https://www.gartner.com/en/documents/3004417/smart-cities-will-include-10-billion-things-by-2020-star. Last access: November 2019. (Cited at page 1)

[17] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011. (Cited at page 6)

[18] A. Celesti, M. Fazio, M. Giacobbe, A. Puliafito, and M. Villari. Characterizing cloud federation in iot. In *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 93–98, March 2016. (Cited at pages 13, 18, 32 e 146)

[19] Gartner Glossary: Edge Computing. https://www.gartner.com/en/information-technology/glossary/edge-computing. Last access: November 2019. (Cited at page 14)

[20] C. Mahmoudi, F. Mourlin, and A. Battou. Formal definition of edge computing: An emphasis on mobile cloud and iot composition. In *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 34–42, April 2018. (Cited at page 14)

[21] Gartner Glossary: Microservice. https://www.gartner.com/en/information-technology/glossary/microservice. Last access: November 2019. (Cited at page 14)

[22] Anil Karmel, Ramaswamy Chandramouli, and Michaela Iorga. Nist definition of microservices, application containers and system virtual machines. Technical report, National Institute of Standards and Technology, 2016. (Cited at page 14)

[23] What are microservices? https://www.redhat.com/en/topics/microservices/what-are-microservices. Last access: November 2019. (Cited at page 15)

[24] Amy Nordrum, Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated. `http://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by%2D2020-is-outdated`. Last access: November 2019. (Cited at page 17)

[25] A. Celesti, M. Fazio, M. Villari, M. Giacobbe, and A. Puliafito. Characterizing iot cloud federation. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications Workshops - Workshop on Cloud Computing Project and Initiatives - CCPI'16*. IEEE Computer Society, March 2016. (Cited at page 17)

[26] R. Morabito. A performance evaluation of container technologies on internet of things devices. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 999–1000, 2016. (Cited at pages 17 e 18)

[27] A. Celesti, D. Mulfari, M. Fazio, M. Villari, and A. Puliafito. Exploring container virtualization in iot clouds. In *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–6, 2016. (Cited at pages 17, 18 e 146)

[28] Mirjana Maksimović, Vladimir Vujović, Nikola Davidović, Vladimir Milošević, and Branko Perišić. Raspberry pi as internet of things hardware: Performances and constraints. *design issues*, 3:8, 2014. (Cited at page 18)

[29] R. Morabito. Virtualization on internet of things edge devices with container technologies: A performance evaluation. *IEEE Access*, 5:8835–8850, 2017. (Cited at page 18)

[30] C. Pahl, S. Helmer, L. Miori, J. Sanin, and B. Lee. A container-based edge cloud paas architecture based on raspberry pi clusters. In *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pages 117–124, 2016. (Cited at page 18)

[31] T. Renner, M. Meldau, and A. Kliem. Towards container-based resource management for the internet of things. In *2016 International Conference on Software Networking (ICSN)*, pages 1–5, May 2016. (Cited at page 18)

[32] Y. Park, H. Yang, T. Dinh, and Y. Kim. Design and implementation of a container-based virtual client architecture for interactive digital signage systems. *International Journal of Distributed Sensor Networks*, 13(7), 2017. (Cited at page 18)

[33] A. Krylovskiy. Internet of things gateways meet linux containers: Performance evaluation and discussion. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 222–227, Dec 2015. (Cited at page 18)

[34] C. Dupont, R. Giaffreda, and L. Capra. Edge computing in iot context: Horizontal and vertical linux container migration. In *2017 Global Internet of Things Summit (GIoTS)*, pages 1–4, June 2017. (Cited at page 18)

[35] R. Morabito, R. Petrolo, V. Loscri, and N. Mitton. Enabling a lightweight edge gateway-as-a-service for the internet of things. In *2016 7th International Conference on the Network of the Future, NOF 2016*, 2017. (Cited at page 18)

[36] Design and implementation of a novel service management framework for iot devices in cloud. *Journal of Systems and Software*, 119:149 – 161, 2016. (Cited at page 19)

[37] Efficient auto-scaling scheme for rapid storage service using many-core of desktop storage virtualization based on iot. *Neurocomputing*, 209:67 – 74, 2016. (Cited at page 19)

[38] Developing a distributed software defined networking testbed for iot. *Procedia Computer Science*, 83:680 – 684, 2016. The 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016) / The 6th International Conference on Sustainable Energy Information Technology (SEIT-2016) / Affiliated Workshops. (Cited at page 19)

[39] S. Alam, M. M. R. Chowdhury, and J. Noll. Senaas: An event-driven sensor virtualization approach for internet of things cloud. In *2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications*, pages 1–6, 2010. (Cited at page 19)

[40] M.G. Xavier, M.V. Neves, F.D. Rossi, T.C. Ferreto, T. Lange, and C.A.F. De Rose. Performance evaluation of container-based virtualization for high performance computing environments. In *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*, pages 233–240, Feb 2013. (Cited at page 19)

[41] Matt Richardson and Shawn Wallace. *Getting Started with Raspberry Pi*. " O'Reilly Media, Inc.", 2012. (Cited at page 22)

[42] Nogal Memari, Shaiful Jahari B. Hashim, and Khairulmizam B. Samsudin. Towards virtual honeynet based on lxc virtualization. In *Region 10 Symposium, 2014 IEEE*, pages 496–501, April 2014. (Cited at page 23)

[43] Mayra Samaniego and Ralph Deters. Using blockchain to push software-defined iot components onto edge hosts. In *Proceedings of the International Conference on Big Data and Advanced Wireless Technologies*, BDAW '16, pages 58:1–58:9, New York, NY, USA, 2016. ACM. (Cited at page 23)

[44] Antonio Celesti, Anna Levin, Philippe Massonet, Liran Schour, and Massimo Villari. *Federated Networking Services in Multiple OpenStack Clouds*, pages 338–352. Springer International Publishing, Cham, 2016. (Cited at page 31)

[45] F. Tusa, A. Celesti, M. Villari, and A. Puliafito. How to enhance cloud architectures to enable cross-federation. In *Proceedings of IEEE CLOUD '10*, pages 337–345. IEEE, July 2010. (Cited at page 32)

[46] G. Vernik, A Shulman-Peleg, S. Dippl, C. Formisano, M.C. Jaeger, E.K. Kolodner, and M. Villari. Data on-boarding in federated storage clouds. In *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*, pages 244–251, June 2013. (Cited at page 32)

[47] I. Goiri, J. Guitart, and J. Torres. Characterizing cloud federation for enhancing providers' profit. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 123 –130, july 2010. (Cited at page 32)

[48] M. Fazio, A. Celesti, M. Villari, and A. Puliafito. The need of a hybrid storage approach for iot in paas cloud federation. In *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, pages 779–784, 2014. (Cited at pages 32 e 143)

[49] S. Azodolmolky, P. Wieder, and R. Yahyapour. Cloud computing networking: challenges and opportunities for innovations. *Communications Magazine, IEEE*, 51(7):54–62, July 2013. (Cited at page 32)

[50] M. Villari, M. Fazio, S. Dustdar, O. Rana, and R. Ranjan. Osmotic computing: A new paradigm for edge/cloud integration. *IEEE Cloud Computing*, 3(6):76–83, Nov 2016. (Cited at page 32)

[51] H. Benfenatki, C. Ferreira Da Silva, G. Kemp, A.-N. Benharkat, P. Ghodous, and Z. Maamar. Madona: a method for automated provisioning of cloud-based component-oriented business applications. *Service Oriented Computing and Applications*, 11(1):87–100, 2017. (Cited at page 32)

[52] A. Ceselli, M. Premoli, and S. Secci. Mobile edge cloud network design optimization. *IEEE/ACM Transactions on Networking*, 2017. (Cited at page 33)

[53] D. Pizzolli, G. Cossu, D. Santoro, L. Capra, C. Dupont, D. Charalampos, F. De Pellegrini, F. Antonelli, and S. Cretti. Cloud4iot: A heterogeneous, distributed and autonomic cloud platform for the iot. In *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom*, pages 476–479, 2017. (Cited at page 33)

[54] K. Yongsiriwit, M. Sellami, and W. Gaaloul. A semantic framework supporting cloud resource descriptions interoperability. In *IEEE International Conference on Cloud Computing, CLOUD*, pages 585–592, 2017. (Cited at page 33)

[55] A. Sampaio, T. Rolim, N.C. MendonÃ§a, and M. Cunha. An approach for evaluating cloud application topologies based on tosca. In *IEEE International Conference on Cloud Computing, CLOUD*, pages 407–414, 2017. (Cited at page 33)

[56] A. Mayoral, R. Vilalta, R. Munoz, R. Casellas, R. Martinez, M.S. Moreolo, J.M. Fabrega, A. Aguado, S. Yan, D. Simeonidou, J.M. Gran, V. Lopez, P. Kaczmarek, R. Szwedowski, T. Szyrkowiec, A. Autenrieth, N. Yoshikane, T. Tsuritani, I. Morita, M. Shiraiwa, N. Wada, M. Nishihara, T. Tanaka, T. Takahara, J.C. Rasmussen, Y. Yoshida, and K.-I. Kitayama. Control orchestration protocol: Unified transport api for distributed cloud and network orchestration. *Journal of Optical Communications and Networking*, 9(2):A216–A222, 2017. (Cited at page 33)

[57] J. Park, Y. An, and K. Yeom. Virtual cloud bank: Cloud service broker for intermediating services based on semantic analysis models. In *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, pages 1022–1029, Aug 2015. (Cited at page 34)

[58] G.B. Fioccola, P. Donadio, R. Canonico, and G. Ventre. Dynamic routing and virtual machine consolidation in green clouds. In *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom*, pages 590–595, 2017. (Cited at page 34)

[59] Security Guidance for Critical Areas of Focus in Cloud Computing, v3.0. `https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf`. Last access: November 2019. (Cited at page 45)

[60] A. Celesti, M. Fazio, and M. Villari. Se clever: A secure message oriented middleware for cloud federation. pages 35–40, 2013. cited By 21. (Cited at page 45)

[61] A. Celesti, M. Fazio, and M. Villari. Enabling secure xmpp communications in federated iot clouds through xep 0027 and saml/sasl sso. *Sensors (Switzerland)*, 17(2), 2017. cited By 3. (Cited at page 45)

[62] Extensible Messaging and Presence Protocol (XMPP). `http://xmpp.org/`. Last access: November 2019. (Cited at page 46)

[63] XEP-0373: Current Jabber OpenPGP Usage. `https://xmpp.org/extensions/attic/xep-0373-0.1.2.html`. Last access: November 2019. (Cited at pages 46 e 50)

[64] OpenPGP Message Format. `http://www.rfc-editor.org/info/rfc4880`. Last access: November 2019. (Cited at pages 46 e 51)

[65] A. Celesti, F. Tusa, M. Villari, and A. Puliafito. Integration of clever clouds with third party software systems through a rest web service interface. pages 000827–000832, 2012. cited By 20. (Cited at page 46)

[66] S.N. Shirazi, A. Gouglidis, A. Farshad, and D. Hutchison. The extended cloud: Review and analysis of mobile edge computing and fog from a security and resilience perspective. *IEEE Journal on Selected Areas in Communications*, 35(11):2586–2595, 2017. (Cited at page 46)

[67] C. Esposito, A. Castiglione, F. Pop, and K.-K.R. Choo. Challenges of connecting edge and cloud computing: A security and forensic perspective. *IEEE Cloud Computing*, 4(2):13–17, 2017. (Cited at page 46)

[68] S. Rathore, P.K. Sharma, A.K. Sangaiah, and J.J. Park. A hesitant fuzzy based security approach for fog and mobile-edge computing. *IEEE Access*, 6:688–701, 2017. cited By 0. (Cited at page 46)

[69] G. Li, H. Zhou, B. Feng, G. Li, T. Li, Q. Xu, and W. Quan. Fuzzy theory based security service chaining for sustainable mobile-edge computing. *Mobile Information Systems*, 2017, 2017. cited By 1. (Cited at page 46)

[70] W. Abdul, Z. Ali, S. Ghouzali, B. Alfawaz, G. Muhammad, and M.S. Hossain. Biometric security through visual encryption for fog edge computing. *IEEE Access*, 5:5531–5538, 2017. (Cited at page 47)

[71] V. Vassilakis, I.P. Chochliouros, A.S. Spiliopoulou, E. Sfakianakis, M. Belesioti, N. Bompetsis, M. Wilson, C. Turyagyenda, and A. Dardamanis. Security analysis of mobile edge computing in virtualized small cell networks. *IFIP Advances in Information and Communication Technology*, 475:653–665, 2016. (Cited at page 47)

[72] V. Vassilakis, E. Panaousis, and H. Mouratidis. Security challenges of small cell as a service in virtualized mobile edge computing environments. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9895 LNCS:70–84, 2016. (Cited at page 47)

[73] Cloud Security Alliance. `http://www.cloudsecurityalliance.org/`. Last access: November 2019. (Cited at page 47)

[74] Elisa Bertino, Federica Paci, Rodolfo Ferrini, and Ning Shang. Privacy-preserving digital identity management for cloud computing. *Computer*, 32(1):21–27, March 2009. (Cited at page 50)

[75] XML Digital Signature API (XMLDsig). `http://docs.oracle.com/javase/7/docs/technotes/guides/security/xmldsig/XMLDigitalSignature.html`. Last access: March 2019. (Cited at page 57)

[76] EJBCA PKI by Primekey. `http://www.ejbca.org`. Last access: March 2019. (Cited at page 57)

[77] Java Simple Certificate Enrollment Protocol (JSCEP). `https://code.google.com/p/jscep`. Last access: March 2019. (Cited at page 57)

[78] OpenLDAP. `http://www.openldap.org`. Last access: March 2019. (Cited at page 57)

[79] Java Naming and Directory Interface (JNDI). `http://www.oracle.com/technetwork/java/jndi/index.html`. Last access: March 2019. (Cited at page 57)

[80] The Legion of the Bouncy Castle. `http://www.bouncycastle.org`. Last access: March 2019. (Cited at page 57)

[81] M. Villari, M. Fazio, S. Dustdar, O. Rana, and R. Ranjan. Osmotic computing: A new paradigm for edge/cloud integration. *IEEE Cloud Computing*, 3(6):76–83, 2016. cited By 18. (Cited at page 66)

[82] More Than 1 Million Google Accounts Breached by Gooligan. http://blog.checkpoint.com/2016/11/30/1-million-google-accounts-breached-gooligan/. Last access: March 2019. (Cited at pages 68 e 101)

[83] Yahoo hack: 1bn accounts compromised by biggest data breach in history. https://www.theguardian.com/technology/2016/dec/14/yahoo-hack-security-of-one-billion-accounts-breached. Last access: March 2019. (Cited at pages 68 e 101)

[84] Stolen Dropbox passwords are circulating online. Here's how to check if your account's compromised. http://qz.com/771196/stolen-dropbox-passwords-are-circulating-online-heres-how-to%2Dcheck-if-your-accounts-compromised/. Last access: March 2019. (Cited at pages 68 e 101)

[85] Copy Cloud Storage service's life ends on May 1, 2016. https://www.ghacks.net/2016/02/02/copy-cloud-storage-services-life-ends-on-may-1-2016/. Last access: March 2019. (Cited at pages 68 e 101)

[86] Shutting down Ubuntu One file services. https://blog.ubuntu.com/2014/04/02/shutting-down-ubuntu-one-file-services. Last access: March 2019. (Cited at pages 68 e 101)

[87] Dell DataSafe Online. https://web.archive.org/web/20140625035604/https://www.delldatasafe.com/. Last access: March 2019. (Cited at pages 68 e 101)

[88] Justice Opara-Martins, Reza Sahandi, and Feng Tian. Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective. *Journal of Cloud Computing*, 5(1):4, Apr 2016. (Cited at pages 68 e 101)

[89] S. Hsiao and D. Kao. The static analysis of wannacry ransomware. In *2018 20th International Conference on Advanced Communication Technology (ICACT)*, pages 153–158, Feb 2018. (Cited at pages 68 e 101)

[90] Liao-Jun Pang and Yu-Min Wang. A new (t,n) multi-secret sharing scheme based on shamir's secret sharing. *Applied Mathematics and Computation*, 167(2):840 – 848, 2005. (Cited at page 69)

[91] Giovanni Di Crescenzo. Sharing one secret vs. sharing many secrets: Tight bounds for the max improvement ratio. In Jiří Sgall, Aleš Pultr, and Petr Kolman, editors, *Mathematical Foundations of Computer Science 2001*, pages 292–304, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. (Cited at page 69)

[92] V. J. Kapadiya, L. S. Desai, and Y. K. Meghrajani. Boolean-based multi secret sharing scheme using meaningful shares. In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pages 840–844, April 2018. (Cited at page 69)

[93] P. R. Kamble and S. Patil. Exploring secret image sharing with embedding of shares. In *2018 2nd International Conference on Inventive Systems and Control (ICISC)*, pages 1090–1093, Jan 2018. (Cited at page 69)

[94] L. Tan, K. Liu, X. Yan, S. Wan, J. Chen, and C. Chang. Visual secret sharing scheme for color qr code. In *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, pages 961–965, June 2018. (Cited at page 69)

[95] F. Ding, Y. Long, and P. Wu. Study on secret sharing for sm2 digital signature and its application. In *2018 14th International Conference on Computational Intelligence and Security (CIS)*, pages 205–209, Nov 2018. (Cited at page 69)

[96] H. Prasetyo, D. Rosiyadi, and S. Horng. Modified generalized random grids-based progressive secret sharing with lossless ability for binary image. In *2018 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, pages 181–186, Nov 2018. (Cited at page 70)

[97] S. J. Shyu, Y. Z. Tsai, and Y. T. Tsai. Implementing secret sharing scheme in parallel. In *2018 15th International Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN)*, pages 362–365, Oct 2018. (Cited at page 70)

[98] J. Imai, M. Mimura, and H. Tanaka. Verifiable secret sharing scheme using hash values. In *2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW)*, pages 405–409, Nov 2018. (Cited at page 70)

[99] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979. (Cited at page 71)

[100] G. R. Blakley. Safeguarding cryptographic keys. In *Managing Requirements Knowledge, International Workshop on*, page 313, Los Alamitos, CA, USA, jun 1979. IEEE Computer Society. (Cited at pages 71 e 72)

[101] Alireza Shamsoshoara. Overview of blakley's secret sharing scheme. *CoRR*, abs/1901.02802, 2019. (Cited at page 72)

[102] Amir Herzberg, Stanisław Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In Don Coppersmith, editor, *Advances in Cryptology — CRYPT0' 95*, pages 339–352, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. (Cited at page 72)

[103] R. W. Watson and C. W. Hastings. Self-checked computation using residue arithmetic. *Proceedings of the IEEE*, 54(12):1920–1931, Dec 1966. (Cited at page 72)

[104] Iot number of connected devices worldwide. `https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/`. Last accessed January 2018. (Cited at page 84)

[105] Global connected wearable devices. `https://www.statista.com/statistics/487291/global-connected-wearable-devices/`. Last accessed January 2018. (Cited at page 84)

[106] Monday inspiration data visualization and infographics. `https://www.smashingmagazine.com/2008/01/monday-inspiration-data-visualization-and-infographics/`. Last accessed January 2018. (Cited at page 85)

[107] Manuela Aparicio and Carlos J. Costa. Data visualization. *Communication Design Quarterly Review*, 3(1):7–11, jan 2015. (Cited at page 85)

[108] Big data: The next frontier for innovation, competition, and productivity. `https://www.mckinsey.com/business-functions/mckinsey-digital/`

our-insights/big-data-the-next-frontier-for-innovation. Last accessed December 2019. (Cited at page 85)

[109] Matthew Berman and Andrea Fenaughty. Technology and managed care: patient benefits of telemedicine in a rural health care network. *Health Economics*, 14(6):559–573, 2005. (Cited at page 85)

[110] George Demiris, Marilyn J Rantz, Myra A Aud, Karen D Marek, Harry W Tyrer, Marjorie Skubic, and Ali A Hussam. Older adults' attitudes towards and perceptions of 'smart home' technologies: a pilot study. *Medical Informatics and the Internet in Medicine*, 29(2):87–94, jun 2004. (Cited at page 85)

[111] A. Coronato, G. De Pietro, and G. Sannino. Middleware services for pervasive monitoring elderly and ill people in smart environments. In *2010 Seventh International Conference on Information Technology: New Generations*, pages 810–815, April 2010. (Cited at page 85)

[112] D.H. Stefanov, Z. Bien, and W.-C. Bang. The smart house for older persons and persons with physical disabilities: Structure, technology arrangements, and perspectives. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 12(2):228–250, jun 2004. (Cited at page 85)

[113] Giovanna Sannino and Giuseppe De Pietro. A mobile system for real-time context-aware monitoring of patients&#39; health and fainting. *Int. J. Data Min. Bioinformatics*, 10(4):407–423, September 2014. (Cited at page 85)

[114] N. Yang, Z. Wang, R. Gravina, and G. Fortino. A survey of open body sensor networks: Applications and challenges. In *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 65–70, Jan 2017. (Cited at page 85)

[115] E M Strehle and N Shabde. One hundred years of telemedicine: does this new technology have a place in paediatrics? *Archives of Disease in Childhood*, 91(12):956–959, jul 2006. (Cited at page 85)

[116] George Kontaxakis, Dimitris Visvikis, Roland Ohl, Ilias Sachpazidis, Juan Suarez, Peter Selby, Catherine Cheze-Le Rest, Andres Santos, Fernando Ortega, Javier Diaz, Leyun Pan, Ludwig Strauss, Antonia Dimitrakopoulou-Strauss, Georgios Sakas, and Miguel Pozo. Integrated telemedicine applications and services for oncological positron emission tomography. *Oncology Reports*, apr 2006. (Cited at page 85)

[117] Raffaele Gravina, Parastoo Alinia, Hassan Ghasemzadeh, and Giancarlo Fortino. Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges. *Information Fusion*, 35:68 – 80, 2017. (Cited at page 85)

[118] M. Cavallo, G. Di Modica, C. Polito, and O. Tomarchio. A lahc-based job scheduling strategy to improve big data processing in geo-distributed contexts. pages 92–101, 2017. cited By 1. (Cited at page 85)

[119] M. Cavallo, G. Di Modica, C. Polito, and O. Tomarchio. H2f: A hierarchical hadoop framework for big data processing in geo-distributed environments. pages 27–35, 2016. cited By 2. (Cited at page 85)

[120] N M Hjelm. Benefits and drawbacks of telemedicine. *Journal of Telemedicine and Telecare*, 11(2):60–70, mar 2005. (Cited at page 86)

[121] Pedro Cardoso, Nuno Datia, and M.P.M. Pato. Integrated electromyography visualization with multi temporal resolution. In *2017 11th International Symposium on Medical Information and Communication Technology (ISMICT)*. IEEE, feb 2017. (Cited at page 86)

[122] Mohamed Adel Serhani, Mohamed El Menshawy, Abdelghani Benharref, Saad Harous, and Alramzana Nujum Navaz. New algorithms for processing time-series big EEG data within mobile health monitoring systems. *Computer Methods and Programs in Biomedicine*, 149:79–94, oct 2017. (Cited at page 86)

[123] Alexis Allot, Kirsley Chennen, Yannis Nevers, Laetitia Poidevin, Arnaud Kress, Raymond Ripp, Julie Dawn Thompson, Olivier Poch, and Odile Lecompte. MyGeneFriends: A social network linking genes, genetic diseases, and researchers. *Journal of Medical Internet Research*, 19(6):e212, jun 2017. (Cited at page 86)

[124] Q. Liu, X. Guo, H. Fan, and H. Zhu. A novel data visualization approach and scheme for supporting heterogeneous data. In *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pages 1259–1263, Dec 2017. (Cited at page 87)

[125] T. M. Frink, J. V. Gyllinsky, and K. Mankodiya. Visualization of multidimensional clinical data from wearables on the web and on apps. In *2017 IEEE MIT Undergraduate Research Technology Conference (URTC)*, pages 1–4, Nov 2017. (Cited at page 87)

[126] Joao H.G. Sa, Marina S. Rebelo, Alexandra Brentani, Sandra J.F.E. Grisi, Leonardo H. Iwaya, Marcos A. Simplicio, Tereza C.M.B. Carvalho, and Marco A. Gutierrez. Georeferenced and secure mobile health system for large scale data collection in primary care. *International Journal of Medical Informatics*, 94:91–99, oct 2016. (Cited at page 87)

[127] WannaCry cyber attack cost the NHS Â£92m as 19,000 appointments cancelled. `https://www.telegraph.co.uk/technology/2018/10/11/wannacry-cyber-attack-cost-nhs-92m-19000-appointments-cancelled/`. Last access: April 2019. (Cited at page 101)

[128] 2018 reform of eu data protection rules. `https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en`. Last accessed April 2019. (Cited at pages 101 e 104)

[129] R. K. Pathinarupothi, P. Durga, and E. S. Rangan. Iot based smart edge for global health: Remote monitoring with severity detection and alerts transmission. *IEEE Internet of Things Journal*, pages 1–1, 2019. (Cited at page 102)

[130] H. Zhang, J. Li, B. Wen, Y. Xun, and J. Liu. Connecting intelligent things in smart hospitals using nb-iot. *IEEE Internet of Things Journal*, 5(3):1550–1560, June 2018. (Cited at page 102)

[131] Chenghao He, Xi Jin, Zhanxiang Zhao, and Tian Xiang. A cloud computing solution for hospital information system. In *2010 IEEE International Conference on Intelligent Computing and Intelligent Systems*, volume 2, pages 517–520, Oct 2010. (Cited at page 102)

[132] A Porter, H Potts, S Mason, H Morgan, Z Morrison, N Rees, D Shaw, N Siriwardena, H Snooks, and V Williams. 70 the digital ambulance: electronic patient clinical records in prehospital emergency care, 2018. (Cited at page 103)

[133] T. Sigwele, Y. F. Hu, M. Ali, J. Hou, M. Susanto, and H. Fitriawan. An intelligent edge computing based semantic gateway for healthcare systems interoperability and collaboration. In *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 370–376, Aug 2018. (Cited at page 103)

[134] S. P. Khare, J. Sallai, A. Dubey, and A. Gokhale. Short paper: Towards low-cost indoor localization using edge computing resources. In *2017 IEEE 20th International Symposium on Real-Time Distributed Computing (ISORC)*, pages 28–31, May 2017. (Cited at page 103)

[135] S. Tata, R. Jain, H. Ludwig, and S. Gopisetty. Living in the cloud or on the edge: Opportunities and challenges of iot application architecture. In *2017 IEEE International Conference on Services Computing (SCC)*, pages 220–224, June 2017. (Cited at page 103)

[136] Tracy D Gunter and Nicolas P Terry. The emergence of national electronic health record architectures in the united states and australia: Models, costs, and questions. *J Med Internet Res*, 7(1):e3, Mar 2005. (Cited at page 104)

[137] Australian government department of health and ageing. `http://www.healthconnect.gov.au/pdf/bav1.pdf`. Last accessed January 2019. (Cited at page 104)

[138] Introduction to hl7 standards. `http://www.hl7.org/implement/standards/`. Last accessed January 2019. (Cited at page 105)

[139] What are the differences between electronic medical records, electronic health records, and personal health records? `https://www.healthit.gov/faq/what-are-differences-between-electronic-medical-records%2Delectronic-health-records-and-personal`. Last accessed January 2019. (Cited at page 106)

[140] Patient access to electronic health records: Differences across ten countries. *Health Policy and Technology*, 7(1):44 – 56, 2018. (Cited at page 106)

[141] Scambio di cartelle cliniche, donna rischia l'intervento chirurgico. `https://www.agro24.it/2016/03/scambio-di-cartelle-cliniche%2Ddonna-rischia-lintervento-chirurgico/`. Last accessed January 2019. (Cited at page 107)

[142] Scambio di cartelle cliniche in ospedale: morto un 62enne. `http://www.responsabilecivile.it/scambio-di-cartelle%2Dcliniche-ospedale-morto-un-62enne/`. Last accessed January 2019. (Cited at page 107)

[143] Scambio di cartelle cliniche: 70enne subisce un ciclo di chemioterapia, ma non ha un tumore e muore. `https://retenews24.it/scambio-di-cartelle-cliniche-70enne-subisce-un-ciclo-di%2Dchemioterapia-ma-non-ha-un-tumore/`. Last accessed January 2019. (Cited at page 107)

[144] Rapporto annuale sull'attività di ricovero ospedaliero. http://www.salute.gov.it/imgs/C_17_pubblicazioni_2651_allegato.pdf. Last accessed January 2019. (Cited at page 107)

[145] Sistema di emergenza sanitaria territoriale "118". rilevazione nazionale. http://www.salute.gov.it/imgs/C_17_pubblicazioni_845_allegato.pdf. Last accessed January 2019. (Cited at page 107)

[146] C. Esposito, A. De Santis, G. Tortora, H. Chang, and K. R. Choo. Blockchain: A panacea for healthcare cloud-based data security and privacy? *IEEE Cloud Computing*, 5(1):31–37, Jan 2018. (Cited at page 117)

[147] Civil drones in the European Union. http://www.europarl.europa.eu/RegData/etudes/BRIE/2015/571305/EPRS_BRI%282015%29571305_EN.pdf. Last access: September 2019. (Cited at page 121)

[148] Iran shoots down US drone aircraft, raising tensions further in Strait of Hormuz. https://edition.cnn.com/2019/06/20/middleeast/iran-drone-claim-hnk-intl/index.html. Last access: November 2019. (Cited at page 121)

[149] Dutch police will stop using drone-hunting eagles since they weren't doing what they're told. https://www.theverge.com/2017/12/12/16767000/police-netherlands-eagles-rogue-drones. Last access: November 2019. (Cited at page 121)

[150] N. C. Mhala and A. R. Pais. An improved and secure visual secret sharing (vss) scheme for medical images. In *2019 11th International Conference on Communication Systems Networks (COMSNETS)*, pages 823–828, Jan 2019. (Cited at page 122)

[151] D. Naveen and K. Praveen. Puf authentication using visual secret sharing scheme. In *2019 5th International Conference on Advanced Computing Communication Systems (ICACCS)*, pages 472–475, March 2019. (Cited at page 122)

[152] Antonio Celesti, Antonino Galletta, Maria Fazio, and Massimo Villari. Towards hybrid multi-cloud storage systems: Understanding how to perform data transfer. *Big Data Research*, 16:1–17, 2019. (Cited at page 122)

[153] Antonino Galletta, Antonio Celesti, Francesco Tusa, Maria Fazio, Placido Bramanti, and Massimo Villari. Big mri data dissemination and retrieval in a multi-cloud hospital

storage system. In *Proceedings of the 2017 International Conference on Digital Health*, pages 162–166. ACM, 2017. (Cited at page 122)

[154] Antonino Galletta, Lilla Bonanno, Antonio Celesti, Silvia Marino, Placido Bramanti, and Massimo Villari. An approach to share mri data over the cloud preserving patients' privacy. In *2017 IEEE Symposium on Computers and Communications (ISCC)*, pages 94–99. IEEE, 2017. (Cited at page 122)

[155] A. Kamble and S. Bhutad. Iot based patient health monitoring system with nested cloud security. In *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, pages 1–5, Dec 2018. (Cited at page 122)

[156] A. K. Chattopadhyay, D. Ghosh, P. Maitra, A. Nag, and H. N. Saha. A verifiable (n, n) secret image sharing scheme using xor operations. In *2018 9th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, pages 1025–1031, Nov 2018. (Cited at page 122)

[157] S. D. Patil and P. K. Ithape. Verifiable image secret sharing with cheater identification. In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pages 1238–1241, April 2018. (Cited at page 122)

[158] A. Galletta, J. Taheri, and M. Villari. On the applicability of secret share algorithms for saving data on iot, edge and cloud devices. In *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 14–21, July 2019. (Cited at page 122)

[159] M. Z. Ejaz, K. Khurshid, Z. Abbas, M. A. Aizaz, and A. Nawaz. A novel image encoding and communication technique of b/w images for iot, robotics and drones using (15, 11) reed solomon scheme. In *2018 Advances in Science and Engineering Technology International Conferences (ASET)*, pages 1–6, Feb 2018. (Cited at page 123)

[160] G. Devi Ramaraj, S. Venkatakrishnan, G. Balasubramanian, and S. Sridhar. Aerial surveillance of public areas with autonomous track and follow using image processing. In *2017 International Conference on Computer and Drone Applications (IConDA)*, pages 92–95, Nov 2017. (Cited at page 123)

[161] P. Agarwal and M. K. Singh. A multipurpose drone for water sampling video surveillance. In *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*, pages 1–5, Feb 2019. (Cited at page 123)

[162] S. Yong, A. L. W. Chung, W. K. Yeap, and P. Sallis. Motion detection using drone's vision. In *2017 Asia Modelling Symposium (AMS)*, pages 108–112, Dec 2017. (Cited at page 123)

[163] A. Chowdhery and M. Chiang. Model predictive compression for drone video analytics. In *2018 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops)*, pages 1–5, June 2018. (Cited at page 123)

[164] Power Consumption Benchmarks. `https://www.pidramble.com/wiki/benchmarks/power-consumption`. Last access: June 2019. (Cited at page 135)

[165] Real time traffic accident statistics. `http://www.icebike.org/real-time-traffic-accident-statistics`. Last accessed September 2019. (Cited at page 139)

[166] Nsc motor vehicle fatality estimates. statistics department national safety council (nsn). `http://www.nsc.org/NewsDocuments/2017/12-month-estimates.pdf`. Last accessed September 2019. (Cited at page 139)

[167] Number of passenger cars and commercial vehicles in use worldwide from 2006 to 2015 in (1,000 units). `https://www.statista.com/statistics/281134/number-of-vehicles-in-use-worldwide`. Last accessed September 2019. (Cited at page 139)

[168] The number of cars worldwide is set to double by 2040. `https://www.weforum.org/agenda/2016/04/the-number-of-cars-worldwide-is-set-to-double-by-2040`. Last accessed September 2019. (Cited at page 139)

[169] European commission - fact sheet, 2016 road safety statistics: What is behind the figures? `http://europa.eu/rapid/press-release_MEMO-17-675_en.htm`. Last accessed September 2019. (Cited at page 140)

[170] A. Lay-Ekuakille, P. Vergallo, D. Saracino, and A. Trotta. Optimizing and post processing of a smart beamformer for obstacle retrieval. *IEEE Sensors Journal*, 12(5):1294–1299, May 2012. (Cited at page 141)

[171] A. Lay-Ekuakille, A. Trotta, and G. Vendramin. Beamforming-based acoustic imaging for distance retrieval. In *2008 IEEE Instrumentation and Measurement Technology Conference*, pages 1466–1470, May 2008. (Cited at page 141)

[172] H. Al Najada and I. Mahgoub. Big vehicular traffic data mining: Towards accident and congestion prevention. In *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 256–261, Sept 2016. (Cited at page 141)

[173] G. Galati, M. Gasbarra, and E. G. Piracci. Decoding techniques for ssr mode s signals in high traffic environment. In *European Radar Conference, 2005. EURAD 2005.*, pages 383–386, Oct 2005. (Cited at page 141)

[174] P. Vergallo, A. Lay-Ekuakille, and D. Caratelli. Sparsity of the field signal-based method for improving spatial resolution in antenna sensor array processing. *Progress in Electromagnetics Research*, 142:369–388, 2013. (Cited at page 141)

[175] A. Lay-Ekuakille, N.I. Giannoccaro, S. Casciaro, F. Conversano, and R. Velazquez. Modeling and designing a full beamformer for acoustic sensing and measurement. *International Journal on Smart Sensing and Intelligent Systems*, 10(3):718–734, 2017. (Cited at page 141)

[176] I. Ahmad, R.M. Noor, I. Ali, and M.A. Qureshi. The role of vehicular cloud computing in road traffic management: A survey. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 185:123–131, 2017. (Cited at page 141)

[177] I. Ahmad, R.M. Noor, I. Ali, M. Imran, and A. Vasilakos. Characterizing the role of vehicular cloud computing in road traffic management. *International Journal of Distributed Sensor Networks*, 13(5), April 2017. (Cited at page 141)

[178] Y.U. Devi and M.S.S. Rukmini. Iot in connected vehicles: Challenges and issues - a review. In *International Conference on Signal Processing, Communication, Power and Embedded System, SCOPES 2016 - Proceedings*, pages 1864–1867, 2017. (Cited at page 141)

[179] J. A. Guerrero-ibanez, S. Zeadally, and J. Contreras-Castillo. Integration challenges of intelligent transportation systems with connected vehicle, cloud computing, and internet of things technologies. *IEEE Wireless Communications*, 22(6):122–128, December 2015. (Cited at page 141)

[180] F. Lin, Q.H. Huang, and Q. Tan. Distributed storage video cloud of intelligent transportation system based on the backward cloud generator. *Advances in Transportation Studies*, 2:15–22, 2015. (Cited at page 141)

[181] Yen-Wen Lin, Jie-Min Shen, and Hao-Chun Weng. Cloud-supported seamless internet access in intelligent transportation systems. *Wireless Personal Communications*, 72(4):2081–2106, October 2013. (Cited at page 142)

[182] B.-J. Chang, Y.-L. Tsai, and Y.-H. Liang. Platoon-based cooperative adaptive cruise control for achieving active safe driving through mobile vehicular cloud computing. *Wireless Personal Communications*, pages 1–27, August 2017. (Cited at page 142)

[183] A.B. Abhale and S.A. Khandelwal. Intelligence transportation service using vehicular cloud network. In *2016 IEEE International Conference on Advances in Electronics, Communication and Computer Technology, ICAECCT 2016*, pages 277–282, 2017. (Cited at page 142)

[184] N. Fida, F. Khan, M.A. Jan, and Z. Khan. Performance analysis of vehicular adhoc network using different highway traffic scenarios in cloud computing. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 185:157–166, 2017. (Cited at page 142)

[185] H. Al Najada and I. Mahgoub. Big vehicular traffic data mining: Towards accident and congestion prevention. In *2016 International Wireless Communications and Mobile Computing Conference, IWCMC 2016*, pages 256–261, 2016. (Cited at page 142)

[186] Giuseppe Di Modica and Orazio Tomarchio. Matching the business perspectives of providers and customers in future cloud markets. *Cluster Computing*, 18(1):457–475, 2015. (Cited at page 146)

[187] Giuseppe Di Modica, Orazio Tomarchio, and Lorenzo Vita. Resource and Service Discovery in SOA: a P2P Oriented Semantic Approach. *International Journal of Applied Mathematics and Computer Science*, 21(2):285–294, 2011. (Cited at page 146)

[188] G. D. Modica and O. Tomarchio. Semantic security policy matching in service oriented architectures. In *2011 IEEE World Congress on Services*, pages 399–405, July 2011. (Cited at page 146)

[189] Advanced Analytics Market by Big Data Analytics, Social Analytics, Visual Analytics, Customer Analytics, Risk Analytics, Business Analytics, Statistical Analysis,

Predictive Modeling - Global Advancement, Worldwide Market Forecast & Analysis 2014 - 2019. https://www.marketsandmarkets.com/Market-Reports/advanced-analytics-market-58104148.html. Last access: November 2019.

[190] Giacomo Domini, Marco Grazzi, Daniele Moschella, and Tania Treibich. Threats and opportunities in the digital era: automation spikes and employment dynamics. LEM Papers Series 2019/22, Laboratory of Economics and Management (LEM), Sant'Anna School of Advanced Studies, Pisa, Italy, July 2019.

[191] M. Fazio, A. Celesti, A. Puliafito, and M.s Villari. A message oriented middleware for cloud computing to improve efficiency in risk management systems. *Scalable Computing: Practice and Experience*, 14(4):201–213, 2013.

[192] AES. https://csrc.nist.gov/projects/cryptographic-standards-and%2Dguidelines/archived-crypto-projects/aes-development. Last access: November 2019.

[193] Twofish web page, with full specifications, free source code, and other Twofish resources. https://www.schneier.com/academic/twofish/. Last access: November 2019.

[194] Triple DES Encryption. https://www.ibm.com/support/knowledgecenter/SSLTBW_2.1.0/com.ibm.zos.v2r1.csfb400/gloss.htm. Last access: November 2019.

[195] The Original RSA Patent as filed with the U.S. Patent Office by Rivest; Ronald L. (Belmont, MA), Shamir; Adi (Cambridge, MA), Adleman; Leonard M. (Arlington, MA), December 14, 1977. https://patents.google.com/patent/US4405829. Last access: November 2019.

[196] RFC 1321, section 3.4, "Step 4. Process Message in 16-Word Blocks", page 5. https://tools.ietf.org/html/rfc1321. Last access: November 2019.

[197] https://tools.ietf.org/html/rfc4634. https://tools.ietf.org/html/rfc4634. Last access: November 2019.

[198] HAVAL (the official HAVAL page with the research paper on HAVAL, the latest C source code and HAVAL OIDs). https://web.archive.org/web/20150111210116/http://labs.calyptix.com/haval.php. Last access: November 2019.

[199] B. Sotomayor, R.S. Montero, I.M. Llorente, and I Foster. Virtual Infrastructure Management in Private and Hybrid Clouds. *Internet Computing, IEEE*, 13(5):14–22, Sept.-Oct. 2009.

[200] Native Database XML System. `http://www.sedna.org`. Last access: March 2019.

[201] Ejabberd, the Erlang Jabber/XMPP daemon. `http://www.ejabberd.im/`. Last access: March 2019.

[202] jClouds, “The Java Multi-Cloud Toolkit”. `https://jclouds.apache.org/`. Last access: March 2019.