The University of Sheffield
International Faculty

Computer Science Department

IoT and Big data

This report is submitted in partial fulfillment for the degree of Master of Science in Advanced Software Engineering, by

Ognen Bekjarovski

December, 2018

Supervisor

Dr. George Eleftherakis

IoT and Big data

by Ognen Bekjarovski

Supervisor: Dr. George Eleftherakis

Abstract

This project presents the work performed on the design and implementation of a service that is providing storing and processing big amounts of data produced by an IoT devices. In order to fulfill this, the main challenges of the IoT paradigm have to be overcomed. Therefore this report will present the security issues and the challenges of processing large datasets, along with an detailed software architecture through which they are tackled. Afterwards, the proposed architecture would be tested and evaluated. Finally, the results would be shown, illustrating the success of overcoming the IoT challenges.

DECLARATION

All sentences or passages quoted in this dissertation from other people's work have been specifically acknowledgement by clear cross-referencing to author, work and page(s). I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this dissertation and the degree examination as a whole.

Name: Ognen Bekjarovski
Signed: Date: December, 2018

Acknowledgements

The following paragraphs are dedicated to all of those who helped me and supported me realising this project and throughout my academic studies.

First, i would like to thank my family for giving me the opportunity and support in fulfilling my dreams. Second, special gratitude to my friends helping me i both good and bad times.

Additionally, I would like to give my appreciation to the entire staff of CITY college, helping me develop my knowledge further, during this short and fulfilled year. FInally, i would like to give my deepest appreciation to Dr. George Eleftherakis who was always here for me, giving me an immense knowledge and guidance in developing this project.

Contents

List of Figures

Listings

Chapter 1

Introduction

Internet of Things (IoT) is a socio-technical phenomena that is rapidly increasing popularity in the society. In the broadest sense it encompasses different kind of physical devices connected on the internet, embedded with sensors, actuators, software and connectivity enabling them to exchange data [1]. The term Internet of things actually means a world-wide network of numerous interconnected heterogeneous devices, each with different identifier and purpose. Together with the increase of popularity the number of connected devices also increased, with expectation of almost 100 billion devices on the network by 2025 [2].

By combining these devices in automated systems, gathering information from them and analyzing that data is becoming possible, which contributes to reaching goals and objectives in different fields that were infeasible in the past. Therefore IoT has found many implications in different areas. One application can be found in transportation and logistics domain. Starting from devices which help drivers to get to desired destination, to totally autonomous vehicles. Next, smart environments are belonging in this category, including numerous connected light switches, home appliances and even locks. Furthermore, many benefits are provided to the healthcare domain, where the devices can monitor and track the patients health state, but also collecting and analyzing patients health state to predict and discover serious diseases. There are many more implications and expectations of the IoT possibilities, with one achievable idea of creating a smart cities, which will change the future of our everyday life completely.

Before fulfilling its full potential, couple of challenges need to be addressed [3]. One of the biggest challenges is the interoperability problem. As mentioned before the network consist of many different kind of devices, from different manufacturers, types, etc. Therefore one general protocol needs to be found in connecting this heterogeneous network. Another important challenge is the security of those devices. There are many examples of exploiting these powerful devices. launching the biggest DoS attack so far [4]. Many of these exploits come from the low understandings of the power and the influence that these devices have in our everyday life. Furthermore, there exist many issues related to the resilience and the reliability of the devices, and development of efficient software which can run on those devices

with low capabilities.

The data that IoT devices produce is big in volume, multi dimensional and updates frequently. For Instance one GPS application used in a vehicle commits new record of 100 kilobytes each 60 seconds, therefore 1 million users will generate data of 144 Gigabytes in only one day [5]. These vast amounts of information are characterized as a Big data. In general, this term specifies datasets which could not be stored, analyzed and managed using traditional software and hardware tools within a tolerable time [6]. Big data is characterized by: volume, variety, velocity and value. Volume refers to the amount of data generated from all sources , variety is about the different types of data collected from different sources (sensors, smartphones, social network...), the speed of data transfer and generation refers to the velocity while value refers to the discoverings from the big datasets. The data can have also different formats, structured unstructured and semi-structured, where each type have to be processed in different manner.

Although the data can come from a different sources, IoT will have the most important part of big data [7], as predictions are almost a trillion devices by the end of the next decade. This fact leads to a recognition that these two technologies, Big data and IoT, are highly inter-dependent and should be jointly developed. The growth of devices drives high growth of data providing an opportunity for application and development of big data. On the other hand, improvement in big data technology accelerates research advances of IoT. As mentioned before, the enormous volumes of data can not be processed and stored on a traditional way. Therefore many distributed system for processing and storing big data have emerged. Within these systems processing is performed in parallel on many nodes, machines, distributed on different physical locations. Many providers exist which offer services in processing and storing data on their infrastructure. These services are needed as it is highly expensive and demanding to create and maintain the infrastructure additionally providing efficient management. These advantages led to a many applications that leverage various big data processing services to have a big success.

1.1 Aims and Objectives

With the increase of the number of IoT devices and the capabilities of processing the big amounts of data generated, we have realised the potential this paradigm can reach and the benefits we can receive. However the data gathering demands highly sophisticated tools and models, in order that data to be processed and stored later. There is a high demand of services which will fulfill those tasks. The reason is that creating such a sophisticated tools is usually expensive and infeasible as it requires expertise along with lot of computing resources. Therefore the **aim** of this project would be defined as following:

Design and develop service which will enable efficient, flexible and secure transmission, storing and processing of big amounts of data generated from various

kinds of IoT devices.

In realizing such a service couple of **objectives** needs to be fulfilled.

O1 Creating a secure and lightweight two way data transmission service interface, for communication with the IoT devices.

O2 Creating the main functionalities of the service: processing, storing and interface for data retrieving.

O3 Implementing the service to act as a broker for different big data processing tools, meaning that the clients can define the tool they want (Apache Hadoop, Spark, Hive etc.) for processing their data.

O4 Creating a functionality on the broker, which will enable the IoT devices to define the model describing the manner in which the data will be processed.

O5 Defining a structure for the data accompanied by a metadata. The metadata will contain the information for the IoT data manipulation actions: storing, processing and retrieving, big data processing tools and the data processing model.

O6 Implementing data transmission protection mechanism.

O7 Implementing security level allowing only to authorised IoT devices to access the service.

1.2 Report Structure

The report is structured in nine sections. The first one is the introduction, in which this chapter is located. The purpose of this chapter is giving an brief overview of the project and defining the main aims and objectives it would fulfill. Chapter 2, Literature Review, presents the main review conducted in the academic literature, which gives a better understanding and further guidance in order the main goal of the system to be achieved. Furthermore, this chapter defines and further explains the issues and challenges of the IoT network, its relation with Big data, the security issues of the network and a proper lightweight communication protocol. Chapter 3, Problem Solving Process, explains the end vision of the project, along with the software management, defining the proper software process model. In addition, the risk of the project is defined and mitigation strategy is further explained in this chapter. In the next Chapter 4, Analysis, a detailed analysis of the functional and non-functional requirements of the project are presented. These requirements are the guidance for the development of the further phases. By fulfilling them, the main objectives and aim of the project are completed.

Following is the Chapter 5, Design, which is responsible for translating the requirements in a well defined system. The main goal is defining the architecture of the system, following the findings defined in the literature review. Finally in this chapter the achievement of the main project objectives is presented. After defining the overall design, the Chapter 6, Implementation, represents the implementation process of the architecture previously defined. Therefore, it can be seen that description of the requirements and definition architecture have a direct implications at this stage. This highlights the importance of each stage of the development. In order the implementation to be validated, Chapter 7, Testing, provides different mechanisms of checking the success of the previous phase. Explained are three types of testings, starting with the unit testing, then performance and ending with security testings.

In the end, after the system is implemented and tested, it is time to validate whether the main aims and objectives defined at the beginning are met. Chapter 8, Evaluation, discusses the employed system evaluating whether the main aim of the project was successfully achieved. In the end, Chapter 9, Conclusion and Future Work, provides a final conclusion of the developed system and defines further possibilities for an extension of the project.

Chapter 2

Literature Review

2.1 Internet of things

2.1.1 Definition of IoT

In the last decade the internet of things, IoT, has been gaining popularity in the society giving benefit in many different domains. The definition of the term is straightforward and agreed upon all the experts in the field, a network of interconnected devices embedded with electronics, software, sensors and connectivity enabling the devices to communicate collect and exchange data [8]. The devices involved are beyond the standard machines, such as desktop computers, laptops, smartphones, etc. to a less powerful and capable physical devices and everyday objects. These items, embedded with technology can communicate over the network, collect valuable information from the environment, and can be remotely controlled.

The main strength of the IoT idea is the impact it will have on our everyday life, improving our quality of life. The possibility of the things to collect data and communicate it provides many benefits and gives opportunities for big advancements in many different fields. These devices have found application in both the commercial and industrial areas. Many devices can already be noticed in households, at workspaces and in the cities. However, there is a big potential of finding many more applications of the IoT. The main domains in which the things are integrated can be divided in the following groups [1]:

- Transportation and logistics
- Healthcare
- Smart environment (smart home, city)
- Personal and social domain

Transportation and logistics Many advanced cars, trains, bicycles as well as roads and rails are becoming more instrumented with sensors and processing power. This gives the opportunity of assisting and making the driving more secure. Further on, these devices give the possibility of creating an autonomous vehicles, which can

communicate between each other in eliminating any kind of accidents. Additionally mobile ticketing and purchasing have become reality with the emerge of the IoT. A great benefit in the logistics provides the possibility of monitoring the environment, for example, many fresh fruits, meats and dairy products are transported daily, so the conservation status(temperature, humidity) can be monitored and controlled, improving the food quality.

Healthcare Many benefits are provided by the IoT in the healthcare domain. Applications can be found in different areas of the health domain, starting from tracking of the workflow in the hospitals and thus improving it, tracking the equipment and materials for maintenance, to a scenarios where tracking is employed to prevent a surgery left-ins. One of the crucial benefits of the things is the monitoring the state of the body, collecting information about its state,predicting a future diseases and preventing them from occurring.Finally, one such example is the monitoring for possible heart attacks.

Smart environment These environments employ the constrained devices in making the office, home or industrial plant more comfortable and easier to control. In the homes there are many appliances that can be controlled remotely, thus making the everyday life easier. Therefore, the room temperature can be adapted to our preferences, energy can be saved by automatically switching the lights off, monitoring and alarm system can be integrated, and many more applications can be found. The industries can benefit also from these devices, as the automation of the industrial plants can be improved. The IoT is integrated in the cities in creating the paradigm known as smart cities. In them, the things are increasing the operational efficiency, information can be shared to the public and improving the public services and citizens welfare. For this purpose monitoring is employed on traffic, air pollution, areas with high crime, etc. using the constrained devices.

Personal and social domain The applications belonging to this domain are the ones that enable the user to interact with other people and therefore building and maintaining a social relationships. Many social networking applications have been utilised , providing a ways of better organization of social gatherings and events. Additionally, the sensors on the devices provide an easier way of finding our lost things or stolen things. One of the greatest benefits is the ease of which people with memory and orientation problems can be found if one such a device is kept by them. However, this last domain can bring many ethical problems, if not implemented properly.

2.1.2 Challenges in IoT

As it can be concluded the IoT brings many benefits and improvements. In the future it will become even more present and deeper integrated in the environment. However, before the IoT fulfills its potential, many challenges are standing on the way. According to the academia [3] the main difficulties to be overcomed are listed below:

Technological interoperability: This challenge occurs as there are many devices from various manufacturers, implemented with different architectures. However, the community tries to overcome it by defining common communication protocols and standards for interaction with the devices. Additionally, these devices can differ regarding their technological capabilities.

Devices capabilities: Devices with low power consumption and with capabilities of harsh environment have to be developed. Moreover, these limited devices have to enable parallel processing in multi-processor systems, have to easily adapt to the changes of the environment, providing autonomous behavior while guaranteeing security and privacy. As they can be placed in inaccessible locations, properties like battery saving and energy harvesting and storage technologies are also among the core challenges in the IoT.

Resilience and reliability: As applications of IoT can be found in the industrial environments or emergencies cases, where outrages cannot be accepted. Therefore, resilience and reliability have to be appropriately approached and investigated from an overall system view. In resolving these challenges, aspects like robustness, availability, flexibility in hardware and communication to changing conditions in the environment have to be considered, additionally, single point of failure have to be avoided.

Security: Given that the IoT work with vast amount of data which in many cases can be private and sensitive, preserving information security on the data is the most important challenge that cannot be neglected [9]. According the impact factors, the diversity of the things and their communication, security challenges are divided in two categories. First, a security issue arises from careless program design, making the things vulnerable to malwares and backdoors installation. Based on the heterogeneity of the IoT devices, such a security problems are becoming more complex in comparison with the security problems the community is facing so far. As for the communication, the hetegenoricity again becomes a big issue, as different communication protocols and medias exist, facing different challenges.

However, many issues need to be overcomed when creating a secure IoT network. The way of authorising and authenticating the devices is an important area. Many traditional access control mechanism have been applied with no success, as the network can grow rapidly in a scales not experienced yet. Different solutions have been proposed to resolve the problem of authorisation and authentication [10] [11], however, there are still no common agreements on standards in the specific area. Another large issue, which is gaining popularity nowadays, is the privacy of the data collected, as it can hold sensitive information, giving an information about users daily routines. Owners of this data can exploit it if mechanism preserving the privacy of the collected information are not properly addressed. In the end, the transmission have to be properly protected, although many cryptosystems exist, they may not be suitable for the resource constrained devices. Therefore, lightweight cryptosystems and protocols have to be utilised.

However, the idea of IoT became possible with the occurrence of distributed system. The data collected by the devices is on a scale which traditional storage and processing system are unable to comprehend. Therefore, distributed systems connected on the internet provide the basis for further realisation of the IoT. These systems and the specific data they hold would be described in more details in the following chapter.

2.2 Big data

2.2.1 Big Data Defnitoin

Same as the IoT, In the last decade Big data is becoming increasingly popular and researched field in the IT community, attracting the interest of many professionals. According the statistics [12], in 2017 the overall created and copied data in the world was 25 zetabytes (size of ZB is 10^{21} bytes), which have increased 15 times since 2011. The challenge of the big data is finding an optimal way of processing and storing the enormous datasets. This enormous data brings new opportunities for discovering new values in different fields, gaining an in-depth understanding of the hidden values, and also presents a new challenges of manipulating and organising such datasets.

Although the importance and benefits of big data have been recognised, experts in this field still have not agreed in its strict definition [6]. In its beginning, Apache hadoop defined the concept of big data as "datasets which could not be captured, managed, and processed by general computers within an acceptable scope". Meaning that the traditional ways of managing the data are not suitable and can not comprehend the new amounts of information that are generated. According this, it can be seen that the volume of the data is not the only criteria in defining big data. The increasing growth and its management difficulties that could not be handled by the traditional database management systems are also a features describing big data.

According to Gartner, IBM and the research department of Microsoft, big data can be defined by the 3 Vs model, describing the high-volume, high-velocity and high-variety of the data. Although this is a highly accepted description, the International data corporation, one of the most influential leaders in big data, have added one more V in the model. This last feature refers to the value, measurement of the usefulness of the data. The modified 4V [13] model is one of the most accepted description regarding defining big data, therefore it would be elaborated in more details in the following paragraphs:

Volume The main characteristics that makes data big is its sheer volume. As already described the data have increased enormously in the past years and that trend is going to continue with predictions of doubling its size every year. Therefore, new techniques in its storing and processing have been utilised. The tools responsible

for managing big data are extended over many nodes inside a cluster of machines, sharing their resources in order to create an environment for manipulating those enormous datasets. Additionally these systems enable flexibility in means of introducing new nodes in the network essential in a situations when the scale of the data is overlapping the memory capabilities of the system.

Velocity It is a second measurement unit associated with big data, and it refers to the frequency of incoming data and finding an efficient way of ingest, process the data and later retrieve it in a quick manner. One such example is Facebook, couple of years ago there were measured around 900 million photo uploads per day. This represents a big issue, as the growth of the data and its speed of generation is becoming even more difficult and infeasible to manage and process it in a reasonable amount of time. Therefore, big data tools and services are providing solutions for these issues, highlighting their importance in the new data era that comes.

Variety variety is one of the most interesting features of the big data as more and more information is being digitized. Traditional data types include more structured data, defined by variables like amount, date, time, names etc., however unstructured data, which is a fundamental concept in big data, can take different forms and structure. Things like twitter feeds, audio, video, web logs, anything that can be captured but does not have a meta model is defined by unstructured data. Big data technologies are constructed in manner of supporting this vast diverse data types, providing necessary methods in storing, processing and analysis of the diverse data.

Value In the end the data gathered and processed have to bring a value and usefulness in making a specific decision. It has to be noted that the purpose of the computing is extracting a real value out of the data, instead of just generating numbers. This feature of the big data depends mainly on the approach taken in analysing and gaining knowledge out of the data. This responsibility have to be taken by the data analysts, and their expertise in using the big data tools available.

2.2.2 Big Data Generating Sources

The first step to big data is data generation. There are a variety of sources, which are producing big data. Therefore, some of the data generated can be closely related to people's behaviors, giving a valuable information about their habits and everyday life. In other case data can be collected in the industry, in order to be learnt how a specific task is performed and the ways it can be improved. Many reasons for collecting data exist and some of the most commonly sources are listed below.

Enterprise data Internal enterprise data is one of the main big data sources. This specific data mainly consist of online trading data and online trading analysis, additionally this enterprise group is composed of production and inventory data, sales, financial data, etc. With the occurrence of the global e-commerce sites like Amazon, Alibaba and eBay, this data is drastically increasing. As the business

turnover of Alibaba is estimated to be 9 millions dollars in 2018 [14], the importance of managing and real time analysis of the data is becoming crucial for the company.

IoT data IoT is among the most important data sources. As discussed, the smart things are find applicable in different domains, and with their number constantly increasing, the amounts of data that they produce will become even more significant in the near future. The different sensors existing on these things, are making them able to consistently collect data, therefore data generated by IoT is characterised by large scale.

Bio-medical data Series of high-throughput bio-measurement technologies have emerged in the beginning of the 21st century, meaning that the bio-medicine field also entered the field of big data. Many smart and efficient analytical models for bio-medicine applications are constructed, with a purpose of discovering complex biological phenomena. Many devices measuring the state of the organisms have been discovered, producing vast amounts of data, which forms the basis of the new biological discoveries. Additionally, the completion of Human Genome Project HGP also lead to a widespread applications of big data. One sequence of human gene may generate a data of 1.5Gigabytes, while many gene banks are already containing 30 millions of human gene samples, with these numbers expected to increase for the needs of further research and analysis.

Social media and media content Variety of social medias are used in our everyday lives, such as Twitter, Facebook, Instagram etc. The content they produce appears in different variations as text, images, video. Additionally, similar data is produced by media sites like Youtube. The amount of data the social media and content websites produce is enormous, and their management is becoming a big challenge for the industries.

All of these examples show the importance of successful big data storing, processing and analysis tools. Important information about the human behavior, health, entertainment, etc. can be gained if the data is managed on a efficient way and the analysis is approached appropriately. Furthermore, as the data will grow, this field would become even more crucial and influental in the future.

2.3 Security

As the number of connected entities and services provided on the world wide web increases, information security is becoming increasingly significant and important on the internet. Scenarios of breaching the security of organisations and stealing sensitive information are becoming more frequent, with the last years security statistics showing over 130 large-scale breaches in 2017 only in United states, with that number growing with a rate of 27 percent per year [15]. Another serious attack caused 3 billion yahoo accounts to be stolen. Moreover, in the last few years ransomware malware caused 400,000 to be infected, all of these cases, including many

more which were not mentioned, caused the top of the FBI's most wanted list to be filled with cybercriminals [16]. Therefore, as one of the security experts, Bruce Schneier, says [17], the need of security in any service is crucial task, and it should be part of any developed product.

However, security can mean many things in different environments, hence, it has to be properly defined, including the main goals and mechanisms leading to the desired secure environment. Security engineering, is a field of engineering which focuses on defining the security and the aforementioned mechanism in achieving it. One of the security gurus and pioneer in this field, Ross Anderson, in his book "security engineering" [18], he gives the finest definition of the term:

> *"Security engineering is about building systems to remain dependable in the face of malice, error, or mischance. As a discipline, it focuses on the tools, processes, and methods needed to design, implement, and test complete systems, and to adapt existing systems as their environment evolves".*

Security engineering requires cross-disciplinary expertise, ranging from computer security, hardware tamper resistance, cryptography to knowledge of psychology, law, economics etc. additionally it includes many subfields like physical security, technical surveillance and countermeasures, economics of security and the most important for the IT community is the information security field. The purpose of information security is preventing an unauthorised access, use, modification, recording, disclosure or destruction of information.

According many experts in the field, the primary information security goals is the balanced protection of the confidentiality, integrity and availability, the so called CIA triad [19]. However, according to Anderson, non-repudiation is also be part of the main security goals, which many experts agree with. Clear definition of the goals is the first step of creating a security environment, therefore their description is given in the following bullets.

- **Confidentiality:** it is a property that information is not made available or disclosed to an unauthorised entities, individuals or processes. Confidentiality employs the mechanism of secrecy, limiting the number of principals able to access of a specific information.
- **Integrity:** data integrity means ensuring the accuracy and completeness of the data during its entire lifecycle. Thus, mechanisms needs to be provided in ensuring that the data have not been altered or modified by an unauthorised entities during its transmission or storing. These mechanism should notify the modifications not only in a scenarios of malicious intents, but also in situations of error or mischance.
- **Availability:** in order the system to serve its purpose it has to be available when it is needed. For the users, working with the system this is one of the most important aspects, as the system should work as it is expected, remaining

available even in a cases of power outages, hardware failure, system updates. Availability also means prevention of denial of service attacks, ensuring that the system will support organisation operations in case of message flooding which naturally would force the system to shut down.

- **Non-repudiation:** it implies that an entity can not deny the actions it have taken, meaning that if some information or statement is sent or stated by an entity, it can not deny it. Many issues can arise in providing this property, as during the transmission or storing of the message it can be altered, and therefore the statements in it can not be claimed to originate from the assumed entity. Therefore, when considering non-repudiation one have to be extremely careful the ways it is implemented and adopted.

As the systems are growing and many new types of cyber attacks are introduced every day, this core list of security goals is further expanded by additional security requirements [20]. One of these additional goals is the auditability, an ability of the system to conduct monitoring on all of the actions. Another one is anonymity, ensuring that the identity of an entity of a specific action would remain secret. Trustworthiness is also an important security goal, by the definition of NSA it is a property of a third party system which would not fail, and thus break the security policies. Privacy is also important for many systems, enabling individuals to control their own personal information. It is similar to confidentiality, but there is one fundamental difference between these two terms. Confidentiality is protecting other persons or organisations information while privacy is protective of one's personal secrets. Other important security concept mentioned by Anderson is authenticity, ensuring that a message, transaction or any other kind of information exchange is from a source it claims to be. This concept is based on a proof of identity, where a proof is something the individual knows, like a password or other credential, or the individual can provide proof with something it posses, keycard, biometric item etc. AAs the system are growing and many new types of cyber attacks are introduced every day, this core list of security goals is further expanded by additional security requirements [20]. One of these additional goals is the auditability, an ability of the system to conduct monitoring on all of the actions. Another one is anonymity, ensuring that the identity of an entity of a specific action would remain secret. Trustworthiness is also an important security goal, by the definition of NSA it is a property of a third party system which would not fail, and thus break the security policies. Privacy is also important for many systems, enabling individuals to control their own personal information. It is similar to confidentiality, but there is one fundamental difference between these two terms. Confidentiality is protecting other persons or organisations information while privacy is protective of one's personal secrets. Other important security concept mentioned by Anderson is authenticity, ensuring that a message, transaction or any other kind of information exchange is from a source it claims to be. This concept is based on a proof of identity, where a proof is something the individual knows, like a password or other credential, or the individual can provide proof with something it posses, keycard, biometric item etc. Additionally, the ease of use of the system is also an important goal, as if the security properties does not let the users work efficiently and as expected, they would

omit the security mechanisms in every occasion it is possible to do so.dditionally, the ease of use of the system is also an important goal, as if the security properties does not let the users work efficiently and as expected, they would omit the security mechanisms in every occasion it is possible to do so.

When creating a secure system, some of these protection goals would be more in the focus and approached with more attention than the others. This reasoning comes from the fact that different system are securing different resources and the attacks can be provided in different manner. Therefore, before starting to implement any security property in the system, it is important to be known what have to be protected and who is going to attack the developed system. For example, in a law institution, the most important goal is to prove the non-repudiation of the actions done by the individuals. If a person is accused of robbery or murder, its name and actions have to remembered and not tampered or changed. In a situation of a security attack, the attacker would like to change the mentioned information. Therefore the system would have to provide strong non-repudiation mechanisms. On the other hand, confidentiality is not a crucial task, as the identification of the individual on the trial is important to be known. On the other hand, in military confidentiality is the main goal that a system have to achieve, as discovering an information by an unauthorized individual can cost the lives of many. In addition, creating an profile of possible attacker is essential when developing a secure system.This profile would have to provide information of what type of individual would try to attack the system, with the kind of knowledge and resources it possesses and his. If we go back to the military attack, hypothetically the most possible attacker would be someone inside the military network, therefore the system would have to provide strong mechanisms protecting any data leakages outside the network. In conclusion, the most important aspects before creating a secure system is defining the resources that to be protected, defining the main protection goals which have to be achieved and creating an assumptions of the possible attackers and the actions they would take. In addition, it is always worth knowing the value of the resources that are protected, as their protection should not be more costly than the resources themselves.

After developing the secure system, the next goal is keeping it protected during its work in the environment it was built for. For this reason, rules and constraints on behavior and access of the users to the system have to be defined, and also constraints on the functions in the system and the flow among them has to be defined. These constraints and rules are defined as a security policies. The system might have best security tools and mechanism implemented, however its protection would fail if the policies are not properly defined, followed and regularly checked if they are followed. One simple example is the weak passwords used frequently in an organisation. The system could have the best encryption mechanisms, however if the passwords can be easily guessed and retrieved, the whole infrastructure would fail from one simple weak credential usage. Applying and following, these policies are the most crucial in the security infrastructure, because as Anderson mentions people are the weakest link when it comes to security, stating that most of the breaches come from a human mistake.

However, when constructing a system many tools and mechanisms exist which can achieve the main security goals, but the choice of which mechanism would be used is defined by the nature of the project, and the things we try to protect. The tools and mechanism utilised in the project will be defined in the following sections.

2.3.1 Cryptography

According to one of the most important figures in the field of cryptography, Ronald Rivest, the term cryptography, defines the practice and study of techniques for secure communication in the presence of third parties [21]. In general, cryptography is about constructing and analysing protocols that prevent third parties from reading a private message. The goals of the information security, the CIA triad and all the other concepts defined in the previous chapter, are central to cryptography. Additionally, the reason the cryptography is so powerful and relied on, is that all of the protection algorithms have mathematical basis behind them, leaving nothing to the coincidence. Importance of cryptography is also highlighted by Schneier, where in one of his essays [22], he describes cryptographic encryption as the most critical component of security, describing it as a tool for protecting every critical infrastructure in the society, our communications network, the data we store everyday, our transportation infrastructure etc. However he suggests that security is not only cryptography, it involves many other aspects which have to be considered. However, in utilizing the possibilities cryptography has to provide, first the different cryptographic algorithms and mechanisms which already exist have to be understood and defined.

When we talk about cryptography, the first thing that comes on our mind is encryption, which uses algorithms and a special key in transforming the input, plaintext, into an encrypted output, ciphertext, which later on is decrypted with the same or another key, and the plaintext is transformed back. Figure 2.1. Illustrates this process, where a plaintext is encrypted in a specific ciphertext, which if it is acquired in this form, without the knowledge of the secret key and mechanism of decryption, it is impossible to get the meaning of the message. These processes of transformation of the messages, are known way in history long time before the computer era. Those mechanisms are not any more used and supported, as their breaking can be conducted in a matter of seconds using a personal computer. However, they form the basis of the modern encryption algorithms and provide a guidance of the goals they have to accomplish. The inception of the modern era of cryptography is closely related to the beginning of the computer era. The beginning of the digital computers was initiated in WWII with a purpose of cracking the algorithms (ciphers) of encryption and decryption, generated by the german army's machines. The modern era of cryptography is divided into two types of encryption methods, symmetric-key cryptography and Public-key cryptography.

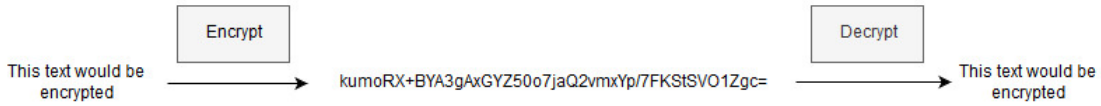


Figure 2.1: Illustration of encryption and decryption of a message

Symmetric-key cryptography use the same key for encryption and decryption of a message, therefore the sender and the receiver share the same secret key. These type of ciphers were introduced before the public-key cryptography emerged. Symmetric encryption provides a confidentiality of the messages, as no one, except the individuals having the secret key can get the meaning of the message. Additionally, it provides authenticity, as the both sides of the communication know each other, because the secret key have to be agreed between them before the communication begins. Integrity is not achieved using this type of ciphers, however some modification of the algorithms provide this property. Although, it satisfies some of the information security requirements, symmetric key cryptography have one big drawback in implementing it in the modern internet environment. As the communication on the Internet can be performed by any two entities, separated spatially from each other, the agreement and sharing of a secret key is impossible to be done in a reasonable amount of time. Also securely sharing a secret key over an insecure channel, presents the chicken-and-egg problem, as if we find a way to securely transfer the secret key, with that any particular data, where is the need of establishing such a key at all.

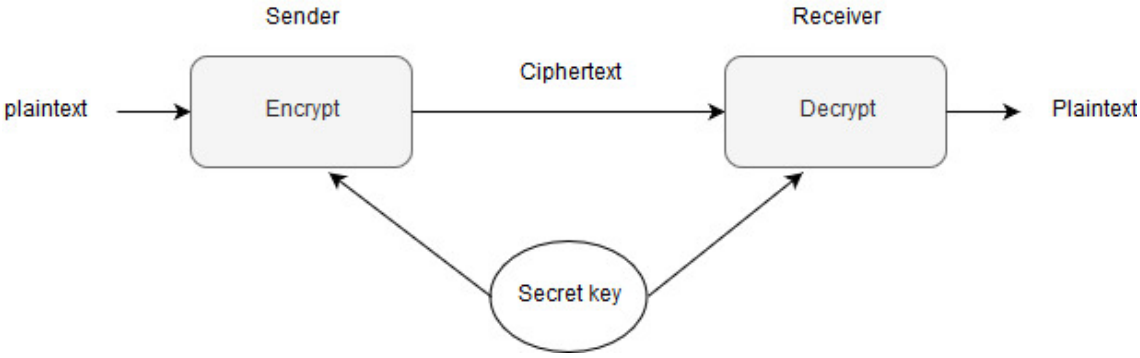


Figure 2.2: Symmetric key cryptography

Therefore, in 1976, the first asymmetric, public-key cryptography was introduced by Diffie and Hellman [23]. In contrast of the symmetric ciphers, the encryption and decryption are performed using different keys, one of them is public while the other one is private. The public key is known to everyone and it can be used in encrypting the message and sent to the holder of the private key, which is the only one capable of decrypting the message. The way that this key pair is generated, provides a mathematical proof that from the public key it is computationally infeasible

the private key to be constructed. Public-key cryptography made it possible the communication on the World Wide Web to be secure, as anyone knowing the public key of an entity that wants to send some data to, would be able to perform it in secure way, without any previous key agreements. It is obvious that this technique provides confidentiality of the system, as only the one holding the private key is able to understand the meaning of the message. However, the central problem with the use of public key cryptography is proof that a particular public key is authentic, meaning that it belongs to an entity claiming the possession of the key. The way that the community solved this problem is by introducing a public key infrastructure PKI, consisting of one or many third parties known as certificate authorities, which certify the ownership of the key pairs. The PKI additionally consist of revocation list, which stores the public keys which have been breached and a central directory, which securely stores the public keys. So, whenever a message has to be send on a secure way, the sender acquires the public key, of the receiver, from the certificate authority. Usage of PKI introduces authenticity in the system, as it can be proven who is standing on the other side of the communication. Another important aspect of the PKI is the three different types of trust models: monopoly, in which only one certificate authority exist, oligarchy, which is used on the global Internet, where couple of certificate authorities exist in certifying the public keys of the entities, and in the end the anarchy model, in which everyone trust each other.

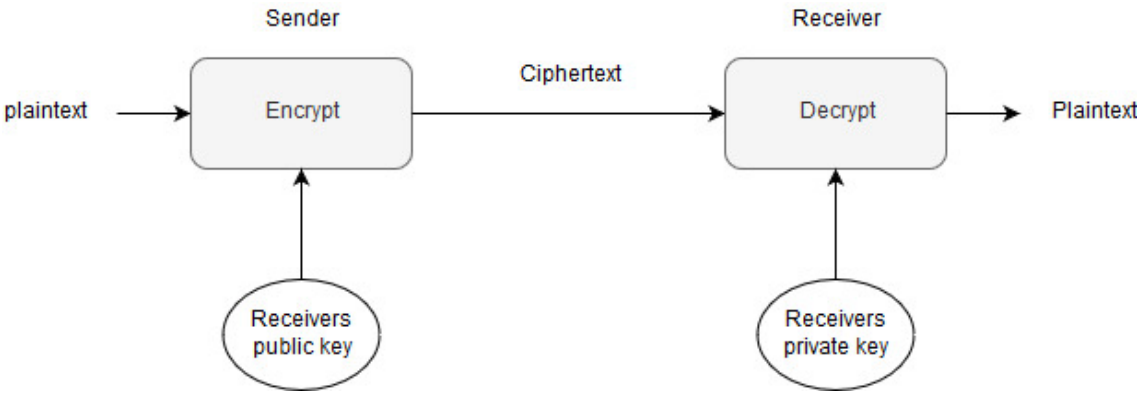


Figure 2.3: Public-key cryptography

However there are couple of disadvantages of the public-key cryptography in comparison with the symmetric one. First of all, the whole PKI is complex and introduces a single point of failure in the system, as if the security is breached of this entity the whole infrastructure would be insecure. Furthermore, the encryption is much slower when using asymmetric ciphers. This comes from the fact that that symmetric encryption and decryption is performed by switching the bits of the plaintext. On the other side, the asymmetric is performed through more complex mathematical operations, e.g. exponentiation. The speed comparison of one of the most efficient algorithms from the both categories, illustrates s that symmetric encryption is faster up to 100 times [24]. Therefore, a combination of the two approaches is used utilising their advantages, the public-key is used at the initialisation of the communication, with a purpose of securely sharing a symmetric key, which is

later used in securing the communication between the entities.

As already mentioned, the purpose of the certificate authority is certifying the ownership of a public key by the owner of the key. This procedure is conducted by issuing digital certificates, holding the identity of the owner of the public key and the corresponding key. Therefore the main potential goal would be keeping the integrity of a certificate, as there should not be possible any unauthorised change to be done in the certificate after it is issued by the authority. Without this property, anyone would be able to change the information held in the certificate, and thus putting their own uncertified keys in it. Asymmetric cryptography provides a way of ensuring integrity of the data by the mechanism of signing the data. This signing actually presents an encrypting of the data with a private asymmetric key. If the message is signed after its creation, the procedure of checking the integrity of the same message can be done by just decrypting the signature, with the corresponding public key, and validating whether the decrypted message is the same as the non signed message. Therefore, both the message and its signature are sent, in order its integrity to be checked. Additionally, the signing mechanism achieves authenticity, as the private key used for signing belongs to a known individual, furthermore, it also achieves non-repudiation, as the individual who signed the message can not deny the signing, as he is the only individual knowing the specific private key used for the signing.

However, in some of the public-key algorithms, a vulnerability exist when signing a large message, in which it is possible to change the order of the content of the message without disrupting the integrity of the message. One such an attack was noticed in RSA [25], one of the first public-key cryptosystems, in which if the message to be signed is too big it is divided in couple of chunks, `M=(m1, m2, m3)` and consequently their signatures `S=(s1,s2,s3)`. The attacker would change the places of the message and the corresponding signatures, `M'=(m2,m1,m3)`, `S'=(s2,s1,s3)`, which will lead to a valid signature but still it is clear that the integrity of the data was not kept. This clever attack, was one of the main reasons cryptographic hashes were introduced in the digital signatures, along with the fact that signing a hashed value is more efficient. How this hashing mechanism enforces integrity and its definition and other applications it founds in the security field, would be further explained.

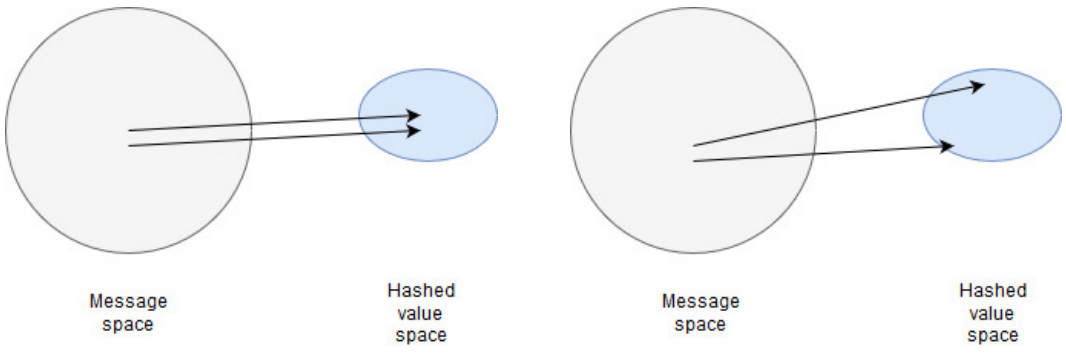
In the project, both symmetric and asymmetric cryptographic mechanisms are utilised complementing their flaws. The efficiency of the symmetric and the ease of sharing a public key of the asymmetric cryptography are exploited. However, their implementation is described in more details in the design section.

Cryptographic Hashes

Cryptographic hash function is a mathematical algorithm that maps, hashes, data of any size to a string of a fixed size, also known as message digest or hash, and it is designed to be a one-way function, which means that the data can not be generated from the message digest, it can not be inverted. According to Schneier, it is “the

workhorse of the modern cryptography" [26], as it can be used in many applications, in encryption and digital signatures, authentication, integrity checking etc. However, in order one hash function to be defined as cryptographic hash function it has to fulfill the following properties [25]:

- It has to be deterministic, the same message always results in the same hash
- It has to be quick in computing the hash from any given message
- Any small change to the message should change the hash value completely extensively, such that the new hash value is uncorrelated with the old hash, Illustrated on Figure 2.4
- *Pre-image resistance:* For any hashed value It has to be computationally infeasible to find any input which hashes to that hashed value, i.e. it is infeasible finding any x' such that $f(x') = y$ for any y for which the corresponding message is not known.
- *Second pre-image resistance:* It has to be computationally infeasible finding any message which has the same hash value as any specified message, i.e. for a given x, it is infeasible a second message $x' \neq x$ to be found, such that $f(x = f(x')$
- *Collision resistance:*It has to be computationally infeasible to find any two distinct messages which hash to the same hashed value, i.e. for any messages x, x' such that $x \neq x'$ finding $f(x) = f(x')$ is infeasible



(a) Correlated message chenges (b) Uncorellated message changes

Figure 2.4: Hashing uncorrelation

The larger gray circles show the message space, while the corresponding hashing set is presented with the smaller blue circles. The picture on the left shows a correlation in the hashing when some bits of the messages are changed. On the other picture, it is shown the uncorellatedness, which is required in the hashing algorithms.

Informally the three collision properties mean that a malicious entity can not modify the message without changing its hashed value. Therefore, if two messages produce the same hashed value, one can be very confident, that both messages are identical.

The second pre-image resistance ensures that attacker is unable to craft a message and produce the same hashed value. Collision resistance prevents an malicious attacker in creating two distinct documents with the same hashed value. However, the collision properties can be broken in some scenarios, but as already mentioned it is computationally infeasible in doing so.

The reason why the hashing functions are remarkable lies in the fact that the size of the hashed value is small in size, varying from 64 bits to 512, depending on the algorithm, while still holding the rules defining a cryptographic hash function. In addition, the hashing should be quick for any size of a message. As already mentioned, hashing functions are crucial in digital signature, and one of the main reasons is the small size of the hashed values. Because the signing is expensive operation, the message is first hashed and then signed, thus improving the efficiency. Additionally, the resistance properties are highly important in keeping the integrity of the message, as any changes done after hashing the message will result in completely different and uncorrelated values. For example, an malicious individual would not be able to change the original message and still provide the same hashed value. Moreover, the small size of the hashed value overcomes the aforementioned flaw of RSA, the integrity issue of signing a large messages.

Another application hashing have found its application is in storing credentials in a database, or in any kind of storing tool. In a scenario where the data is stored as it is, without any changes performed on it, breaching the security and stealing the data would reveal all of the sensitive data to the attackers. The one-way property of the hash functions, provides a solution for this issue. Credentials can be first hashed and then stored in the database, so whenever validity of some credential have to be checked, it would be first hashed and the value would be compared if the same as in the database. In case of stealing the data from the database, the attacker would not be able in reversing the credentials from their hashed values. Even though this approach seems secure enough, attack like look up table [27] would extremely effectively and quickly crack the hashes of the same type. The idea is, generating a hashes of possible credential values and store them, so later on these hashed values will be compared with the ones found in the stolen database. If any hashed value from the table can be found in the database, the corresponding credentials have been retrieved. Another more efficient modified look up table attacks exist, meaning that better protection is required. Credential salting [28], is a mechanism used in challenging these attacks. Evidently, hashing can be cracked in a situation when two identical messages are hashed, therefore, a random string, known as salt, is appended or prepended on the credential before hashing. This technique ensures that no two credentials would result in the same hashed values, therefore the look up table attacks becomes ineffective.

2.3.2 Access Control

Another important property a secure system needs to have is restriction of access to an information, only to authorised entities [29]. The authorisation have to be

enforced not only to persons but also to computer programs and in some cases the computers that process the information. The importance of access control even more increased with the emerging of the distributed and heterogeneous network systems, as many users can be connected on the network with different priorities and allowances. However before deciding if a specific entity has an authorised access, it has to be authenticated first.

Authentication

Authentication is a mechanism of verifying a claim of identity. For better explanation of the term, an example of authenticating an arbitrary Jack Beglin in a bank, would be provided. When Jack wants to withdraw money, he states his name at the bank teller, thus claiming his identity. The bank teller asks for identity card, to check whether Jack Beglin is written on it and compares the photograph on the card against the person claiming to be Beglin. If the name is correct and the photo is matching, then the teller has authenticated that Jack Beglin is who he claimed to be. In general, authentication is conducted similarly to the given example, just a different kind of information can be used for authentication. The authentication information can be divided in three categories:

Something the entity knows, such as password, PIN, signature with private key.
Something the entity have, such as identity card, magnetic card driver license.
Something the entity is, such as biometric, fingerprint, voice print.

Although, username and password credentials are most common form of identification, computer systems today try to provide more sophisticated mechanism of authentication. Biometrics, digital certificates and other approaches are increasingly popular, while in some occasions when increased security is necessary, more than one authentication information is required.

However, for the purpose of the IoT, biometrics, fingerprints, identity cards and other forms entity possessions are not suitable. Therefore, only the first category, something the entity knows is applicable. One of the most secure authentication mechanisms are utilising the power of cryptography. One scenario is employing an asymmetric cryptography, where each device would have their own unique private key, and by signing the message requests they would authenticate themselves. On the other side, symmetric cryptography is also applicable, where each device and unit in the network would have a unique pairs of symmetric keys. However, the first solution is not applicable, regarding the IoT, as the devices have limited capabilities the signing would be too demanding. The latter solution also have a big drawback, as the IoT network is capable of growing and incorporating many devices, establishing a key between each of them is a demanding task [30]. Therefore, the concept of username and password credentials are found to be most suitable for the IoT needs. However, more sophisticated mechanisms exist in protecting the credentials exist. One of them is incorporating the hashing methods, as explained before, another one is the zero-knowledge proof explained in more details in the conclusion/refconclusion.

Authorisation

Main purpose of the access control is to handle any request to data or services provided by a system and determine whether a permission should be granted or denied. In case it is granted, the entity is authorised in performing the desired actions. The authorisation service is able to determine which entities should be granted an access control through a set of regulations, policies, that can be stated. These policies are written in appropriate specification language and later enforced by the authorisation mechanisms. However, the definition of access control policies is far from a trivial task, as often the real world security policies, which are usually complex and ambiguous, are hard to interpret in a well defined rules enforceable by a computer system. For this reason each access control language expressing security policies needs to integrate the following features [29]:

- The policy language has to be defined using a format understandable both for human and machine, easy to interchange and inspect
- It should support both fine-grained and coarse-grained specifications. It should allow providing an fine-grained reference of subjects and objects in the system. However, fine-grained should not be enforced, it has to be only supported, as specifying access rules for each subject and object would make the task a heavy burden. Therefore, it should also provide a coarse-grained definition of groups of subjects and objects
- The policy has to enable conditional authorisation, meaning that the language should support authorisation decisions to be depend on evaluation of some conditions. These conditions can be in a different form, such as date, location, age etc. An example of conditional policy would be allowing the users to access the service from 9 to 12 in the morning
- Combination and conflict-resolution is also an important feature of a access control language. As multiple policies can exist in the system, the language should provide a means for specifying how different policies would interact with each other. For example whether the union will dominate, more permits or denials, their intersection, the minimum defined privilege, in some cases denial can overwrite the permits or vice versa

When it comes to choosing a proper authorisation language there is a couple of different types. At the beginning, it was realised the power of logic languages in enforcing the authorisation. One of them is the Flexible Authorisation Framework (FAF) [31]. It is a powerful and elegant logic-based framework which is based on a access control model. Any policy can be represented through the syntax of the model. However, FAF and the other logic languages are hard to understand and implement, therefore more clear and simplistic languages are needed. Therefore, the community proposed a XML based access control languages [32]. Additionally, a big advantage of the XML based languages is the interoperability, as exchanging of the policies and requests is conducted throughout the widely adopted XML data structure.

One of the most relevant languages of this type, which are fulfilling the previously defined features that an policy language should have are WS-Policy [33] and eXtensible Access Control Markup Language XACML [34]. Although there are many similarities between the both, the latter have an advantage to express variety of different policies, including the basic functionalities of the most policy languages. Additionally, XACML provides a mechanisms of extending and defining new functions, data types, logic, etc. Therefore, it was concluded that XACML is most suitable for the need of the project.

The language is an OASIS standard based on XML, capable of expressing and interchanging access control policies. It is designed to express XML policies against and requests from objects that are themselves also in XML format. Despite the already described features, XACMl is providing an additional features:

- It supports the policies to be defined based on attributes describing the subjects and resources other than their identities. This enables definition of more powerful and strong policies, as additional properties like name, address, occupation etc. can be associated with the subject. These attributes can be utilised in the actions these subjects take, and the objects they try to access
- XACML allows defining a multiple subjects in a policy, relevant to a decision request
- Policies support distribution, meaning that different entities can define them and they can enforced at different points
- The language have built-in operators for comparing attribute values but additionally supports defining a non-standard comparison functions

Defining the main features give an overall insight of the language, on the other side, the data flow model, describing the main entities responsible for storing, managing and enforcing the policies would give a better understanding of the insights of XACML. Each entity would be defined further below. **Policy Evaluation Point (PEP)** is a point which intercepts the users access request to a resource, sends a decision request to the PDP to obtain the access decision, whether the access is permitted or denied, and acts according the received decision. **Policy Decision Point (PDP)** receives the access request, evaluates it against the authorisation policies and issues a decision, permission or denial. In the evaluation process PDP interacts with PAP that holds the information needed to identify the applicable policies. **Policy Administration Point (PAP)** is responsible for managing the authorisation policies, it retrieves the policies applicable to a specific request and returns them to PDP. **Policy Information Point (PIP)** stores and manages the attribute values about the subject, resource and action.

Figure 2.5 illustrates the data flow, in which a entity, named John, is requesting an authorisation request in accessing a specific record #2258. At the begging the request is sent to the PEP unit, which intercepts it, converts the request to XACML

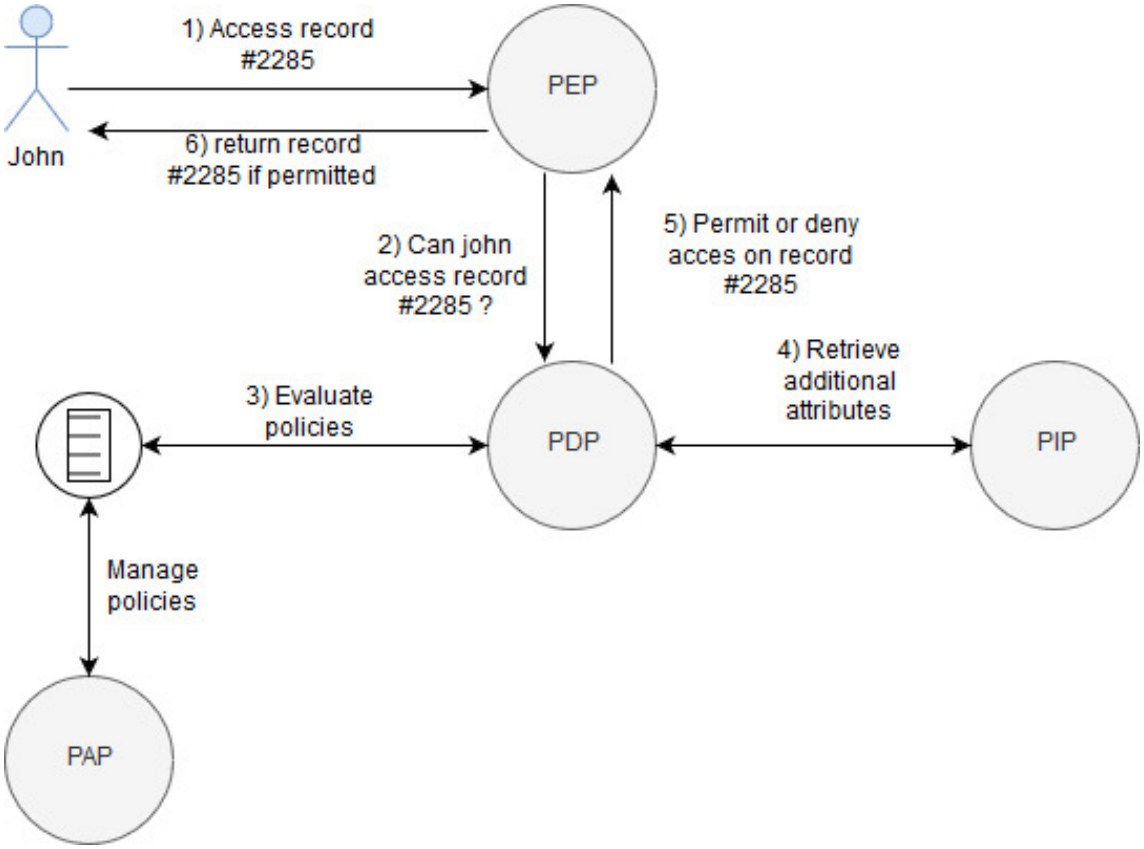


Figure 2.5: XACML data flow

authorisation request and forwards it to the PDP unit. PDP evaluates the request, for this purpose it interacts with PAP which retrieves back the policies applicable to the specific request. Additionally PDP retrieves the attribute values from PIP. After gathering this information the unit reaches a decision in one of the four possible formats, Permit / Deny / NotApplicable / indeterminate, and returns it to the PEP. In the end, the the entity is notified about the decision, from the PEP.

2.4 Communication Protocol

For IoT devices, internet connection is a requirement. Connecting the things to the internet allows them to work with each other and with the backend services responsible for further storing and analysis of the data generated. Therefore, implementing an suitable communication protocol for the IoT is a necessity. By suitable, it is meant a protocol which would balance the weaknesses of the devices, hence, it needs to employ the following characteristics:

- The protocol have to be lightweight, allowing it to be implemented on heavily constrained devices and on high latency or limited bandwidth networks
- It has to be flexible, making it possible to support diverse IoT applications and devices
- The protocol has to be asynchronous, as the high number of devices and the usually unreliable or high latency network in which they communicate, have made the synchronous communication infeasible. Therefore, in the asynchronous approach, the devices can send their generated data and let the other parts of the network figure out the path and the right timing for delivering to the destination
- Broadcasting is a frequent action in the IoT, therefore the communication have to be developed to support many-to-many message transferring, as the one-to-one way is difficult and expensive in handling such a broadcast events

However, an obvious question can occur to the reader, why not using the mostly used and highly popular HTTP protocol, as it is easy to integrate and work with, additionally there is not a need in creating and studying a new communication protocols. However, comparisons [35] [36] are showing that one of the most popular IoT communication protocols, MQTT, performs better in a scenarios of sending data frequently, as can be met in IoT, with efficiency up to 15% faster while consuming around 22% less energy.

In the project MQTT protocol was utilised for the communication of the IoT devices, having all of the characteristics for a suitable IoT protocol. The decision of choosing the specific protocol, among the many IoT protocols existing, highly depends on the system architecture, therefore the reason of choosing MQTT is described in the design chapter. The protocol fulfills all of the required characteristics regarding IoT communication protocol listed above, an the way they are achieved would be described in the following chapters.

2.4.1 Definition of MQTT Protocol

MQTT (Message Queue Telemetry Transport) is a lightweight, ISO standard publish/subscribe message protocol. As already described, it is designed for work with limited capabilities devices and in networks with high latency and limited bandwidth. The protocol was originally invented by IBM in the late 90's [37], developed for the needs of linking sensors on pipe oils with satellites. However, since late 2014 it is officially an OASIS standard [38]. The protocol is supported in many programming languages by using the multiple open source implementations. In addition it is based on TCP/IP internet protocol, which is the basic rule defining the internet.

The primary objective of the MQTT architecture is decoupling the senders from the receivers of the message, realised by the publish/subscribe model. The senders of the message, also known as the publishers, are completely not aware of the receivers of the message, the subscribers, and vice versa. Therefore, the connection between them is handled by a third party component, known as the broker. The responsibility of the broker is all incoming messages from the publishers to be distributed to the subscribers. Main benefit, coming from this decoupling, is the possibility of adding new components in the network, without knowing about each other, suitable for the large scale of IoT devices. To publish and receive message the publishers and subscribers only need to know the IP address and the port of the broker. It is worth mentioning that many clients can subscribe and publish on the same topic, making the protocol suitable for many-to-many communication. The broker additionally provides the asynchronicity property of the protocol, as it decides when the message would be delivered and to which receiver, in the same time relieving the publishers of these concerns.

So far, it is clear that the responsibility of the broker is routing the messages, however filtering must exist so that each subscriber receives only messages of interest. This filtering is conducted through topics that are part of each message. The receiving client can subscribe to the broker for any topic of interest. Figure 2.6. Illustrates the publish/subscribe model, in a scenario of a client publishing data on a temperature topic, to the broker, which is later published to clients subscribed on the same temperature topic.

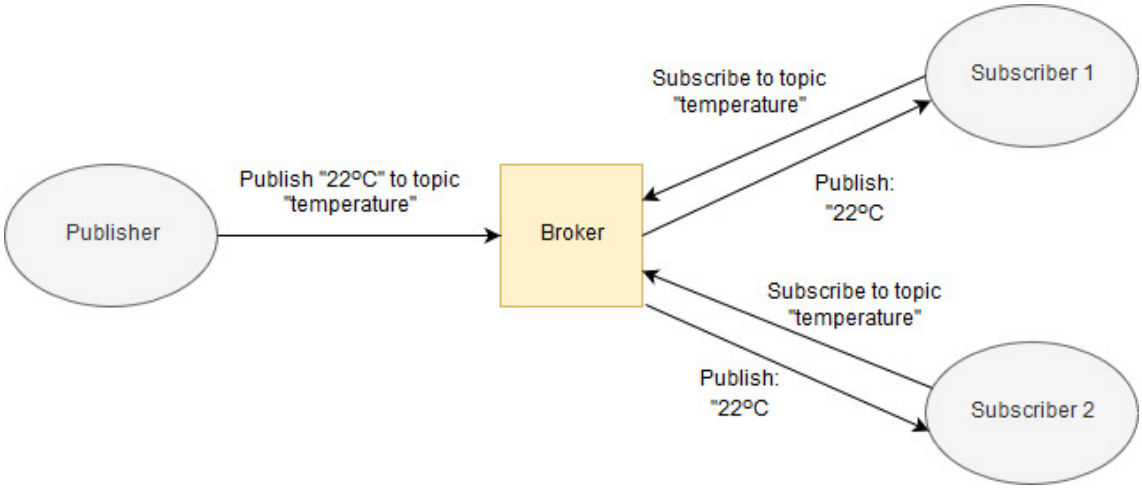
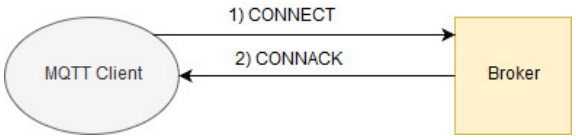
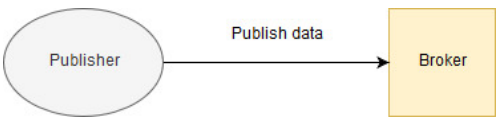


Figure 2.6: MQTT publish / subscribe

However, an obvious question can occur to the reader, why not using the mostly used and highly popular HTTP protocol, as it is easy to integrate and work with, additionally there is not need in creating and studying a new communication protocols. Comparing it to the list of the characteristics needed an IoT protocol to fulfill, HTTP does not fulfill mst of them, having a efficiency issues, which will be discussed in the following chapter. In the project MQTT protocol was utilised for the communication of the IoT devices, having all of the characteristics for a suitable IoT protocol. The decision of choosing the specific protocol, among the many IoT protocols existing, highly depends on the system architecture, therefore the reason of choosing MQTT is described in the design chapter. The MQTT handshake is illustrated on Figure 2.7, where both, publish and subscription, are preceded by the connection procedure.



(a) Initial connection



(b) Publishing a message

(c) Subscibtion to topic

Figure 2.7: MQTT handshake

Both publish and subscription events start with connecting to the broker.

The lightweightness of the protocol comes from the small number of steps required the client to establish connection with the broker. The connection is performed over two steps, at first, the client sends CONNECT message which is followed by the brokers CONNACK. The connect message is required to hold two mandatory

fields, the clientId and the clientSession, which will be described in details in the following section. After the client is connected the publish of a message is only done through one step, publishing the desired action. However, on the other side, the subscription is performed through two steps, the subscriber sends SUBSCRIBE message, stating the topics it wants to listen to, which is followed by SUBACK message from the broker. This procedure enables the subscriber to receive any message on the previously defined topics.

2.4.2 Properties of MQTT

Despite the main features of the protocol, MQTT provides few additional properties, making it even more suitable for the IoT. One of those properties is the Quality of Service (QoS) [39], agreement between the sender and the receiver of the message that defines the guarantee of delivery of the specific message. There are three QoS levels in MQTT, which will be described in the following paragraphs, under each graphic illustration of the messages sent would be provided.

QoS 0: at this level, the message is sent at most once, without any guarantee of delivery. This means that the broker does not acknowledge the client that the message was received, therefore if a message is lost during its transfer it would not be retransmitted again. This level is the most efficient but most unreliable in the same time.

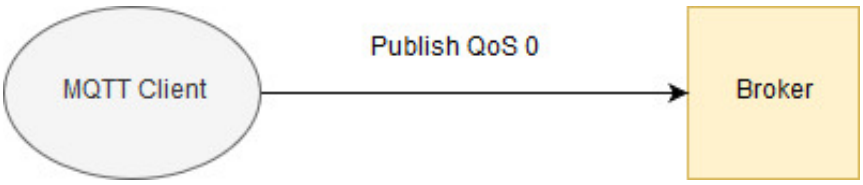


Figure 2.8: QoS level 0

QOS 1: this level guarantees that a message would be send at least once. The sender sends a message to the broker and additionally it stores it until it gets PUBACK packet back, acknowledging the receipt of the message. In a situation where the sender does not receive the PUBACK in a specific amount of time, it sends the message again. This way the message is guaranteed to be sent, however if the PUBACK was send but not delivered in the specific amount of time, the message sent again would emerge as a duplicate at the broker, as there is not a mechanism for checking the duplicate messages. This level is not as efficient as the QoS 0, however it provides a better reliability.

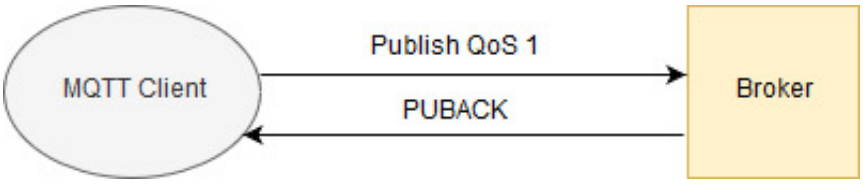


Figure 2.9: QoS level 1

QoS 2: is a level which guarantees that each message would be received exactly once, meaning that duplicate messages, which could have occured in the previous level, are discarded. QoS 2 begins the same, with the client sending the message and waiting for a PUBREC packet back. If the PUBREC is not received, the message is sent back again, however, this time it is sent with a duplicate flag. This procedure is repeated until the sender receives the PUBREC packet back from the broker. After receiving this packet, the sender can safely discard the initial message. Next step is sending back a PUBREL message to the broker. After receiving it, the broker can also safely discard all stored states and send the final PUBCOMP packet to the sender. After receiving it the sender also discards all previously stored states. This procedure ensures that the message is sent, and it is only delivered once at the broker side. In contrast to the level 0, this is the most inefficient level but in the same time the most reliable.

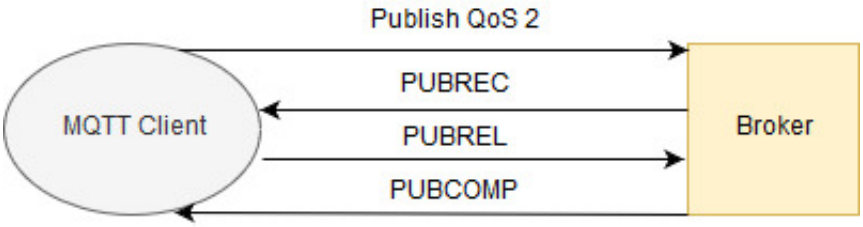


Figure 2.10: QoS level 2

As it can be seen, different levels of QoS provide different efficiency and reliability. In a scenario where the most effective way is needed, without any concerns about losing a messages, the most suitable solution would be QoS 0. On the other side, if the quality of the message is priority, and none of the packages are allowed to be lost, the highest level of QoS would be the preferred solution.

Additionally MQTT have mechanisms in providing persistent sessions between the broker and the subscribers [40]. As already mentioned in the previous chapter, when the connection is initialised between these two components, the subscriber can request a persistent session by setting the clientSession tag to false. In persistent connection, if the connection between the broker and the subscriber is interrupted, all the topics of the latter, and the messages that should be sent to it are saved until the connection is reestablished. In contrary, in non persistent mode, this information is lost and have to be initialized again in a situation of interruption. This re-initialization is a burden on a constrained devices and high latency networks.

2.5 Similar Projects

Findings of a similar projects and analysing their structure and design is of a big importance in order all the requirements of the project to be fully understand, in addition, many suggestions and ideas could come from the already implemented systems. There are many projects which cover IoT, Big data and security, however

combination of all of them leading to a more narrow choice. Some of the similar projects findings are described in the following sections.

2.5.1 Microsoft Azure, Google Cloud and Amazon AWS

One of the leading technological giants, have already implemented a complete solutions when it comes to creating a services with a purpose of storing and process big amounts of data generated by IoT devices. Such a solutions can be found in the products such as Microsoft azure [41], Google cloud services [42] and Amazon AWS [43]. All of these services, provide a cloud solutions for different needs and areas, one of them is the IoT. They provide interface, rich in features and possibilities, capable of storing processing and analysing big amounts of IoT generated datasets. Additionally, a lot of already developed and tested processing and analysis tools are offered. The security challenge is considered in their solutions, providing a mechanisms of achieving the main security goals (confidentiality, integrity etc.). The communication is protected using the already existing PKI on the internet, and the X.509 certificate standard. Also, access control mechanisms are offered, without detailed explanation of the way it is implemented on the services.

Although these solutions provide all of the needed features and properties a specific Iot and Big data service requires, one of the most important security goal, data privacy, is regularly violated by the companies developing these solutions. According to Schneier [44], and many times publicly confirmed, the private data they gather is usually used or sold for further analysis. The occurrence of the new GDPR (General Data Protection Regulation) confirms these data exploitations even further. Therefore, before considering of using these services it should be determined whether these companies are reliable and trustworthy of keeping a sensitive data.

2.5.2 Architecture for Secure E-Health Applications

One similar architecture can be found in the academic paper [45], which elaborates a solution for developing a secure E-health application. The main vision of the project is implementing many sensors on the human body, creating a network known as Body Area Network (BAN), which will measure the biological information it gives, and then send the data to a service on the internet for analysis. The goal is to check the state of the patients and to find or predict any diseases in the body. One of the main objectives of the project is finding a way of secure data transfer and storage, as the authors suggest that the privacy is the most important aspect when people body information is gathered. Their main suggestion is utilising the enhancements of the public-key cryptography, however the keys used data encryption are generated on a very specific and unique way. The approach which is used is called Physiological Signal Based key Agreement. In which the sensors placed on the body generate a key based on the physiological signs of the body, such as heart rate signals or ECG signals. As these signals are unique for each human body, a distinct key can be generated for each individual. Additionally, the services containing these

vital information about the individuals, can regenerate the key using the stored physiological data.

The security decisions taken are highly strong and are able to construct well protected system. On the other side, the communication is conducted using the HTTP protocol. It was already mentioned, that this protocol is not efficient when it comes to sending small packages in very short time intervals, scenario which can be seen in the E-health project. Therefore, some of the communication protocols specially designed for the needs of the IoT network should be utilised.

2.5.3 A Novel Secure IoT-Based Smart Home Automation

IoT devices can be further used in home environments, providing couple features, controlling the different electrical devices in the house, calibrating the temperature of the air conditioner or the fridge, controlling the lights or turning on any device from any place. Another usage of these devices can be found in storing the data and analysis of energy consumption, or habits and time spent on watching tv or using phone etc. One architecture, suggesting a way of incorporating IoT devices in fulfilling this goals is described in [46]. Like in the previous e-health project, the researchers concluded that the security is the most important challenge of the project. This was concluded from the fact that many home appliances can be controlled over the network and private data about the individuals owning the household is collected. Additionally, the project suggests that, because of the resource constraints of the devices, an effective encryption algorithm should be specified. For that purpose, Triangle Based Security Algorithm (TBSA) is adopted in the home automation project. However, the main drawback of their architecture is the need of using different key for each device, which can lead to key management issues and lower efficiency in case many IoT devices are connected. So far their implementation is developed for the needs of a small household, therefore these issues would not lead to a big issues. On the other side, the idea of the researchers was adopting their architecture in more complex projects like smart cities and health organisations. Consequently, the issue in their key management approach, would lead to big issues and drawbacks if integrated in the mentioned complex projects. Nevertheless, their idea of choosing an efficient encryption algorithm should be requirement in any project offering IoT solutions.

2.6 Summary

This section is a result of an in depth literature review and serves as a basis for the IoT service to be designed. Therefore, at the beginning it defines the term IoT and gives an insight of its importance and the enormous impact it would have on our future. Its main challenges and application are also illustrated. As these devices, part of the IoT network, produce enormous amounts of data, definition of big data and its challenges was naturally followed to be described. In addition as the main challenge of the IoT is its security, a detailed research was conducted on this field, giving a valuable information of what does security means and what

are the main protection goals. Afterwards different mechanisms and tools needed in achieving these goals were explored and the most suitable for the specific project were disclosed. One of the main tools which enforce the protection goals are cryptography and access control, both of them providing crucial security measures. In the end as the IoT devices have limited capabilities, and operate in network with high latency and limited bandwidth, a specific communication protocol was defined, answering these challenges.

Chapter 3

Problem Solving Process

In the chapter that follows a succinct presentation of the main vision of the project would be provided. Further, the research methodology, responsible for effective acquiring of the essential knowledge, would be described in detailed steps along with the tools used. The project management and its importance is defined. In addition, the followed software process model would be described, with all of its important phases. In the end the main risks of the projects would be described along with their management and the different approaches taken in mitigating them.

3.1 Vision of the Project

This section would introduce the main vision of the project. In order a project to be successfully developed, there is a need of a vision to be defined, guiding the developer in accomplishing it, without putting a blindfolded effort. The vision of the project is highly linked with the aims and objectives of the project, as those form main evaluation of success of the project. Therefore the vision of the project is designing and developing a service which enables secure transmission, storage and processing of big amounts of data collected from various kinds of IoT devices. In addition only authorised devices should be served on the service. However, in order the project to be successful it has to answer the main challenges and issues of the IoT.

These challenges, were primary source of inspiration and motivation of developing the specific project. The in depth literature review conducted, documented in the previous chapter, gave an insight of the issues and the approaches that have to be taken in overcoming them. It was concluded that the main challenges are related to storage, processing power, security and communication.

1) Storage and Processing Power as the IoT devices are constantly surveilling the environment big amounts of data are collected which further need to be stored and processed. One of the simplest solution would be these processes to be managed in the devices, however, as described in the literature review, most of the devices are with limited resources and capabilities. Therefore, a specific service, designed for processing and storing the produced data is essential in order the whole IoT idea

to be put in practice. The fact that, even a simple sensor device can overwhelm the storage capacity of a machine in a short period of time, implies the need of a distributed system of machines, in storing the collected data. The occurrence of enormous datasets and the corresponding system developed for managing those datasets introduced the new big data era. Consequently, it can be seen that IoT and big data are highly related topics in the information society, as both fulfill each other in fulfilling their potentials. Therefore, it is evident the need of utilizing a distributed big data processing tool, responsible for grasping and managing the datasets generated by the IoT devices.

This distributed system consist of clusters of machines, placed on different physical locations, capable of processing and storing the data parallely distributed on different clusters. The invention of tools capable of orchestrating such a parallelism, enabled the idea of IoT and other big data generating paradigms to become reality. Additionally, as these tools work over a distributed network of machines, single point of failure can be easily challenged, by creating backups on different clusters and nodes contained in those clusters.

2) Security was recognised as a highly important area in the research conducted. Securing the IoT network, was noticed as a crucial requirement in order the network to fulfill its potential.These remarks were concluded by many IoT experts along with their colleagues working in the field of security. The IoT devices can find applications in many areas and industries, therefore in many occasions private and sensitive data can be collected by them. This implies the crucial need of secure transfer of the data from these devices to the service responsible for further processing and storing. The literature review suggest two different cryptographic approaches, in achieving protected transmission, suggesting the public-key cryptography as more suitable in environments in which entities which communicate are large in numbers, and sharing a secret key between them, before any communication is infeasible. Such approach is suitable for the IoT network, as everyday the number of the devices in the network is increasing, with expectation of this fashion to continue in even higher rates. On the other side, the limited capabilities of the devices demand lightweight and more efficient algorithms for data encryption. Therefore, the ideal solution would be a mixture of symmetric and public-key cryptography, in which the latter would be used only for sharing symmetric keys, method already described in the review.

The enormous number of devices introduces a new threat, which in this case is not coming from the outside of the network, but contrary from the inside. Unauthorised access to a services resources and data can be a big issue, if it is known that the big data services keep sensitive information. Therefore, access control mechanisms are crucial, protecting from this type of vulnerability. The mechanisms provide a selective restriction of access to the data and resources on the services. Unfortunately, this is not an easy task, as the network can contain millions of connected devices, making the access control highly complex. Access control models and languages challenge this issue, decreasing the complexity by providing policy defining mechanisms. The languages enable defining a coarse-grained policies, making it achievable

to combine many of the devices in groups, thus grasping the issue of authorising vast number of device.

3) Communication Despite the fact that many communication protocols already existed before the occurrence of the IoT, the challenges it faces requires a different approach when it comes to communication. The environments in which the devices are implemented can vary, some of them are suitable and optimal for transmission of the data collected, but on the other side many devices are placed in environments with limited bandwidth and high latency. Therefore, in general lightweight communication protocol is required, answering these environmental challenges. In addition, the protocol is required to support the continuous growth of the numbers of devices, providing a mechanisms of easy integration of new devices in the network. Therefore, the communication protocol chosen needs to fulfill the two mentioned requirements, in order to be suitable for the specific project.

Challenging the described main issues of the project, will lead to fulfill its aims set. The Figure 3.1 illustrates the main challenges, and the initial vision of the project as an outcome of the challenges. Illustration was provided, as a conclusion and for the reader to better understanding the vision of the project, but additionally it provided a more general guidance for further development.

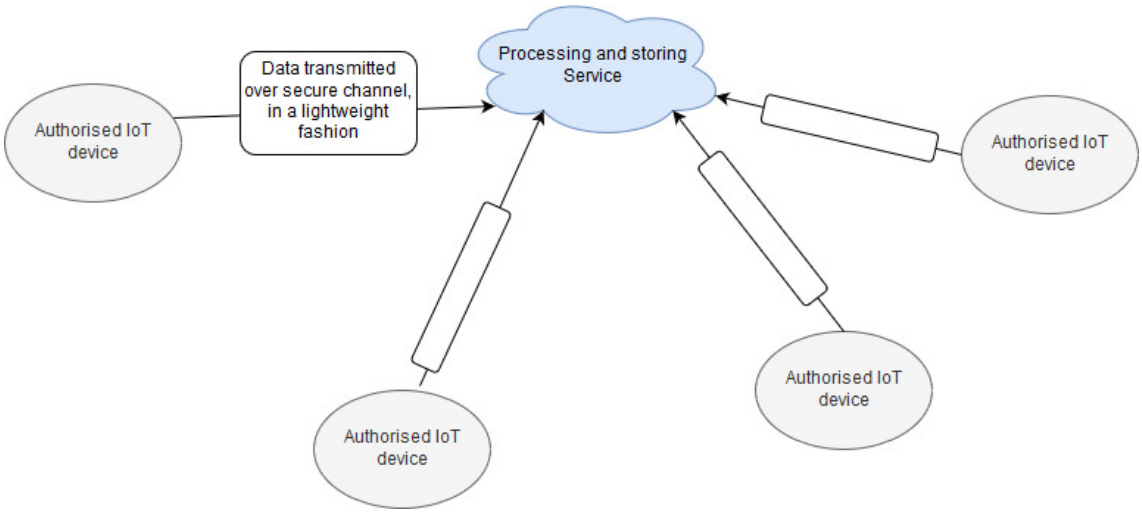


Figure 3.1: The vision of the project illustrated

3.2 Research Methodology

In order a quality and exhaustive research to be conducted a certain methodology have to be followed. Clearly many different methodologies exist, depending of the nature of the issue and the current skill of the researcher. First, a proper approach needs to be created for storing and managing the researched resources and papers. For this purpose, a specific tool Mendeley [47] was utilised, providing a ways of managing and sharing research papers, discovering new data and organising articles

using easy tagging. The tool was of a big benefit, providing assistance in organising neatly and going through the research conducted with ease. However, the tool would provide just a better support in conducting the research, strict steps have to be followed in order a circumstantial research to be provided.

At first, the focus have to be in identification and definition of the problem. This step have to integrate a closer look at the problem, and identified have to be the most important issues regarding the specific project. Defining the IoT and all of the challenges it has to overcome.

The second step is problem analysis, gathering the most important information and characteristics of the problem, leading to clarification of the issues and better understanding of its challenges. Further, it is also necessary a thorough research of similar topics and issues to be conducted, which will create a clear vision of the possible solutions and alternatives of the specific problem, and moreover what are the additional obstacles that could arise further on in the project.

Consequently, after the initial analysis, follows the definition of the aim and the objectives of the project, which form the basic guidance and framework for the further development, additionally providing a criterion of the success and completeness of the project. The objectives provide a basis of what features and functionalities that need to be fulfilled.

In the end, after defining the aim to be achieved, all different approaches and fitting solutions have to be reviewed, in order the most suitable approach to be chosen. This step requires succinctly sketching all the alternatives with just enough information to provide detailed evaluation. After each alternative is evaluated, it is easy to choose which is the most suitable solution for the specific project. In the evaluation different criterias have to be considered, the level of fulfilling the main aims and objectives, the complexity and risk of the specific solution and the performance evaluation in comparison with the other possible solutions.

3.3 Project Management

The basis for successful project, in any engineering area, is its management. The lack of detailed software management is resulting in higher costs, delayed product completion and decreasing of the software quality. The fact that proper project management is crucial in the future success of the project was proven with the emergence of the software crisis 1960s and 1970s [48]. In that period, as software engineering was at its emergence, project management was not considered in developing software products. The lecture learned from the crisis was the need of project management, as the software was getting bigger and more complex, making it infeasible to grasp it without any documentation and management. However, good project management does not always leads to a successful project, but on the other side poor management is always a warranty of failure [49].

Since the software crisis, a framework for a good project management was developed, containing four main activities [50]:

- **Planning:** this activity starts before the production of the software, defining concrete activities that has direct connection with the software production. In this phase the resource needed are defined along with developing a development plan. Main focus of this phase is planning the analysis of the issues, design of the solution, defining the software development process.
- **Scheduling:** this activity comprises all the tasks required in completing the assignments, additionally assigning them a specific time slots. Suitable way of scheduling representation are Gantt Charts.
- **Risk management:** This activity is responsible for detecting the risks of the project. Risk management is essential for every project, in order all the possible risks to be known and proper plan and manner in which to be conformed to be constructed.
- **Human resource management:** This activity is Important for any sized project, even for small number of people. However, as the number of people grow, it is becoming more important and complex task.

3.3.1 Process Model

Software process model is one of the most important activities in the project management, as it forms the ground for quality product, representing a framework of the software project. The software process model provides a guidance of all of the phases the project should proceed through and accomplish, while its development. The importance of defining the vision of the project is reflected in this phase, as the nature of the project and the challenges it has to overcome are crucial in defining the most suitable process model for the specific project. Moreover, employing a suitable process model is proven to be very important, by the fact that its choice affects the determination if the the project will accomplish the estimated costs and time.

3.3.2 Waterfall Process Model

The requirements of the project were strongly defined at the beginning of the project, which was not intended to be changed over time. Although minor changes are possible to occur, the main aim and objectives of the project were known at the beginning, and their completeness estimate the success of the project. Further, the project required a well elaborated documentation after completion of each phase. Therefore, a process model which will provide these mechanisms of documentation along with stable definition of the primary project requirements was needed.

These requirements strongly suggest that the Royce waterfall [51] process model is the most suitable model for the specific project and others similar to it. Although, this process model is unpopular and criticised by the community, in case it is properly utilised it provides many benefits to the project. One of the main reason of his

infamousity is the fact that from the beginning it was not implemented as Royce was suggesting to. In his article, describing the process, the first illustration, which is generally known as the waterfall model, was actually a criticism how the process model should not be implemented. From this fact, it is understandable why the widely known "waterfall model" have such a low success rate. Figure 3.2 illustrates the stages of the waterfall model. The main idea of Royce was describing a process model which would enable going back or advancing to each phase of the project, without completing the current phase in total. This way providing a more flexible process model, which would enable changes of requirements during design, or changes in the design during the testing period. However, the wrongly understood model was implemented without providing this flexibility and iteration through different phases, making it impossible any changes of the requirements to be made after completing this phase. Each of the phases would be briefly described in the following bullets:

- **System requirements:** This phase is responsible for defining the main goals and objectives of the system. Their definition had to be conducted at the beginning of the project, and until completing this phase, the net stage should not be started. In order this phase to be completed, the aims and objectives were defined in the section 1.1.

- **Software requirements:** In this phase the software requirements are defined, in which issues related with the functional requirements of the system are described, along with the non-functional requirements it has to fulfill. As these requirements can vary over time, this phase can not be completely closed, and there is a need to be revisited couple of times after analysis of the problem.In the project, this stage is described in sections 4.2 and 4.3.

- **Analysis:** After defining the system and software requirements the next step is the analysis of the project. In this stage the definition of the problem is defined, along with the general requirements the project needs to fulfill, in which the functional and non-functional are taking part. As the requirements and the understanding of the project can change, this phase can be revisited couple of times during the development. This phase also forms the framework for the further development, forming the backbone of the design phase. The analysis process in the project is shown in chapter 4.

- **Design:** This stage focuses on the development of the system design. In this phase the detailed architecture of the system is designed, and decisions are made why specific design actions should be taken, forming the guidance for development of the softwareThis stage in the project is described in chapter 5.

- **Implementation:** After creating the system architecture and deciding the main design issues, the implementation can be performed. In this phase all of the previous work would be accumulated and the success of the analysis and design would decide the success of the implementation. After finishing this phase, the system would be ready for testing and revisited in a situations where issues would be found. The outcome of the implementation in the project can be viewed in section 6.

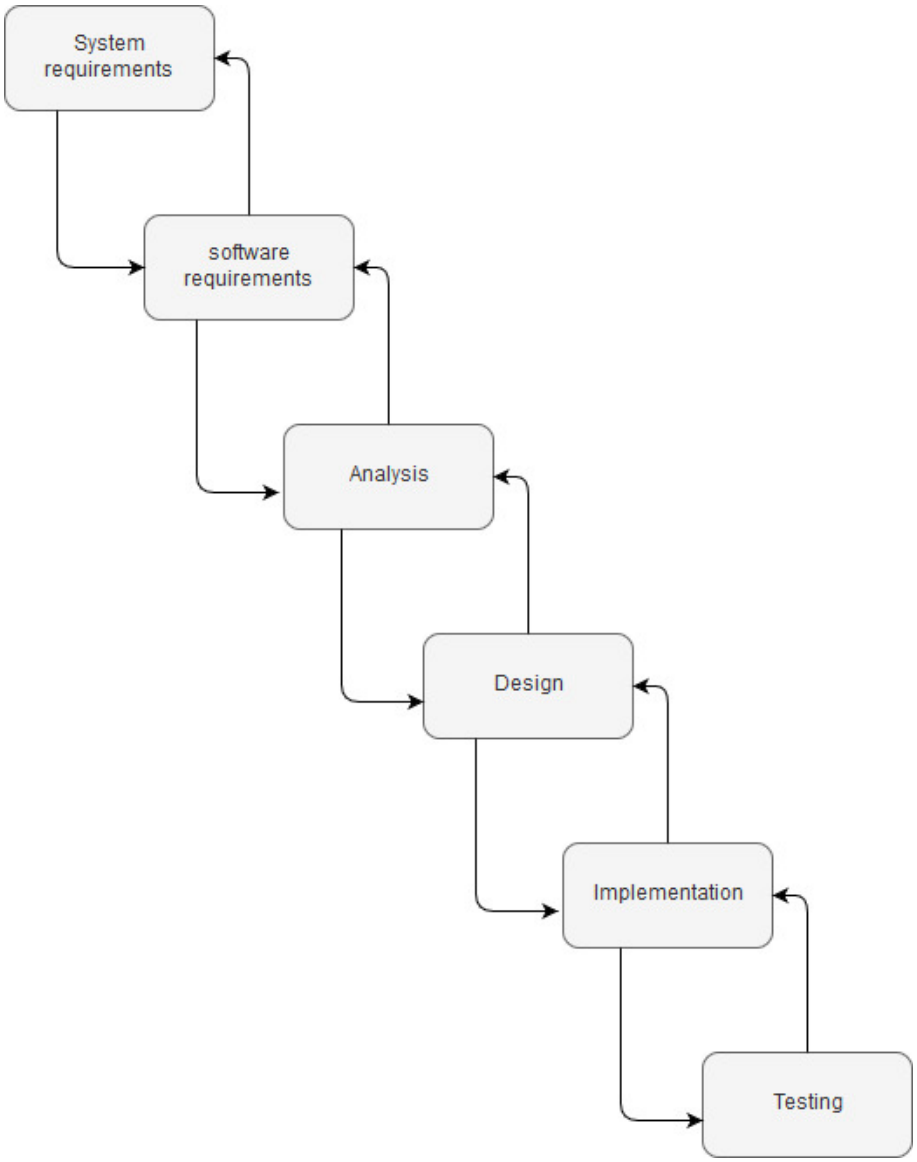


Figure 3.2: Waterfall process model

- Testing: The last stage of the waterfall model is the testing model. This phase includes different testing approaches, at first the implementation needs to be tested, whether working correctly without any stallings and errors, testing the functional requirements defined. Next, is the testing of the non-functional requirements, or testing of the performance of the system developed. In the end, the most abstract testing would have to be conducted, responsible for validating whether the main aim and objectives of the project were successfully fulfilled. The testing in the project is documented in section 7.

The description of the different phases illustrates how highly connected and dependable each phase is. Therefore, for the success of one phase, all of the others have influence. Additionally, the possibility of iterating through the phases is of a big benefit, as the requirements and design decisions can not be set in stone, meaning that their modification is expected.

Waterfall model has many advantages over the other processing models when used in an appropriate fashion. It is easy to understand, therefore it is easy to be implemented well. Further, it organises the stages gradually, first the requirements are set, providing a basis for the design, which forms the framework for a further guided implementation. Therefore, the requirements are clarified at the beginning and the design errors could be noticed before any implementation have initiated. Additionally it can be followed by small teams, which does not have much experience. The cost, time and risk estimation can be accurately conducted at the beginning of the project, which is from a great importance for the specific project, as the time frame is strictly defined. In the end, the testing and evaluation of the final product can be easily performed, when the requirements are strongly defined at the beginning of the project.

Aside from the advantages of the waterfall model, couple of drawbacks exist which have to be mentioned when implementing this model. First of all, many times the wrongly defined waterfall model have been utilised, therefore the proper version of the model has to be carefully researched Additionally, many time overestimations and underestimations can occur, especially in a teams lacking of experimental background. Moreover, it is realistic to have a wrong estimations about the requirements at the inception phase of the project [52].

3.3.3 Project Plan

After determining the main aim and objective of the project, and the process model which would be adopted, next step is creating a detailed plan, which would be followed throughout the development of the project. Following a specific plan gives couple of benefits, it helps better estimating the effort required in completing the project, reduces the risk and the work done is more visible and measurable. which can be essential in the further planning of other projects, increases the likelihood that the project would fulfill its determined aim. The fact that the Royce waterfall model is utilised, the planning should include all of the stages of the model into

a specific time frames. At the beginning, the plan was more of a general, succinct document, providing a skeleton for the project. However, after having a better of the structure and the nature of the project, a more detailed plan was constructed with strictly determining the time frames of each step in the development. Although, in some occasions there were small deviations from the planning, in general the development was following the initial planning agreement. Figure 3.3 illustrates the detailed project planning, describing the time frames of each step taken in the development process.

3.3.4 Risk Management

Software failures are result of multiple risks inherent in the software project development, therefore, risk management must be part of any software project in order to have a quality product satisfying the time and requirements agreed [53]. First of all, the general risk level of the project needs to be evaluated. The specific project should be considered as a medium-risk project. This level given could be concluded by a closer look of the potential risks that might occur during the project development. The aspects which are affecting the risk are the following:

- Complexity and size of the field
- Integration issues
- Lack of expertise in the field

After identification of the main weaknesses of the project, next step is identifying the risks that can occur from these weaknesses. Further, it is highly important to estimate their influence and the probability of occurrence. These estimations are crucial in avoiding the most critical risks and finding an alternative strategies of mitigating them. Therefore, in order the mitigation to be more successful detailed definition and estimation of the risks should be provided. Figure 3.4 illustrates the possible projects risks, with the probability of occurrence of each risk and the effect it would have on the completeness of the whole project. The risks are measured from 1 to 5, where higher number indicates higher effect it would have on the completeness. In the end the figure illustrates the exposure, measuring the probability multiplied by the effect. In the following bullets, each risk is elaborated in details and different approaches for their mitigation are described.

Priority	Risk	Probability	Effect (1-5)	Exposure (Probability x Effect)
1	The final system does not meet the requirements	20%	5	1
2	Failing to provide access control mechanisms	35%	4	1.4
3	Failing to integrate big data processing tool	10%	4	0.4
4	Failing to construct a PKI infrastructure	15%	2	0.3

Figure 3.4: Possible project risks

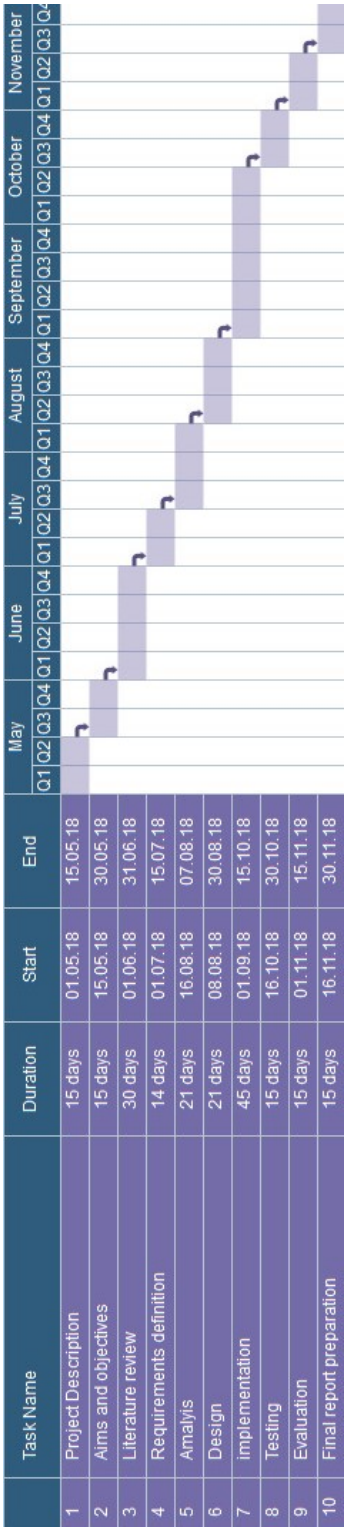


Figure 3.3: Gantt chart of the project planning

1. **The final system does not meet the requirements** This risk is the crucial one and it consist of all of the other risks identified. Therefore, it is mark with the highest effect, as not finding a way of mitigating this risk would result in failure of the project. For this reason, an in-depth research was conducted of many similar topics, and of many mechanisms and techniques which can be utilised in fulfilling the aim of the project. The fact that the IoT, big data and cryptography are highly popular topics in the IT community, a lot of support and suggestion are available. However, finding the proper one is a difficult task.

2. **Failing to provide access control mechanisms** This risk comes from the fact that the topic of access control was not familiar before the beginning of the project. Additionally, as the IoT network is highly complex it requires a more detailed and challenging control mechanisms. Furthermore, authentication and authorisation are not simple tasks, which requires well defined structure and plan of their integration. For this reason, research was done on the subject of access control, acknowledging and defining the different models and methods already existing in the community. As the requirements of the IoT are specific, finding a most suitable solution in acquiring the authorisation and authentication goals was crucial.

3. **Failing to integrate big data processing tool** This risk arises from the fact that these tools are intended to be implemented in embedded system, requiring specific installation methods. However, as many tools exist and they are installed by many individuals and groups, it was highly likeable that a proper way of utilising such a tool would be found. On the other side, failure of utilisation would lead to direct failure of couple objectives of the project.

4. **Failing to construct a PKI infrastructure** The risk of constructing such an infrastructure comes from the many components and mechanisms it requires in order the whole PKI to be established. Components and mechanisms like digital certificates, by understanding the process of hashing and private key signing, data encryption, switching the regular internet protocols to secure ones, etc. Therefore, wide research should be conducted in understanding these problems, but also finding a way for their implementation. The effect given is the lowest from all of the risks, as such a infrastructure already exists and can be directly used, but at a specific cost.

In the end, the conclusion is that risk management is phase of the project with a high importance, as it directly affects the completeness of the requirements, thus fulfilling the aims and goals of the project. As it can be concluded, in resolving each risk, a well conducted research is required, which will discover all the possible solutions and ways of solving the specific risks.

Chapter 4

Analysis

4.1 Analysis Process

The process of analysis encompasses the tasks that need to be taken in determining the the conditions that the project have to meet [54]. These tasks involve processes of analyzing, documenting and managing the system or software requirements. Requirements are the features that the system is composed of, with a purpose of fulfilling the main aim of the project. Therefore, this phase plays very important role in the final project result. Furthermore if the requirements and features to be achieved are mistaken, the objectives of the project would not be met, so it affects the final success of the project. The end product may be implemented on a high quality level, including technologies and algorithms never seen at that moment, however, in case the required features are not developed, the whole project would be marked as failure. The requirements, produced by the analysis phase, should not be written in a technical language, as they need to be understood also by individuals who are not part of the project, and do not have additional knowledge of the techniques and technologies used during the development. In addition the requirements and the features to be implemented should be testable and measurable giving a further guidance of the completeness and the point where the project development stands.

In order the requirements of the system to be defined , first and foremost, the main aims and objectives of the system should be considered. Evidently, the main requirement of the system is providing a service capable of storing and processing data generated by IoT devices, with an emphasis on the ability to manage enormous datasets, defined as big data. Additionally, the system should adopt a specific communication protocol, suitable for the restricted IoT network. Furthermore, protected data transmission and management should be one of the main characteristic of the whole system. As already defined in the literature review, many security goals exist, which have to be implemented in a security conscious projects. However, depending on the nature of the project, some of the goals need to be elaborated and implemented in details, while on the other side, some of them should not be considered as their breaching would not lead to any security issue.

One of the main security goals which have to be implemented is data confidentiality. It is of big importance, as different kind of data can be transmitted from the IoT devices, including sensitive and private data. Additionally, proper control of accessing permissions should be implemented, as the service can serve many devices. This indicates that authorisation mechanisms are required in the system. Providing a mechanisms for defining rules which will guide the access control, should be one of the core features of the system. Proper authentication is also needed, in order the access control to be fulfilled. This feature is important part of the system, as the devices need to know who they talk to, and vice versa. Evidently, next goal required is integrity, as it is the mechanism which will provide this identification of the devices and the service not to be changed at any point of the communication. Integrity mechanisms on the data transferred from the IoT devices were considered, however, as the data is constantly generated and transferred over high latency and low bandwidth networks, it is highly infeasible checking the validity of each message sent. This fact indicates that error tolerance of the messages should be considered.

The impressions from the analysis and the vision of the project, indicates that the system to be developed would be more inclined on the security aspects of it. Therefore, the proposed system should be designed in such a way, that it would be easily integrated in other projects of similar nature. Meaning that it should be able to enhance the security aspect of other architectures, developed for managing IoT devices, and analysis of the data they produce.

In the following sections, the functional and non-functional requirements of the system would be defined, explaining the main features of the developed software. Furthermore the manner in which all of the previously stated goals and aims would be implemented will be described.

4.2 Functional Requirements

How the system would behave is defined by the functional requirements it has. These requirements describe the system functionalities and the functionalities of the internal components. Therefore, these functionalities specify the desired features the system should have, performing the desired actions and tasks. The functional requirements determined by the detailed analysis conducted could be defined as the following:

- **Big data processing and storing**
  As this is the main aim of the project, it is obvious that corresponding tools should be implemented. However, these tools should enable storing and processing of the data on distributed system, as already defined in the literature review big amounts of data generated from IoT are impossible to be managed in a traditional database management system. Therefore, the system should enable utilising of different big data processing tools, capable of exploiting distributed systems and their property of parallel storing and processing. As there are many tools with the specified characteristic, which are suitable in

different scenarios, the project should not be developed around a specific one. Adoption of design patterns could support this property, described further in the design phase /refdesign. Anyway, even interchangeable one of the available tools should be implemented for testing and demonstration reasons.

- **Lightweight data transmission**
  The data collected from the IoT devices should be properly transmitted to the services responsible for further management of the data. A suitable protocol should be found answering the challenges of the IoT network, high latency, low bandwidth and also the limited capabilities of the devices requiring simpler initialisation of the communication. Therefore, lightweight communication protocol is required. Many IoT protocols exist, providing different solutions, thus choosing a proper one depends on the nature of the design decisions in the project.

- **Secure data transmission**
  The communication must be encrypted, in order to fulfill the confidentiality goal. Symmetric and public-key cryptography are the two possible solutions in enabling confidentiality. Both of them have essential characteristics for the IoT, the first, enables faster and more efficient encryption, while the later provides ease of including new devices in the network. Hence, providing an hybrid solution which will gain the advantage of the both solutions would be the best possible scenario. In addition the devices should know whether they communicate with the right service. Therefore methods of proving the identity of the service should be implemented, including mechanisms of keeping the integrity of these identifiers.

- **Unique identity of each device**
  The number of devices in the network can go up to millions of entities, therefore proper identification of each device is essential. The identification is also needed to distinguish the data produced by each device, as all of them does not have the same responsibilities in the environment.

- **Access control on devices actions**
  It was already mentioned in the analysis process, that authentication and authorisation are important security goals to be implemented. The authenticating should be provided on top of the devices identity, providing a mechanisms of creating, storing and validating the devices credentials. On the other hand, authorisation is more complex issue and the big number of devices are making it even harder to implement. However, proper way of dealing with this problem should be implemented, as different devices would store data on the services, and unauthorised accessing should be eliminated. Different ways and languages exist in dealing with access control, however the architectural needs would define which approach is more suitable.

- **Secure credentials storing**
  As the devices would be authorised using their credentials, securing this information is crucial for the security of the whole network. Stealing the devices

credentials would enable the attacker to access the services resources without any notice on its side, leading to a serious breach. Therefore, when storing the credentials they should be properly secured and protected in case of malicious and erroneous events.

4.3 Non-Functional Requirements

Opposite of the functional, non-functional requirements are used to evaluate the operation of the system, rather than its behaviours. These requirements are also known as the qualities of the system. So, in order to have full analysis of the system these operation evaluations are required. Although, many different non-functional requirements exist, the analysis of the project should consist the following:

- **Efficiency**
  The fact that the project is developed for the IoT, efficiency is one of the primary requirements for the limited constrained devices. All its aspects have to be optimized for efficiency, starting from the communication protocol, encryption mechanisms to the authorisation procedures.

- **Maintainability**
  It is the property of the system to be maintained and improved with ease. This is an important requirement because the product can be further developed and tweaked, requiring easy modification and adaptations. Additionally the data coming from the devices can be in different structure, leaving a place for mistakes and further adaptation of the storage entities. This fact makes the need of maintainable system even more important.

- **Availability**
  Availability is formally defined as the time the system is available. This requirement is described more mathematically, where the percentage of, the time the system is available as opposite of the time it is not responding, is calculated. Because the nature of the IoT is to stream its data, due to the small amount of memory, ideally the system should be available all of the time.

- **Extensibility**
  Extensible system design is the one which takes in consideration the future growth. It can be seen as a measure of the ability extensions to be further implemented. As the nature of the project is integrating different processing tools and different kind of IoT devices, this requirements should be added in the list of the most important ones.

- **Adaptability**
  As a characteristic, adaptability means the ability of a system to easily be integrated, thus adapted, in different environments. It was already mentioned, in the analysis process, that the system should be made in such a way to be integrated in other similar solutions. Thus, this requirement have to be taken in consideration when constructing the general architecture of the system. Therefore, following the microservice architecture pattern [55] would be

suitable in making the system more adaptable. The pattern, suggest creating couple of smaller components providing a different services, leading to decoupling between the components. This decompositions, enables the modules, components, to be more independent, enabling them to be used straightforwardly in different environments.

4.4 Theoretical Design

After defining the main requirements of the system, picture can be made of the initial architecture. Additionally, defining the initial architecture would give a clear guideline on its further detailed development . The fact that the analysis and design are closely related gives the opportunity of defining the initial design of the product. This initial architecture should answer on all of the defined requirements, functional and non-functional. Figure 4.1 illustrates the initial architecture, the description of the chosen approach would follow.

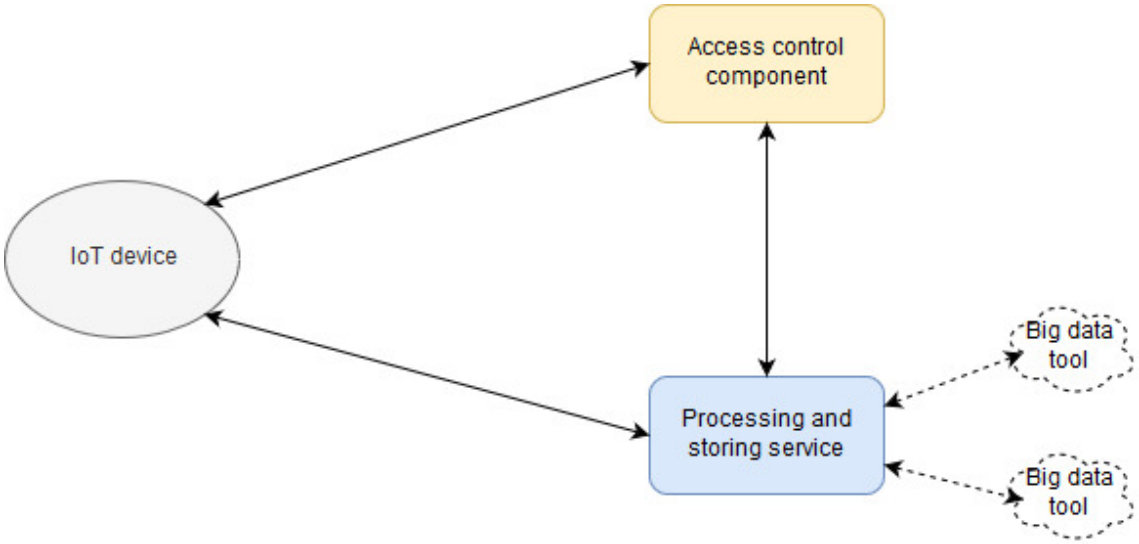


Figure 4.1: Initial architecture description

At first, when implementing the different features of the system, they have to be decoupled as much as possible, in order to fulfill the adaptability requirement, creating many smaller components. For this reason, the processing and storing services should be developed as different components of the access control ones. This approach would provide an easy and practical fashion of adding new services, but also implementing the access control mechanisms in other architectures, providing the same level of adaptability. It is also obvious from the illustration that the services should provide a suitable interface for integrating different processing tools, which will collaborate with the service in order to fulfil the storing and processing aims.

The communication between any of the components and the device should be encrypted, in order the protection to achieve the security goals. However from this point of view, the other requirements could not be illustrated, as they are more

related with the initial design of the components. It is worth mentioning, that this initial architecture is just a proxy interpretation, used for clarification for the reader and for the further development.

4.5 Summary

The analysis phase of the project mainly gives the technical specifications of the software to be developed. Therefore, this chapter forms the basis of the further detailed design and implementation. The main focus was on describing the functional and non-functional requirements that the system have to fulfill. These requirements, are highly related to the projects objectives, however, written in more technical language. Furthermore, the theoretical design of the system was illustrated, giving a broad view of the architecture. This view have a purpose of getting closer to the main design, thus giving an guidance of the end product which should be implemented.

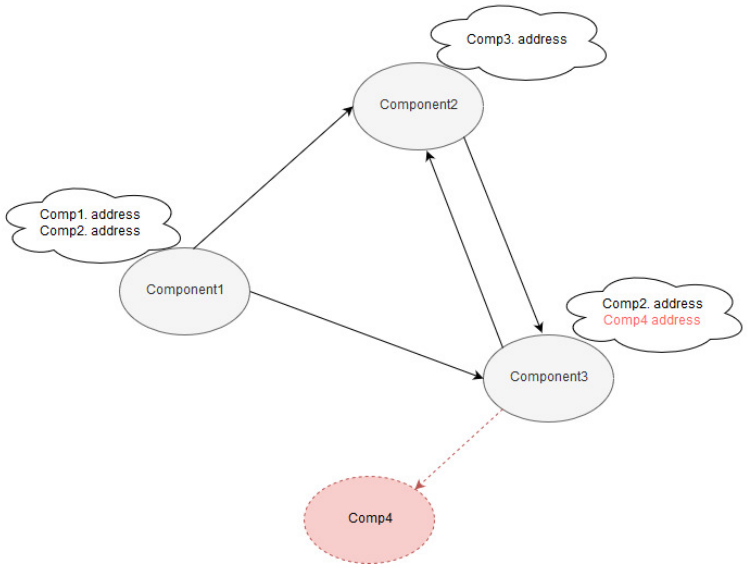
Chapter 5

Design

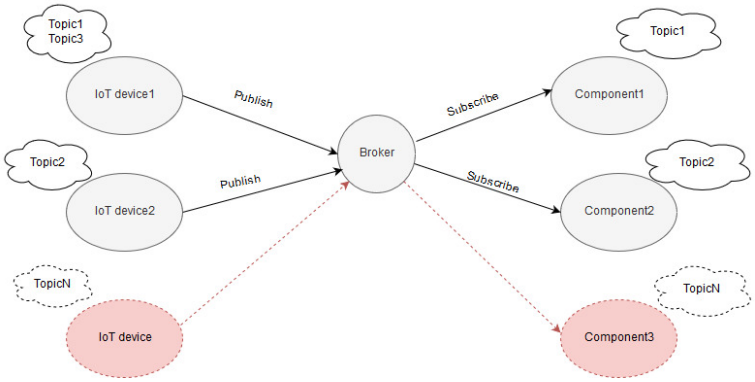
As the size and the complexity of the software systems increased over the years, the software architecture emerged as a discipline to describe and specify the overall system structure [56]. Although software architecture dates from the 1990's, there is not one widely accepted definition of the term [57], however, there can be found many similarities in the definitions of different researchers. Therefore, the general definition could be specified as, a blueprint, consisting of the structures of the system. The structure comprises of: software elements, externally visible properties of those elements and the relationships among them. The architecture gives more abstract and gross-grained view of a system, hiding the unnecessary details, making the system easier to understand and intellectually manageable. These abstractions are satisfying the non-functional requirements of the project like: reliability, maintainability, reusability, extensibility and efficiency. Furthermore, the architecture provides a framework for satisfying requirements and gives basis for cost estimation.

Employing a well defined software architecture brings many benefits in every project. First, it describes the system before it is implemented, giving a basis for further analysis and design decisions. Next, the components and structures are defined in a way that the code is reusable and easier to maintain, which saves further design change of requirement costs. The software architecture also influences the success or failure of the project, as it is described at the beginning and it is the basis for further development and communication. Every system is based on some architecture even if it is not explicitly designed, therefore defining it well is one of the crucial tasks. Successful architecture is the one that overcomes the difficulties and challenges of the project requirements. Variety of architectural styles exist and each one of them is suitable for a different problem. For this purpose two different styles were employed in the project, the explicit and implicit invocation. Fig 5.1. presents both of the chosen architectural styles, with scenarios how new devices and components can be introduced.

Explicit invocation is one of the most common architectures [58], where the components interact directly by invoking services defined in the other components. These services can be methods of objects, explicit queries of database, API calls, etc. While in this approach the "caller" component know the exact explicit reference



(a) Explicit invocation



(b) Implicit invocation

Figure 5.1: Explicit and implicit invocation:

In both cases the clouds above the communication entities describe the data that each entity needs to hold. New components introduced in the architecture are labeled in red color.

(address) of the other components, in implicit invocation it is completely the opposite, meaning that the "caller" can invoke services of a specific components without explicitly having their reference, address. The fact that these architectures are different they are implemented in different scenarios. Explicit invocation causes higher coupling between the components, however in some situations it would not harm the system but it would be desirable implementing it. Suitable scenario applicable for explicit invocation would be when the location of the components(e.g. network address) is known in the design phase of the project. In the project developed, whenever components which are not an IoT devices, communicate between each other, explicit invocation is chosen as a form of communication. This approach is preferable in the described scenario, as the specific components would not be added dynamically, in bigger numbers, so managing the addresses for communication can be conducted or static way, while initialising those components. Additionally, the

architecture should be constructed if it is able using explicit invocation, as the implicit way is harder to implement and manage.

On the other side, the implicit invocation is more applicable in scenarios where the components invoked are not interested in the invocation itself, rather they are interested about the invocation result [59]. These scenarios are seen in environments in which components can be added or removed dynamically without disturbing the communication in the system, thus, statically initializing their addresses for communication at the beginning of the design would be infeasible. This approach is also known as the publish/subscribe message pattern, in which the the "sender" of the message publishes its data into specific classes, also known as topics, without knowing the subscriber on the other side receiving that data. Furthermore, the subscriber, express an interest in one or more topics, without knowing the publishers on the other side. Intermediary entity, known as broker, is responsible for gathering the published messages and further directing them to the appropriate subscriber. In the project, the described communication pattern is adopted whenever an IoT device communicates with the other components in the architecture, which is suitable, as these devices can be dynamically integrated or withdrawn from the network.

Many IoT message protocols exist following the publish/subscribe pattern and choosing the right one depends of the requirements the architecture has. One of the most popular and widely used [60] are the following protocols: Mqtt, CoAP, DDS and XMPP which are all intended for the IoT, however one has to be chosen which would be the most suitable. XMPP is one of the oldest and thus most tested and supported protocol, however it is not suitable for devices with limited resources as it is not lightweight as the others, meaning that this protocol is not preferable in the project. The next one, DDS is another protocol with a large drawback, as it produces high system latencies and bandwidth, making it efficient comparing with the other protocols. In the end, the decision is between Mqtt and CoAP, which are both lightweight and efficient protocols, however the latter is primarily one-to-one protocol, meaning that adding more IoT devices and services on same topic is not suitable and feasible in this protocol. Therefore the Mqtt message protocol is decided to be used in the project, as it fulfills the communication requirements of the architecture. However the manner in which the components communicate between each other is only one property of the whole architecture, the characteristics and responsibilities of the components is another crucial property.

The aim of the the project, developing a service which enables processing and storing of data produced by IoT devices, sounds as a straightforward task. In theory, the IoT device, collects data produced by the environment surrounding it, that later is transferred to the services responsible for further processing and analysis. However, in order this theory model to be put in practice, the architecture has to resolve many challenges. Some of them originate from the limited resources of the IoT devices, thus finding a way of efficiently using them. Interoperability is another issue, coming from the large variety devices including smartwatches, connected cameras, smart appliances and many more, living together within the IoT. However, security

is the main problem that the IoT network needs to overcome, hence it should be resolved before implementing any processing tools and algorithms. Therefore, firstly the security segment of the architecture will be elaborated, as well as clear definition of why particular approaches were taken. After these issues are overcomed, the data transfer from the clients to the services responsible for processing and storing, will be described. Before the detailed description of the system, it is worth mentioning that the Intermediary broker would be omitted when illustrating the communication between the IoT devices and the other components in the architecture, for a simplicity reasons.

5.1 Data Transmission Protection

As mentioned, one of the main challenges, regarding developing a successful IoT network, is the security.The security treats come in many different forms, software attacks, identity theft, theft of intellectual property, etc., therefore, protection from such a variety of threats is wide and includes many disciplines. This project addresses the protection of the data transmission between the client and the service and regulating clients permissions on the data and computational power possessed by the services, access control.

First and foremost, in any system, transferring a data over the global network without any protection is insecure. As the global network can be accessed at any location by anyone connected to it, the data transferred can be easily intercepted and read. Therefore, this can be a big issue when transmitting the sensitive and private data gathered by the IoT. One approach can be constructing a personal network, containing only the IoT devices and the services, however, this solution is extremely costly and infeasible to develop. Therefore, the only practical solution would be using the existing global network, in such a way that, the data can be understood only by the parties that communicate. The aforementioned solution can be achieved by encrypting the communication. Additionally, protecting the data communication accomplishes the 6^{th} objective of the project defined in section 1.1

Earlier security protocols, suggested using the pre-shared key mechanism [61], in achieving data protection. In this mechanism, before any communication is established between the two sides the IoT device and the service, pre-shared secret need to be exchanged. So whenever they try to communicate, the messages are encrypted using the specific secret. This secret is a cryptographic key, usually symmetric as they are more efficient. Additionally, this schema provides an authentication to the system, as the server knows the identity of the holder of a specific shared secret. However, this approach has couple of disadvantages. First, the number of services that the IoT device, communicates with, is equal to the number of keys it has to store in it's memory, and as mentioned before, these devices have limited resources. Moreover, the service needs to hold as many keys as there are devices sending data to it, which can grow up to a million devices. Next, whenever new server is introduced to the network, each device has to be updated with a new shared secret,

leading to a problem of updating vast amount of devices, part of them with limited accessibility.

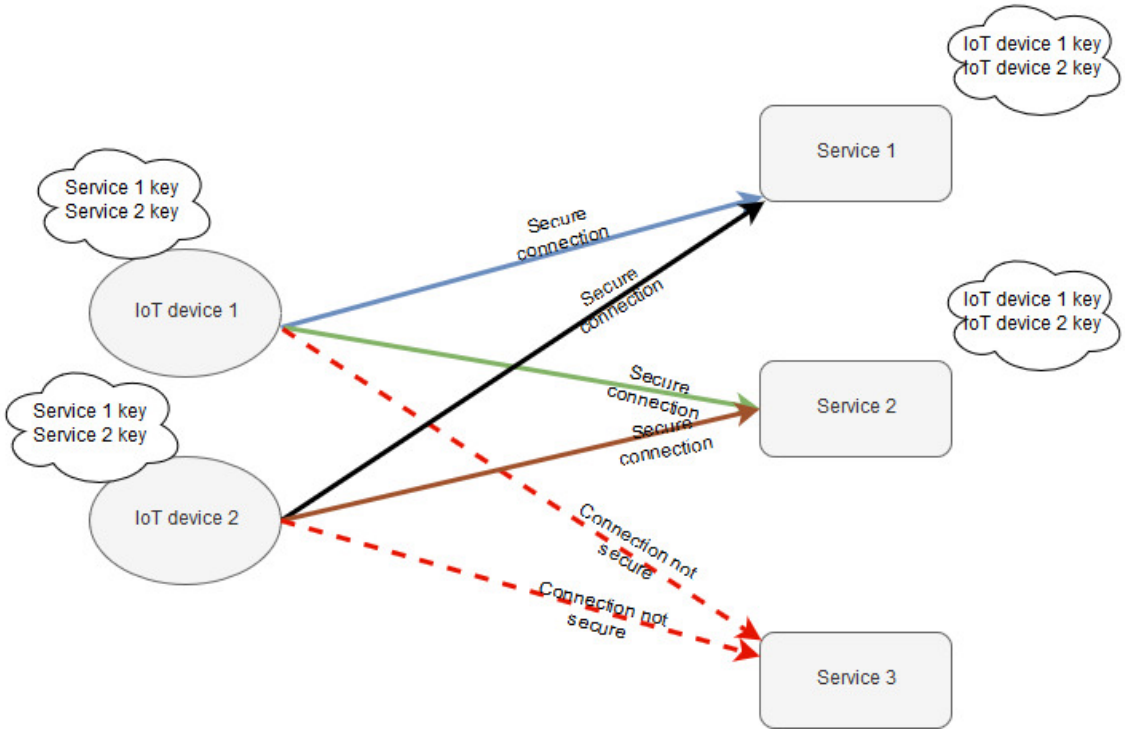


Figure 5.2: Pre-shared key architecture

The Iot device 1 and 2 can securely communicate with the Service 1 and 2 as they have stored pre-shared secret. On the other side communication with service 3 is not secure, therefore shared key have to imported on the IoT and service side

As we have seen, the pre-shared key mechanism opens up a new problems, it requires more resources on a devices with limited capabilities and it causes the network to be non-flexible and difficult to expand. After facing these issues, the community realised that incorporating certificate authority CA, would solve the problems [62]. This authority is a trusted entity which can give certificate to devices and services. So whenever the same trusted certificate is found on the both sides, the communication is trusted. First, the devices have integrated the certificate of the CA, upon their installation, and second, the services generate their own certificates which are later signed by the CA. The signing, creates a chain of certificates, with the CA certificate as standing at the root of it. At the beginning of the communication, the device validates if the service has the CA certificate in its chain. However, in the project any request for signing would be completed, without checking the validity and credentials of the services, which can lead to a breach of the security. Therefore, mechanisms deciding which signing requests would be served needs to be further implemented. With integrating the certificate authority, the network is easy to expand, as the clients would only have to hold the certificate of teh CA, and thus trusting every service which is signed by the authority. Additionally new services can be easily introduced in the system, only by acquiring a signed certificate

from the trusted authority. However, the proposed solution have one disadvantage because it requires a public-key infrastructure increasing the complexity and cost in comparing with the simple pre-shared architecture. Nevertheless, the advantages that it gives are greater and overshadow the drawbacks.

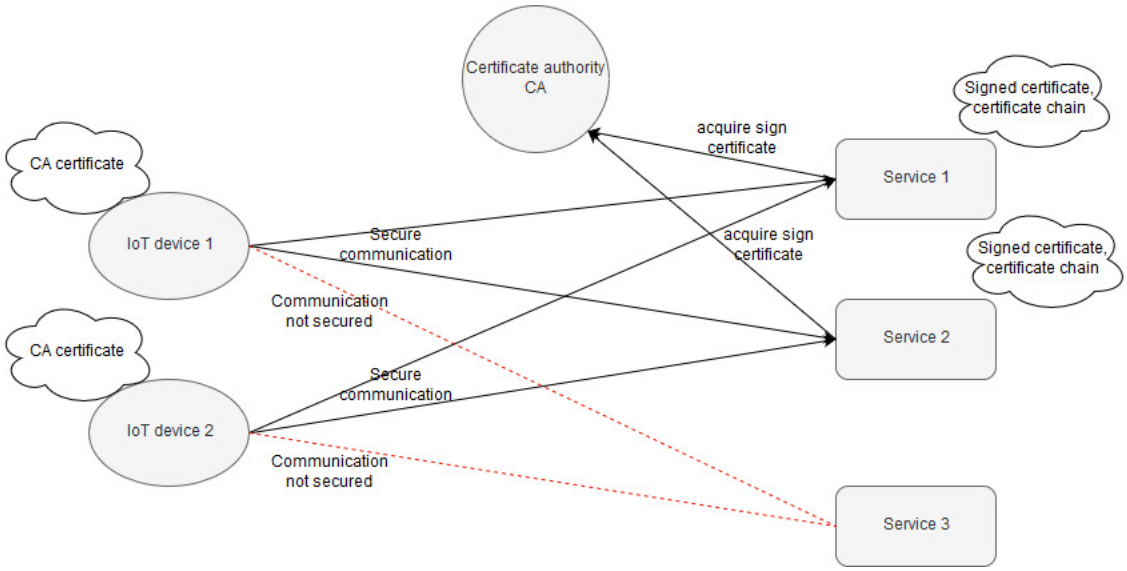


Figure 5.3: Certificate authority architecture

The Iot device, which posses the CA certificate, can communicate with any service having a signed certificate by the CA. Communication with service 1 and 2 is secure, however service 3 has to acquire its certificate to be signed, in order the communication to be secured

So far, the proposed mechanism only describes how trusted network can be constructed, without describing how the data will be protected during its transmission. For this reason, a specific network protocol, that enables encryption of the data have been implemented. Although many different protocols exist, Secure Socket Layer SSL [63] is one of the mostly used. It is responsible for the encryption of most of the websites and applications that run over TCP. The main advantage of SSL, that outgrew the other security protocols, is that it is application layer independent [64]. Meaning that each application running over TCP can also be SSL protected. The fact that the popular IoT message transmission protocols [60]: Mqtt, CoAP and XMPP run over TCP [65] [66] [67] was crucial in choosing SSL for the purpose of the project. When the connection starts, between the device and the service, SSL handshake occurs. This handshake requires the certificate from the service, in order to be authenticated by the device. If the certificate was signed by the trusted authority, the communication can continue. Certificates public key is used in order the handshake to be completed, thus creating a secure connection. During this handshake, shared secret, symmetric key is exchanged, which brings back the idea of pre-shared key but on a more sophisticated level. This way, the efficiency of the pre-shared key is kept, wrapping it around the public key infrastructure, giving the whole architecture more flexibility and ease of growing. Certificates in the project

are utilised according to the X.509 standards and format [68], that is why they are also known as X.509 certificates. These certificates are defined by the International Telecommunications Union's Standardization sector (ITU-T) [69], and are used in SSL and many other internet protocols.

Moreover, SSL supports different public-key cryptosystems (RSA, ECC, DSS etc.) to be used during the handshake procedure. This can be of great benefit, as specific cryptosystems are applicable to different conditions, such as the IoT environment which has limited resources and capabilities. Despite the fact that RSA is the most used asymmetric cryptosystem, the IoT devices can benefit more from Elliptic-curve cryptography ECC [70]. The ECC cryptosystem perform the encryption more efficient with smaller sized keys than the RSA. For example 160-bit ECC encryption provides the same security as a 1024-bit RSA key and can be up to 15 times faster, depending on the platform on which it is implemented. Therefore the advantages of ECC are particularly important in the constrained IoT devices. However, one of the disadvantages of ECC is that the size of the encrypted message are increased significantly more, compared to RSA [71]. Furthermore, the ECC algorithm is more complex and more difficult to implement. The fact that many libraries exist that are providing ECC encryption out of the box, and the fact that the size of the message does not cause significant drawback in the system, this cryptosystem was used in the project, for the purposes of the SSL handshake procedure.

5.2 Access Control

So far, it was described the manner in which the network has been protected from malicious intentions by accessing the information during the data transfer. However, there is still a big threat from unauthorised access on the services. Scenarios of devices operating on data, which are not authorised to do so, are highly probable to occur if there are not prevention mechanisms. For this reason access control is one of the most crucial security technique. Additionally, the fact that it is more likely the security to be compromised due to misconfiguration of the access control policies, resulting in unauthorized access, than from a failure of a cryptographic protocol [72], highlights out the importance of implementing a well defined access control systems. The access control provides a mechanisms for achieving the 7^{th} project objective, defined at the beginning of the project, section 1.1. The authorisation and authentication units would be able of providing only authorised devices to access the services resources.

In the task of finding the most suitable access control standard for the IoT, OAuth [73] emerged with a suitable solution for the specific problem. Although, the OAuth community have already implemented solutions for access control on websites, modified solution was further developed, for devices with limited capabilities, such as the IoT devices. The main idea behind their solution was dividing the authorisation unit from the resource owner, in the project the service responsible for storing and processing the data. However, OAuth by itself does not provide any

protocol for authentication. Therefore, authentication unit was developed and integrated separate from the authorisation unit, with communication protocols between them established. Two of the mentioned units would be further described in more details.

5.2.1 Authentication

The authentication unit is responsible for validating whether a specific IoT device is part of the secured network. The unit keeps a database filled with credentials, Id and password, of all of the authorised devices, so whenever a device needs to be validated, its credentials are checked if present in the database. This check is performed using an interface implemented at the units side. The interface accepts the devices id and password, after that, the credentials are checked whether they exist in the database, and in the end appropriate message is sent back. New devices can be introduced, in the network, by inserting their credentials in the authentication unit, using an interface built for that specific purpose. Obviously, this approach can lead to a serious security breach, as any device can be integrated in the network, just by simply adding their id and password. Therefore, the interface requires additional credentials with the ones send for authentication, which are only known by the individuals, users, responsible for introducing new IoT devices in the network. In order these individuals to be integrated in the system, one more interface exist, responsible for inserting new users of this type. Their credentials are stored in the database, and are validated in the same why as the devices credentials. For simplicity and security reasons, the insertion can be performed only by one user, knowing an specific credentials for this action.

The most simple approach in storing the credentials would be by inserting them in the database as they are, without changing their plaintext. However, as the number of records (passwords, credit card information, private health data, etc.) exposed from data breaches has come to almost 180 million records last year [74], it is clear that the passwords have to be somehow protected, in case the authentication unit is breached. The first and most intuitive approach would be encrypting the passwords, which seems as a great solution for the problem, as only the one holding the secret key, the authentication unit, is able to get the real value of the password. So whenever credentials of a device are received for authentication, the password stored in the database for that specific device is decrypted and compared if the same with the received password. But there exist a major issue in this method, the secret key is often stored in the same machine where the passwords are stored. So whenever the security is breached, the secret key can be obtained and therefore each password would be decrypted. The solution can be found in the hashing algorithms. Passwords are first hashed and then stored in the database, which makes infeasible the value of the password to be generated from its hashed form. using this mechanism, whenever credentials need to authenticated, the password are hashed and then checked in the database, whether such values exist. Although, the infeasibility of regenerating the password seems as enough, they can be easily cracked by fairly simple attacks like, look up tables and rainbow tables [75]. Therefore, the

passwords are hashed together with a random string, the procedure is also known as salted hash [28]. Storing of each credential, in the authentication unit is performed using this method.

5.2.2 Authorisation

As previously mentioned, unauthorised access to the services data and processing resources is one of the main security issues, therefore each device action needs to be authorised for permission. The authorisation was implemented following the OAuth design. In their architecture, the authorisation of the devices actions and the services for processing and storing are performed on different units. This separation of the responsibilities, gives a possibility of new services to be introduced in the network without defining all authorisation mechanism on them. In order the whole authorisation process to be better understood,first it would be briefly described and afterwards the whole procedure would be illustrated in more details.

At the beginning, before any action on the services is taken, the device has to authorise its intentions, by sending its actions to the authorisation unit. At this point the unit does not known whether the device is part of the secure network, therefore the device has to additionally send his id and password. Request is sent to the authentication unit, containing these credentials in order to be checked whether they belong to a authenticated device. In case they are, the next step, the authorisation unit has to take, is deciding whether the device is permitted or denied in performing the specific actions they require. These decisions are taken following previously described policies, in which strict rules are defined regulating which actions a specific device can perform. In the end the services need to be informed whether a device is authenticated and its actions are authorised to be performed. OAuth defines two approaches in resolving this issue. In both of them, a special string called token is generated by the authorisation unit, containing the devices actions along with their permission, and sent back to the device. Thus, each time it communicates with the server, the token is shown as a proof that the actions in it are permitted. However there is one major differences between the both approaches. The first one [76], suggest the validity of the token and thus the actions of the device, to be checked at the authorisation unit that generated it. In this approach all of the tokens have to be stored and their retrieval to be managed by the unit, and additionally an interface for the specific validation has to be provided. The second approach [77], which was followed in the project, is using a methodology called proof of possession [78], in which the token is generated in such a way that any device in possession of the token can perform the actions defined in it, without any further communication with the unit responsible for its generation. This approach, compared to the first mentioned, brings couple of benefits to the architecture:

- decoupling the whole architecture

- decreasing the network bandwidth and the time needed the service to validate the token, as it can be done without communicating the authentication unit
- Consequently, the tokens does not have to be stored and their retrieval managed at the authentication unit

With the big picture of the authorisation process described, a more detailed explanation can be further provided. As described, at the beginning the device has to send its credentials and actions to the authorisation units. Because the devices communicate through the publish/subscribe protocol, the information have to be published on a specific topic, on which the authorisation units are subscribed. Therefore, before the device is installed in the network, this topic have to be stored in its memory. Further on, the credentials are authenticated through the authentication unit as mentioned and the next step is authorising devices actions.

Choosing an appropriate access control model is a key factors for a successful authorisation. One of the oldest and widely used models are discretionary access control (DAC) and mandatory access control (MAC), which are the basis of the modern models [79]. In DAC, the devices have complete authority over the data which they generate, also determining the permissions for other users. This approach has some drawbacks regarding the architecture described in the project. First of all, the devices have complete power over the authorisation, opposite of the desired approach in which centralised unit is responsible for such actions. Another issue, emerging from this authorisation method is that it is difficult to ensure consistency in the access control policies, as each device defines its own. In the end, the IoT devices have limited resources, in order to apply the authorisation decisions. MAC contrast with DAC as the authorisation is controlled by a centralised unit, in which the policies are strictly enforced by a security kernel. This means that the devices do not have any control over the authorisation, which is suitable for the architecture. However, MAC is mostly used in a systems where the priority is based on confidentiality, which makes it difficult the rules to be dynamically altered and it requires predetermined planning to be implemented efficiently.

Considering the drawbacks from MAC and DAC, an access control mechanism is needed which will be controlled centrally as well as being flexible in defining the rules and policies [79]. Role-based access control (RBAC) is a model which overcomes these difficulties. In RBAC [80] various roles are defined and each role is able to perform certain actions. The roles can be assigned to the IoT devices in the network, and each role will specify actions that the device can perform. Some of them can only store their data on the services, others can process or retrieve the data, and many combinations of these actions can be made. However situations can appear in which some data should be restricted of access during a specific period of time or specific information must not be retrieved by devices or stored on services outside of EU borders. Therefore, the most suitable model for the project is the attribute based access control (ABAC) model which evolves from RBAC [81], providing more

flexibility, as it is not only based on user attributes (roles), but also the system to be accessed and the current environmental conditions (time of access).

After choosing the right access control model a proper policy language and architecture, which will evaluate the devices actions, needs to be integrated in the project. Already implemented and tested language have been the main priority, instead of building a whole policy enforcing architecture from scratch, as it is highly complex and difficult requiring an expert knowledge and time to be build. At the beginning, when ABAC emerged as a access control model several proposals have been introduced in enforcing it [82] [29]. Their main focus have been the attribute certificates, which does contain a public key but, attributes that specify access control information associated with the device, the certificate holder (role, ip address, time of access, etc.). However, this approach is based on logic languages, which are not simple and easy to integrate, also they are not human-readable and easy to inspect, thus missing one of the requirements that an access control language need to have. Therefore XML-based languages seem to be more suitable for this context. One of the most relevant access control language using XML is eXtensible Access Control Markup Language (XACML) [34]. It is a language that expresses authorisation policies in XML against objects that are also identified in XML. With its clean and unambiguous semantics, the language is easy to understand giving him a big advantage in choosing it for authorisation purposes. Additionally many tested implementations in different programming languages exist, which can be easily integrated.

In the project, the authorisation unit have integrated XACML engine, responsible for defining access control policies and enforcing them against specific device actions. The policies are containing the following attributes:

- Role: each device has a specific role which are held in the authentication unit. The roles are defined upon their integration in the network. Each of it combined with the other attributes have specific permissions.
- Action: the actions that can be defined in the system are storing, processing, and retrieving. Each role can have different permission for a specific action. For example: role "A" have storing permission only, role "B" have processing and storing etc.
- Environment: the environment provides a different type of attributes as date or region. These attributes are not predefined like the action attributes and can be defined according to the needs of the policy.
- Object: a specific data defined in the object attribute can be permitted or denied of access.

The unit provides an interface for defining the policies to be enforced. As the XACML policies contain a bit more information in order to be easily understand, the interface accepts specifically described JSON format which is simplified version

of the XML policy. After receiving the JSON policy, the unit converts it in XML format, adding the parts that were missing in the simplified version, as these parts can be auto generated and are needed later by the engine. During the generation of a specific policy, the unit additionally generates a XML request template, which is used as an template of how the devices should send their action permission requests. This leads us to the definition of the interface responsible for accepting devices action requests. It accepts a filled request template with the devices id, action the device wants to perform, and additional environment and object attributes if required. For example one request would contain the device id, with processing purpose which would be performed at 10 pm on a hadoop cluster. As the authorisation unit accepts roles instead of device ids, for a simplicity reasons, on authenticating the device it receives also the role of the specific device. This way the authorisation policies does not have to hold all of the device ids only their roles. Afte the role is obtained and placed instead of the devices id in the permission request, the next step is defining whether the specific actions are authorised. This procedure is conducted by the policy engine, which iterates the request through the policies reaching a decision for permitting or denying the action. The structure of the XACML policies and requests would be further defined in details in the implementation chapter.

In case the device is permitted in performing its desired actions, authorisation is granted in a form of token. The token contains the actions of the device, the identifier of the unit authorising the specific actions and expiration time. In the end the token is signed with the private key of the authorisation unit, which ensures the integrity of the token and proofs that the token was generated by that specific unit. The token is delivered back to the IoT device which now can use it in performing its actions on the services without further communication with the authorisation unit. In the project JSON Web Token (JWT) [83] was used, developed by the Internet Engineering Task Force (IETF). The specific token was chosen as it is suggested by the OAuth community as suitable for their architecture, but also many libraries in different languages exist for generating, signing and verifying JWT tokens. Figure Auth. illustrates the whole access control procedure, starting from the device sending its credentials and actions, the units authenticating it and authorising its actions, in the end sending a token as a proof of the granted access control. The figure 5.4 illustrates a best case scenario, in which the credentials and actions are valid and pass all of the access control mechanism.

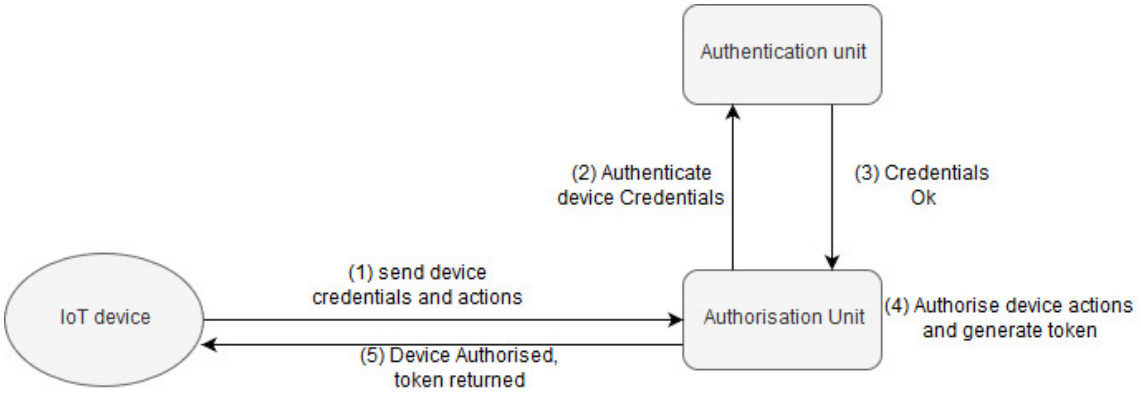


Figure 5.4: Illustration of the process of authorising device in performing its actions.

The specified token, which is used to authorise the devices actions without any further validation at the authorisation unit, is known as bearer token. Its name comes from the property that any holder ("bearer") of the token can perform the actions defined in it, even an unauthorised devices. Therefore, couple of security challenges exist in protecting the token being possessed by devices which were not intended to have it. First of all, its transferring have to be conducted throughout secure channels, property which is ensured in the project with the usage of SSL network protocol. As the communication between the device and the other units is encrypted, anyone else trying to tamper the communication would not be able to understand any of the messages between them. Also, the devices do not have any interface providing an access to their tokens by any other entity. In the end, the devices has to be protected from intents of gaining control and access on their resources, as the token can be easily stolen from their memory. Additionally, each token has an expiration time, after which it becomes invalid, thus a good security strategy is generating short lived tokens providing additional protections in situations when the unauthorized bearer tries to go behind the access control.

5.3 Data Processing and Storing

So far, everything described was concerning the security aspects of the architecture, providing the basis of protected transmission, processing and storing of the data generated by the IoT devices. Therefore, the protocols and procedures in which the devices and the services communicate between each other would be described, with purpose of storing and processing the collected data.

First of all, as the publish/subscribe is used as a messaging pattern for communication between the two sides, the device has to know the topic on which its data would be processed and/or stored. In the project, for simplicity reasons, only one topic was used on which one service was subscribed. However, it is of big importance mechanism to be provided, which would notify the devices whenever a new topic is introduced. This is an essential property, as services with different characteristic like, different processing tools (hadoop, apache spark, hive...), different physical locations, Ip addresses etc. can be introduced in the network, subscribing on specific

topics depending on the characteristics they have. For example, there can be a service which would process the data using hadoop and another one preferring apache spark. These two services has to be subscribed on different topics and in the same time the devices has to know which topic to be used for a specific processing tool. Further on, a protocol has to be defined, describing the procedure for transferring the data from the devices to the services. In the first step of the protocol, before any data is sent for processing/storing, a metadata is sent to the service, holding information necessary for the further actions that will be taken over the data. The metadata has a strictly defined structure in order to accomplish the 5^{th} requirement of the project, defined in section 1.1. The format of the structure is the following:

- The generated token from the authorisation unit, needed for determining devices actions and proving that they are authorised
- The tool to be used in case the data have to be processed
- The algorithms which have to be applied for the processing of the data
- Unique identifier of the action that would be performed, depending on the devices Id and the specific action. The action Id would be further used when sending data for that specific action, essential in scenarios when one device performs multiple actions on the same service
- The topic on which the device is subscribed, which is used for receiving information back from the services

After receiving the token at the service side, couple of processes are initiated. First and foremost, the service has to check the validity of the token. It starts with ensuring that it is not expired, followed by the phase of verifying the origin of the token. This verification is done by decrypting the signature in the token, with the public key of the authorisation unit responsible for its generation. For this reason the service obtains the public key of the specific unit from the certificate authority.The CA provides an interface for retrieving the public keys from each unit which is part of the secured network by providing the units unique identifier. The services use the interface by providing the identifier of the authorisation unit, stored in the token created by it. In the end, after acquiring the public key, the signature can be verified and with that the correctness of the token. In order the connection with the CA to be escaped everytime a token needs to be validated, the units public keys along with the corresponding identifiers are stored in the services memory. Therefore, the public key is first checked if already received by the CA and in cases it is not the specified procedure is followed.

Next, after the validation of the token, specific process is started on the service, named task, which is responsible for each specific action the devices want to perform. The task is identified by the action id defined in the token. It is also responsible for checking which tool have to be used in processing the data, and translating the processing algorithms from the token. By translation it is meant compiling the algorithm classes and keeping them in a specific location to be further executed when

needed. Additionally a specific locations are defined for storing the received data and for the output of its processing. Each task has it own file hierarchy containing these output and storing locations together with the algorithms classes. Finally, after the task is initiated, confirmation message is sent back to the device, on the topic defined in the token. The subscription requires additional resources of the device, therefore as soon as the device receives the confirmation it unsubscribes itself from the specific topic.

The confirmation message means that the device is permitted to send its data to the services. However, the data is accompanied by additional information, the action Id and the authorisation token. The first is required by the service in order the appropriate task to be assigned with the data and the later is used in order to check whether the device is authorised. As the data can be streamed, not only sent in a one big package, the service has to check the validity of the token each time, which can be costly as it requires decrypting the signature on each occasion. In order this process to be more efficient, the service keeps all the validated tokens in its memory, checking them every time a data from the device is received. In the end, the device sends a special character, indicating that no more data would be sent. In case the data needs to be only stored, no further action is taken at the service, on the other side, if some processing has to be performed, the algorithms defined in the specific task are executed over the data, outputting the information in the specific location of the task file hierarchy. Figure 5.5 Illustrates the communication protocol described between the device and the service, the second step is performed only in situation when the public key of the authorisation unit is have not been already obtained and therefore stored in the services memory. As it can be seen, the communication enables processing storing and retrieving of the data, as required in the 2^{nd} project objective, defined in section 1.1.

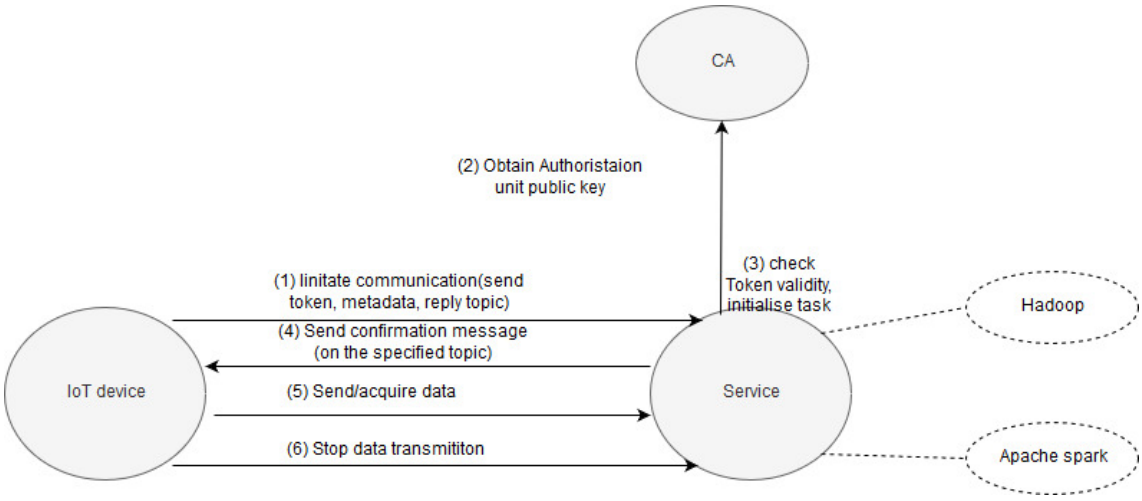


Figure 5.5: Data processing and storing protocol.

5.4 Design Patterns

Design pattern define a general solution for a common problem which repeatedly occurs in different software designs. It presents a blueprint that can be used and customised in solving a particular design problem. Before implementing a design pattern, the nature of the design problem have to understood, as well as the the fashion in which the patterns work, in order appropriate pattern to be implemented. The solution of the problem is defined through a suitable arrangement of the classes that are of interest and their ways of communication. As the patterns are more general, in order to be implemented in a particular problem customisation have to be made [84].

Despite the main purpose of the patterns, solving a particular design problem, by implementing them three nonfunctional requirements are satisfied. First, they are the coupling between the classes in the code and provide higher cohesion. Second, they enhance the documentation of the software design, thus increasing the level of maintainability and the ease of correcting and improving the solution. Finally, it facilitates the ease of change, modifications of the classes and how it affects other classes. For example, it is possible some of the classes to be modified while the others will remain unaffected [85].

Large number of design patterns exist which could be employed, therefore the decision which ones should be followed depends on the specific problem the design has. In the project three design patterns were employed.

From the architecture point of view, two common architectural patterns were utilised: the explicit invocation and the publish/subscribe patterns. Although their nature is completely opposite, in the first one the caller explicitly knows the receivers in contrast to the latter in which the caller does not know the receivers of the message, both of them were essential in overcoming specific design problems. The explicit invocation introduces high coupling if the components grow in numbers, however in a scenario when the specified components would not be dynamically integrated the preferred pattern is easier to be implemented and more efficient. On the other side in situations where components tend to grow in numbers, the publish/subscribe is more suitable as it decouples the design while enforcing the growth.

Strategy pattern is suitable in scenarios when there exist a few classes which are differing from each other only by their behavior [86]. Hence, there is a need those different behaviors of the classes to be separated, and it should be provided a possibility for the algorithms, identifying a specific behavior, be chosen at runtime. The structure of this pattern consist of a common interface for all various algorithms, through which each specific class is able to demonstrate its behavior. The other components are able to communicate with the desired algorithm through the interface. The communication is based on requests sent to the common interface which afterwards is delegated to the specific algorithm class. In the project, the strategy

pattern was implemented in situations when the data had to be processed with a specific tool, as the devices are able to choose it according their needs. This means that as many different tools are served on a specific service, many different processing algorithms and procedures have to be defined. Therefore one basic class, the task class, is implemented with different strategies for processing the data and compiling the received processing algorithm classes, which implementation depends on the tool that to be used. This way, each task class would keep the same structure of the file hierarchies for output and input along with the task identification and other properties, while providing different processing strategies essential for different tools. Additionally the strategy design pattern contributes in achieving the 3^{rd} and the 4^{th} project objectives, defined in section 1.1, by enabling different processing tools and processing models to be easily integrated and used.

Another design pattern employed in the developed system is the builder design pattern [87]. This pattern defines a way of constructing a complex objects step by step. Scenarios exist, where complex objects require laborious step by step initialisation of many fields and nested objects, which in the end would be buried inside a constructors with lots of parameters, or even worse scattered all over the code. This pattern suggest extracting the object construction code out of the class moving it to a separate object called builder. Despite the benefit of constructing the objects on more clear and neat way, by replacing the long and overloaded constructors, this pattern allows the object to be constructed without all of the parameters, which sometimes are not needed. In the project, builder was used when generating the tokens at the authorisation units and the tasks at the service as many parameters had to be set on their construction.

Furthermore, adapter pattern was utilised in couple of occasions for the needs of the developed system [88]. This pattern allows objects with incompatible interfaces to collaborate, by converting the calls sent by one object into a format that the other object understands. In other words it allows incompatible classes, which should not be modified, to collaborate with each other, with adding an interface between them providing the required compatibility. The pattern was implemented in the project during the definition of the access control policies. The authorisation unit provides an interface for declaring the policy in a JSON format, later adapted (translated) in XML format, which is required by the policy engine. Even if the interface requires an XML format, it would still need an adaptation as additional information has to be added to the policy.

5.5 Summary

Chapter 5 focuses on the describing the system design. It starts with the system architecture and architectural styles and patterns. The chapter argues that two architectural styles, explicit and implicit invocation, opposite in their main features are most appropriate to be employed. Furthermore, the design of the whole system may be divided in three different perspectives: data transmission protection, access

control and data processing and storing, which are described in details with additional statements why the specific approach and components were used. Finally, a succinct description about the employed design patterns was provided.

Chapter 6

Implementation

The complete documentation of the design provides appropriate instructions which will instruct the implementation of the system. This chapter focuses on the main details of the developed system. First there are described the development tools and technologies employed. After which the most important parts of the source code are described with detail explanation for them demonstrated.

6.1 Development Tools and Technologies

The whole system, containing all of the devices, authorisation unit, services and other components, was implemented using the Java programming language. Several reasons exist why this particular choice was made. At first, the design of the system requires an object oriented programming paradigm. Further, as there are many components in the whole architecture, there is a need of a programming language that is highly platform independent. Additionally, the mostly used language for IoT development according to eclipse foundation is java [89]. Furthemore, the language performances is of big benefit along with its popularity, providing a support for many different devices. The large community is also of a big importance, as it provides guidance and services for any issue that occurs. In the project, without the help of the open source java libraries for: cryptography, Bouncy Castle [90], the XACML implementation, WSO2 balana [91] and the mqtt broker and client, mosquitto [92], and paho [93], it would be infeasible in developing such a solution as the libraries mentioned require expert knowledge and time in developing them, providing sophisticated out of the box tools for the community. All of the mentioned benefits and characteristics makes the java programming language most suitable for the needs of the project.

Android devices were employed as an IoT devices, as they have many sensors capable of gathering different data and can be developed using java. Android studio development environment [94] was used for building the android solution, as it is an official IDE for android and most widely used. Furthermore, for the needs of developing the other components (Certificate Authority, Authorisation unit, Services...) Eclipse Integrated Development Environment was used [95]. This environment was

utilised as it provides a large plug in library including all of the mentioned essential libraries.

Furthermore, two file formats were adopted in the project, JavaScript Object Notation (JSON) [96] and Extensible Markup Language (XML) [97], both of them human and machine readable, and structured to be simply manipulated and worked with. However, the first was needed in generating the tokens and sending different requests on the network components, while the latter forms the basis for the access control policy language XACML.

When it comes to storing larger amounts of data, it was managed using different tools, depending of the scale of the data. For the purposes of the Authentication unit, for storing the passwords and roles of the devices, and of the CA, for storing the public keys of the other units, MySQL [98] database management system was used. This decision was made as the data to be stored is highly structured, ideal for SQL database systems, additionally it provides many security mechanisms in protecting the data and the access to it and in the end it is the most widely used for java backends. However, for storing and processing the abundant amounts of data generated by the IoT devices Apache Hadoop [99] was utilised.

In the end it is worth mentioning that the certificates and its private and public keys were generated using the java keytool command lines through the windows shell [100], the windows command line interface.

6.2 Data Transmittion Protection

This section will present a detail preview of the implementation of the secured communication between the entities in the network. At first, it will be demonstrated the procedure of generating a certificate. Next, it will be shown the signing of the certificates, by the centralised authority, and in the end an example will be presented of a certificate before signing and after the procedure, which will create a certificate chain establishing a chain of trust.

As described in the design, in order the communication between the components (authorisation and authentication unit, processing services etc.) themselves and with the IoT devices to be secured, SSL network protocol have to be employed, which requires usage of X.509 certificates. Java.security library provides classes for easy creation of the specific certificates. One such class is `CertAndKeyGen`, which is able to generate certificate along with public/private key pairs. Different properties can be set, starting from the encryption algorithm, in this case ECDSA, signature algorithm (SHA256), the key size (256bit), the component/issuer where it is generated (CN field in the certificate). The certificate creation is shown in the code listing given below 6.1. However it is not only needed to generate an certificate, it also have to be signed by the certificate authority. That procedure is shown in the last two lines of the code snippet. First the certificate is sent to the CA through the

method `signCertificate` of the class `CAApi`, which communicates with the Api of the CA, returning a certificate chain, containing the signed certificate and the certificate of the CA, which are later imported in the keystore of the unit through the `importCert` method of the `ImportCertificateToKeystore` class. However when the CA generates its own certificate, the last procedure is omitted as it does not have to sign its own certificate. The keystore is a database, provided by the java security library, for storing the certificate and the public and private key binded to them. Additionally, whenever an SSL connection is established, the certificates checked during the handshake procedure are taken from the keystore.

```
1  private static X509Certificate certificate;
2  private static KeyPair keyPair;
3  CertAndKeyGen keyGen=new CertAndKeyGen("ECDSA","
   SHA256withECDSA",null);
4  keyGen.generate(256);
5  PrivateKey privateKey=keyGen.getPrivateKey();
6
7  X509Certificate certificate = keyGen.getSelfCertificate(
   new X500Name("CN=" + GenerateKeystore.ALIAS), (long) 365 *
    24 * 60 * 60);
8
9  ArrayList<X509Certificate> certificateList = CAApi.
   signCertificate(certificate);
10 ImportCertificateToKeystore.importCert(certificateList,
   privateKey);
```

Listing 6.1: Generating a X.509 certificate and importing it in the keystore

However, before any entity to be able to communicate with the certificate authority, it is needed the CA certificate to be present in their memory. Therefore on installation of any component or device, that specific certificate is imported in their keystore. The code snippet below 6.2, illustrates the importing in the authorisation and authentication unit, and the processing service. The importing is performed using the windows shell accessible by the `Runtime` class. On the other hand the importing at the IoT devices is illustrated in the code snippet 6.3. In both cases it can be noticed that the keystores are accessed using a password, providing a restriction who is able to add certificates which to be trusted. For simplicity reasons in the project the password is kept in the memory of the application. however a more secure approach would be manually inputing the credentials when a certificate have to be added in the keystore.

```
1  String cmdStr = "keytool -import -alias
   caservercertificate -file " + caCertificateLocation + " -
   keystore " + trustStoreLocation + " -storepass " +
   GenerateKeystore.getPassword() + " -noprompt";
2
3  Runtime.getRuntime().exec(cmdStr);
```

Listing 6.2: Importing CA certificate in the keystore

```
InputStream trustStoresIs = context.getResources().
openRawResource(R.raw.androidkeystore);

String trustStoreType = KeyStore.getDefaultType();
KeyStore trustStore = null;
trustStore = KeyStore.getInstance("BKS");
trustStore.load(trustStoresIs, password.toCharArray());
String tmfAlgorithm = TrustManagerFactory.
getDefaultAlgorithm();
TrustManagerFactory tmf = TrustManagerFactory.getInstance(
tmfAlgorithm);
tmf.init(trustStore);
```

Listing 6.3: Importing CA certificate in the keystore of the IoT deivces

So far it was described that the certificates of the components in the network are signed by the CA but the procedure was not illustrated. It is shown on snippet 6.4. For the signing purpose it is needed three items: the certificate to be signed, `certificate`, the certificate of the CA, `issuerCertificate`, and its private key, `issuerPrivateKey`. The second is required in order to be known which is the identity of the signer and which signing algorithm to be used, while the private key is needed for the signing itself. The signing is processed using the `X509CertImpl` class, also part of the java security library.

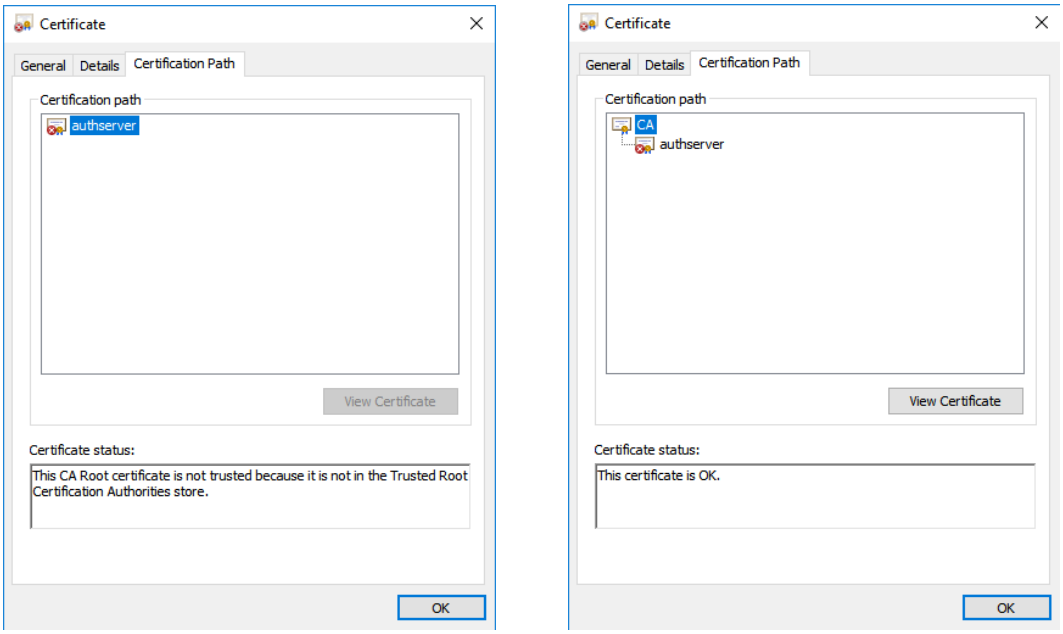
```
X509Certificate certificate;
X509Certificate issuerCertificate;
PrivateKey issuerPrivateKey;
.
.
.
Principal issuer = issuerCertificate.getSubjectDN();
String issuerSigAlg = issuerCertificate.getSigAlgName();

byte[] inCertBytes = certificate.getTBSCertificate();
X509CertInfo info = new X509CertInfo(inCertBytes);
info.set(X509CertInfo.ISSUER, (X500Name) issuer);

X509CertImpl outCert = new X509CertImpl(info);
outCert.sign(issuerPrivateKey, issuerSigAlg);
```

Listing 6.4: Signing certificate

A comparison between signed and non-signed certificate of the authorisation unit is shown on figure 6.1. As it can be noticed in the illustration, the signed certificate is part of a trust chain, with the CA certificate on top of it. This way every entity having the CA certificate is able to communicate with the unit on a secure way.



(a) Non-signed certificate (b) Signed certificate

Figure 6.1: Two certificates:
the left one is non-signed, while the right one is signed, forming a chain of trust

6.3 Access Control

This section will illustrate the implementation of the authorisation and authentication units in details, and the procedure of granting an access control to the devices. First, it will be described the manner in which the credentials of the devices are stored safely. Afterwards, it would be illustrated the structure of one XACML policy, how it is created and the structure of a authorisation request, in XACML format, generated when a specific policy is created. Next, it will be shown the interface on the authorisation unit, responsible for receiving access control requests, and the structure of the data it accepts. In the end, the engine granting a permissions or denials, of the requests, would be demonstrated, along with the token generated on a action permit.

Password credentials are stored in the database of the authentication unit in a secure way, using the mechanism of salting the passwords. Because, it is not safe to store the plaintext of the passwords they are hashed with a random string called salt, as illustrated in the design chapter. The method `hashPassword`, illustrated in the listing 6.5, is responsible for conducting the specific hashing. The parameters of the method are the password to be hashed and the random string, salt. These two are hashed using the `MessageDigest` class, which is part of the java security class. MD5 hashing algorithm is chosen, however the class is flexible providing many other choices. After the password is hashed, it is transformed in the traditional hexadecimal format and casted to a string format, to be more easily transferred and stored. In the end, the hashed password and the specific salt are stored in

the database. This same method is used when a new device credentials are to be added in the authentication unit and in a scenario when a specific credentials have to checked if valid. In the second scenario, devices credentials are sent to the unit, the salt stored in the database for the specific credential is retrieved and later hashed together with the password from the sent credentials. If the currently hashed password is equal to the one stored in the database, the credentials are valid meaning that the device have been authenticated.

```
1  private static String hashPassword(String password, byte[]
    salt) {
2     String hashedPassword = null;
3     MessageDigest md = MessageDigest.getInstance("MD5");
4     md.update(salt);
5     byte[] bytes = md.digest(password.getBytes());
6     //This bytes[] has hashed password in decimal format;
7     //Convert it to hexadecimal format
8     StringBuilder sb = new StringBuilder();
9     for(int i=0; i< bytes.length ;i++)
10    {
11       sb.append(Integer.toString((bytes[i] & 0xff)+0x100
   ,16).substring(1));
12    }
13    hashedPassword = sb.toString();
14    return hashedPassword;
15 }
```

Listing 6.5: Salted hashing

The policies that are enforced on authorising the devices have to be first defined. For that purpose the authorisation unit provides an interface accepting policies and importing them in the XACML engine. However, as it can be seen in listing 6.6, the structure is large and a bit confusing. Therefore the interface accepts a simplified JSON format of the policy, illustrated in listing 6.7, which later is parsed in the required XML format, filling the missing structural elements. The illustrations show the same policy just in the two different formats explained, and as it can be seen the JSON format is smaller in size and easier to read and understand.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd
   -17" PolicyId="StorePolicy"
3  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-
   combining-algorithm:permit-overrides" Version="1.0"><
   Target>
4  <AnyOf>
5  <AllOf>
6  <Match MatchId="urn:oasis:names:tc:xacml:1.0
   :function:string-equal">
7  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
   #string">store</AttributeValue>
```

```
8     <AttributeDesignator Category="urn:oasis:names:tc:xacml:3
      .0:attribute-category:action" AttributeId= "actionId"
      DataType="http://www.w3.org/2001/XMLSchema#string"
      MustBePresent="false"/>
9     </Match>
10    </AllOf>
11    </AnyOf>
12    </Target>
13    <Rule Effect="Permit" RuleId="A-can-Store">
14    <Target>
15    <AnyOf>
16    <AllOf>
17    <Match MatchId="urn:oasis:names:tc:xacml:1.0
      :function:string-equal">
18    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
      #string">A</AttributeValue>
19    <AttributeDesignator Category="urn:oasis:names:tc:xacml:1
      .0:subject-category:access-subject" AttributeId= "roleId"
      DataType="http://www.w3.org/2001/XMLSchema#string"
      MustBePresent="false"/>
20    </Match>
21    </AllOf>
22    </AnyOf>
23    </Target>
24    <Condition>
25    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0
      :function:and"><Apply FunctionId="
      urn:oasis:names:tc:xacml:1.0:function:integer-greater-than
      ">
26    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0
      :function:integer-one-and-only">
27    <AttributeDesignator Category="urn:oasis:names:tc:xacml:3
      .0:attribute-category:environment" AttributeId= "
      environmentId" DataType="http://www.w3.org/2001/XMLSchema#
      integer" MustBePresent="false"/>
28    </Apply><AttributeValue DataType="http://www.w3.org/2001/
      XMLSchema#integer">5</AttributeValue>
29    </Apply>
30    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0
      :function:integer-less-than">
31    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0
      :function:integer-one-and-only">
32    <AttributeDesignator Category="urn:oasis:names:tc:xacml:3
      .0:attribute-category:environment" AttributeId= "
      environmentId" DataType="http://www.w3.org/2001/XMLSchema#
      integer" MustBePresent="false"/>
33    </Apply><AttributeValue DataType="http://www.w3.org/2001/
      XMLSchema#integer">10</AttributeValue>
34    </Apply>
```

```
35    </Apply>
36    </Condition>
37    <AdviceExpressions><AdviceExpression AdviceId="
      urn:notifyUsers" AppliesTo="Permit">
38    <AttributeAssignmentExpression AttributeId="
      urn:notification:message" Category="
      urn:oasis:names:tc:xacml:1.0:subject-category:access-
      subject"><AttributeValue DataType="http://www.w3.org/2001/
      XMLSchema#string">You can store</AttributeValue>
39    </AttributeAssignmentExpression>
40    </AdviceExpression>
41    </AdviceExpressions>
42    </Rule>
43    <Rule RuleId="end-rule" Effect="Deny"><AdviceExpressions><
      AdviceExpression AdviceId="urn:notifyUsers" AppliesTo="
      Deny">
44    <AttributeAssignmentExpression AttributeId="
      urn:notification:message" Category="
      urn:oasis:names:tc:xacml:1.0:subject-category:access-
      subject"><AttributeValue DataType="http://www.w3.org/2001/
      XMLSchema#string">you can not store</AttributeValue>
45    </AttributeAssignmentExpression>
46    </AdviceExpression>
47    </AdviceExpressions>
48    </Rule></Policy>
```

Listing 6.6: XACML poliicy

```
1     {
2     "policyName" : "StorePolicy",
3     "policyAlgorithm" : "permit-overrides",
4     "basicAttributes" : [
5        {
6        "value" : "store", "id" : "actionId", "category" : "
      action"
7        }
8     ],
9     "rule" : [
10       {
11          "action" : "Permit",
12          "id" : "A-can-Store",
13          "basicAttributes" : [
14             {
15                "value" : "A", "id" : "roleId", "category" : "
      subject"
16             }
17          ],
18          "condition" : [
19             {
20                "action" : "and",
```

```
                "basicAttributes" : [
                {
                "value" : "5", "id" : "environmentId", "
   category" : "environment",      "rule":  "greater-than"
                },
                {
                "value" : "10", "id" : "environmentId", "
   category" : "environment"   ,  "rule":  "less-than"
                }
            ]
            }
        ],
        "message" : "You can store"
      }
   ],
   "message" : "you can not store"
   }
```

Listing 6.7: JSON request for creating a XACML policy

After the creation of the policy for a specific actions, a XACML template of how the request for the specific actions should look like is generated. Therefore, if the policy specified the rules of storing a data for a specific role in a time interval, like defined in the policy above, a template is generated defining how the request for storing should look like, including the additional attributes. This template is used by the devices in order their actions to be authorised, including the additional attributes required by the policy. The format of the template is shown below 6.8

```
<Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd
   -17" CombinedDecision="false" ReturnPolicyIdList="false">
<Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-
   category:environment">
<Attribute AttributeId="environmentId" IncludeInResult="false
   ">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
   integer">0</AttributeValue>
</Attribute>
</Attributes>
<Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-
   category:access-subject">
<Attribute AttributeId="roleId" IncludeInResult="false">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
   string">String</AttributeValue>
</Attribute>
</Attributes>
<Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-
   category:action">
<Attribute AttributeId="actionId" IncludeInResult="false">
```

```
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
   string">String</AttributeValue>
</Attribute>
</Attributes>
</Request>
```

Listing 6.8: Authorisation template request

The subscriber responsible for serving the devices request for authorising its actions is given below 6.9. The method `messageArrived` in its parameters has the message received on the topic it was subscribed on. The first commented lines illustrate the structure of that message, which holds the: clientId and password, needed to check whether the device is authenticated, the XACML request, which will be later iterated through the XACML engine to check whether the actions defined in it are permitted, and the topic on which the device waits the result of the authorisation. As it can be seen, the message JSON fields are first parsed into different variables. Next, the validity of the credentials is checked through the method `CheckCredentials.check()`, which communicates with the authentication unit and performs the validation. If the credentials are valid, the actions of the device are iterated through the XACML engine, accessed by the `AuthoriseRequest.authorise()` method. In the end, if the device passes the authorisation, a token is generated through the method `GenerateToken.generate()`, and send to the device on the topic defined in the request message. If this procedure is successful, the device is authorised and can further perform its desired actions.

```
   public void messageArrived(String topicsubscriber,
   MqttMessage message){
      * {
      *  clientId : clientid1
      *  password : pass1
      *  requestXacml : request
      *  topic : returnTopic
      * }
      */
      String response;
      JSONObject json = new JSONObject(message.toString());
      String clientId = json.getString(CLIENTID);
      String password = json.getString(PASSWORD);
      String requestXacml = json.getString(REQUEST_XACML);
      String topicPublishtoken = json.getString(TOPIC);
      if(CheckCredentials.check(clientId, password)) {

         String clientAction = XmlParsings.getClientAction(
   requestXacml);
         AuthorisatonReturnMessage result = AuthoriseRequest.
   authorise(clientId + clientAction, requestXacml);

         if(result.isSuccess()) {
```

```
21              response = GenerateToken.generate(clientId,
    compressRequest(requestXacml));
22          }
23          else
24              response = result.getMessage();
25
26          PublishGeneratedToken.publish(response,
    topicPublishtoken + clientId);
27      }
28  }
```

Listing 6.9: Authorisation template request

So far it was described how the devices actions were authorised, but how the XACML engine is initiated and run, in order the request to be iterated through the policies was not illustrated in details. This can be seen below 6.10, in the method `authorise()`, which invocation was described in the previous snippet. First, the engine, policy decision point (PDP), is initialised, and the XACML request is fed to it through the method `pdp.evaluate()`. When this method is invoked, the request is iterated through all the policies, and the result (permit or deny) is returned back, in the `ResponseCtx` class. In the end the result is checked and further actions are taken, depending whether the actions are permitted or denied.

```
1   public static AuthorisatonReturnMessage authorise(String
    index, String xaccmlRequest) {
2       PDP pdp = new PDP(balana.getPdpConfig());
3       String response = pdp.evaluate(xaccmlRequest);
4
5       ResponseCtx responseCtx = ResponseCtx.getInstance(
    getXacmlResponse(response));
6       AbstractResult result  = responseCtx.getResults().
    iterator().next();
7       if(AbstractResult.DECISION_PERMIT == result.getDecision
    ()){
8       ....
```

Listing 6.10: XACML engine iteratiion through policies

As the devices authorisation was described in more details, in the same manner the token generation is shown in 6.11. It was generated using the AuthO libraries, providing a straight forward method, `JWT.create()`, with many properties for the specific generation. In the payload of the token it is included the unit issuing it, the xacml request, which includes the clients actions and additional attributes, the client id for which the token is generated and the expiration date. In the token presented, the expiration is set in three days, however for better security this interval can be set shorter. In the end the token is signed with the private key of the unit authorising it, and the encryption algorithm is inserted in the header.

```
1     public static String generate(String clientId, String
  xacmlRequest) {
2         Calendar calendar = Calendar.getInstance();
3         calendar.add(Calendar.DATE, 3);
4
5         PrivatePublicKeyHolder holder = getKeys();
6         Date date = calendar.getTime();
7         String issuer =  GenerateKeystore.ALIAS;
8         Algorithm algorithm = Algorithm.ECDSA256((
  ECPublicKey)holder.getPublicKey(), (ECPrivateKey)holder.
  getPrivateKey());
9
10        String token = JWT.create()
11        .withIssuer(issuer)
12        .withClaim("clientId", clientId)
13        .withClaim("xacml", xacmlRequest)
14        .withExpiresAt(date)
15        .sign(algorithm);
16
17        return token;
18        }
```

Listing 6.11: Token generation

6.4 Data Processing and Storing

In the following section it will be described a detailed preview of the implementation of the communication between the devices and the service. Described would be the protocol in which these entities communicate and the processes and methods invoked inside them. First the publishing the devices token and metadata, describing the way the data should be processed, would be illustrated. Following, the methods of checking tokens validity and preparing the services environment for the further data transmission. In the end, the structure of the collected data send to the service would be described.

In initialising the communication between the devices and services, the latter define a strict JSON structure which the device have to follow when sending the first message. The format is given below 6.12. In the commented code in the snippet, it can be noticed that the message contains the following attributes: the authorisation token, the processing tool to be used if the data is processed, the algorithms guiding how the processing should be conducted and the response topic on which the device is subscribed, waiting for the outcome of the initial contact. All of the required fields are parsed into specific variables, needed for different actions, which would be described in the following illustrations.

```
1 *  {
2 *      "token" : "tokenString"
```

```
3  *      "tool" : "procesingTool"
4  *      "algorithm" : "specidig algorithm according the
       tooltool"
5  *      "responseTopic" : "topic"
6  *      "taskId" : "the id of the task to be created"
7  *    }
8     public class TokenReceiverMessageCallback{
9        public static void messageArrived(String
       caCertificatePath, MqttMessage message) {
10           JSONObject json = new JSONObject(message.toString())
       ;
11
12           String token = json.getString(TOKEN_JSON_FIELD);
13           String algorithm = json.getString(
       ALGORITHM_JSON_FIELD);
14           String responseTopic = json.getString(
       RESPONSETOPICJSONAME);
15           String processingTool = json.getString(
       PROCESSING_TOOL_JSON_FIELD);
16           String taskId = json.getString(TASK_ID_JSON_FIELD);
17           ...
```

Listing 6.12: Contact initialisation between the device and the service

After receiving the initial message, first process invoked is checking the validity of the received token. It is illustrated in the code snippet below 6.13. The first step is checking whether the token is not expired, by comparing the current date with the expiration date of the token. As all the validated tokens are stored in the hashset memory variable `tokenSet`, if a token is expired it is removed from the memory and the validation is stopped, with denying the devices actions. In the next step, it is checked whether the token exist in the memory, meaning that it is already validated, in such a case the method return positive value, meaning the device is authorised in performing its actions. However, if the token have not been checked so far, it has to be validated. In doing so, the signature have of the token have to be decrypted with the public key of the unit that generated the token. The unit is defined in the token, so it is checked whether its public key is already in memory in the HashMap variable `authorisationServersSet`. In case such a record does not exist, the service contacts the certificate authority in order the public key to be acquired, using the method `getAuthorityPublicKey()`. After the public key is present, the signature is decrypted, in case it is valid, the token is put in memory and message is sent back informing the validity, allowing other processes to finish the initial contact. with the device.

```
1     private static HashSet<String> tokenSet = new HashSet();
2     private static HashMap<String, PublicKey>
      authorisationServersSet = new HashMap();
3     ...
4     public static boolean checkTokenValidity(String token){
```

```
5       JSONObject json = new JSONObject(getTokenPayload(token)
    );
6
7       Long date = json.getLong("exp");
8       if(date < System.currentTimeMillis()/1000) {
9          if(tokenSet.contains(token))
10            tokenSet.remove(token);
11      return false;
12      }
13
14      if(tokenSet.contains(token))
15         return true;
16
17      String issuer = json.getString("iss");
18            PublicKey pubKey;
19
20      if(!authorisationServersSet.containsKey(issuer)) {
21         String pubKeyStr = getAuthorityPublicKey(issuer);
22         pubKey = convertStringToKey(pubKeyStr);
23         authorisationServersSet.put(issuer, pubKey);
24      }
25      else
26         pubKey = authorisationServersSet.get(issuer);
27
28      try{
29         Jwts.parser().setSigningKey(pubKey).parseClaimsJws(
    token);
30         tokenSet.add(token);
31         return true;
32      } catch (JwtException ex) {
33         return false;
34      }
35   }
```

Listing 6.13: Token validation

After the token has been validated, a task is initiated for the specific action desired by the device, illustrated in the following code snippet 6.14. The illustration starts with checking the tokens validity, after which task is initialised. In the specific example, task is generated specifically for the hadoop processing tool, however, the task can be defined according the needs of the device. For that reason the task class `TaskProperties` is inherited by each specific task class for a different processing tool, in this case the `TaskPropertiesHadoop` class. Certainly if the device have only intentions of storing the data or receiving it, the task would not define any processing algorithms. In the end, the created task is stored in the memory, in the `taskMap` variable of type HashMap, and confirmation message is send back to the device, on previously defined token, meaning that everything is set up and the communication can continue.

```
1    if(TokensHolder.checkTokenValidity(token)){
2       TaskProperties task;
3
4       if(processingTool.equals(HADOOP_PROCESSING_TOOL) &&
    getTask(clientId + taskId) != null)
5       task = new TaskpropertiesHadoop(taskId, algorithm,
    processingTool, action);
6
7       taskMap.put(task.getTaskId(), task);
8
9       TokenValidityResponsePublisher.publish("OK",
    caCertificatePath, responseTopic);
10   }
```

Listing 6.14: Initiate task for a specific action

In the end, the service defines another JSON structure for the data to be sent to it for further processing or storing. The structure is given below 6.15. The token is sent on each message, it is checked whether valid, with the same procedure as defined before. Additionally the taskId is sent, as one token can authorise multiple tasks, and in the end the actual data which needs to be stored and later in some cases processed is sent. The data can be sent in multiple messages, in a streaming manner, however in the end a special character is sent, notifying the end of transmission, and the further processing can be initiated, with the algorithms defined in the initialisation stage.

```
1  {
2    token : token
3    taskId : taskId
4    data : someData
5  }
```

Listing 6.15: Structure of the communication between the devices and the service

6.5 Summary

The most important segments of the implementation stage were presented in this chapter. First, a brief description of the employed development tools and technologies was provided. The following sections presented the developments process of the secure data transmission, the access control mechanisms and the data processing and storing along with the protocols deployed for communicating the data. In order the reader to be further informed with the implementation process code snippets of the most crucial functionalities are illustrated and analyzed in details.

Chapter 7

Testing

7.1 Testring Strategy

Software testing presents a process of executing the developed program with an intent of finding its errors [101]. Additionally this activity can be aimed to evaluate the capabilities of the software and to determine whether it meets its requirements. Despite that the whole activity is initiated in discovering the flaws in the system, finding all of them is generally infeasible. This fact does not come from the fact that the developers are careless, but from the complexity of the software which is generally intractable. However, the testing is still needed, and is one of the essential stages of the development of the project, as product filled with bugs and errors would not be wanted and further used by its users.

Although the testing can be conducted at different stages of the development, as this project was following the Royce waterfall model, the testing was processed after the implementation. In addition on any change in the software, the test cases were repeated, to ensure that nothing was corrupted during the changes, and in the cases of adding additional features, new test cases were written. However, in order to understand the importance of the testing, the main purposes of this process would be defined in the following bullets:

- **Improving the quality:** The quality of the software is crucial in critical situation, in which an occurance of software errors can be severe. Such mistakes can cause airplane crashes, halted trading and market crashes, shuttle missions go awry etc. Therefore testing the software and thus finding these errors are highly important in such an applications.
- **Verification and validation:** Testing can serve as a metric for the completeness of the requirements. Therefore, the completeness of the system and the satisfaction of the developed product can be estimated. Additionally, different products can be measured using the tests, enabling a benchmarking of the given solutions.
- **Reliability estimation:** the reliability on a software product is in high relation to the amount of testings it has been done on it. Therefore the results of

these tests can serve as a statistical samples giving a failure data for further reliability estimations.

Many different types of testing exist which can be suitable in different environments. In the specific project, couple of testing types should be considered. First, each unit of the code should be tested, which is generally integrated in any project, conducted by the unit testings. Next, is the performance testing, which gives valuable information of the performance of the system. As the nature of this project is described by working with computational devices of limited capabilities and resources, and processing a large amount of datasets which are generated at high speed, this type of testing can give valuable information about the efficiency. In the end, the security is also an important aspect of the software developed, therefore security testing is another essential type which have to be considered. Therefore, these tests were implemented, and the errors which occurred were consequently fixed, giving a kind of insurance of a fault free system to some extent, because all the bugs of the system can not be found as stated previously in the chapter. The following chapter would provide a detailed information of the testings conducted and their results.

7.2 Unit Testing

As already stated, the main purpose of unit testing is division of separate testable individual units of the software code, and evaluate their behaviour in isolation from the rest of the units. The worth of this type of testing is increasing with the number of errors identified in the process. Main mechanism of the unit testing is the implementation of drivers and stubs. In more details, the drivers present the unit invocation, while the stubs are presenting the units that to be tested. More simple, the idea of these kind of tests is, providing an input in some isolated unit, and validating whether the specific unit would provide a previously specified outcome. In case it does, the outcome and the prediction are equal, the test can be confirmed as passed. Although, unit testing requires significant amount of time, it produces many advantages. It provides automatisation of the testing process additionally facilitating the detection of the errors. The fact that these tests are automated, means that each test, once created, can be further reused as many times as needed. Therefore, when introducing a changes in the system, or introducing a new feature or functionality, all the units can be retested in an easy and straightforward manner.This step must not be avoided in order to be validated that these additions and modifications are working as expected and in the same time to be ensured that other units are not affected. This further improves the performance of the testing, decreasing the time and cost for validating the quality of each unit.

For the purpose of the project, the unit testing was orchestrated utilising the JUnit framework. It is a java library, which is simple to integrate, providing an out of the box testing possibilities. For better understanding of the testing done, well defined steps would be followed. At first the interface of each test unit would be provided, including the invocation of the unit and the expected value to be returned.

Additionally, the purpose of each specific test would be described. Finally, the results of the tests would be presented, proving the passage and the validity of the testing conducted.

The first unit testing which would be presented is assigned to check the process of importing a new devices in the system. As described in the implementation, the importing can be done only by individuals knowing a special username and password, which are predefined in installation phase of the authentication unit. Together with these "administrator" credentials, the username and password of the new device are sent, and stored in the database of the unit. However, couple of mechanisms are integrated in this process, first the creation can be done only by presenting the correct administrator credentials and the two equal username credentials in the database must not exist, enabling easy identification of the devices. The testings, validating these properties are illustrated on the code snippet 7.1. The first test, `ImportClientCredentials()` validates whether a proper insertion of device and administrator credentials would result in successful importing of the credentials. The second test `ImportClientCredentialsWrongAdministrator()`, validates whether wrong credentials would stop the insertion, thus returning a null value. In the end the third test case `ImportClientCredentialsController` checks whether duplicate credentials are possible to be stored, if this feature is not possible, as it is desired, the method of insertion should also return a null value.

```
1  @Test
2  public void ImportClientCredentials() {
3      ImportClientCredentialsController importClientcredentials
       = new ImportClientCredentialsController();
4      ClientCredentials client = importClientcredentials.
       ImportCredentials("device1", "password", "admin", "admin")
       ;
5      assertEquals("device1", client.getClientId());
6  }
7
8  @Test
9  public void ImportClientCredentialsWrongAdministrator() {
10     ImportClientCredentialsController importClientcredentials
       = new ImportClientCredentialsController();
11     ClientCredentials client = importClientcredentials.
       ImportCredentials("device1", "password", "admin1", "admin"
       );
12     assertEquals(null, client);
13 }
14
15 @Test
16 public void ImportDuplicateClientCredentials() {
17     ImportClientCredentialsController importClientcredentials
       = new ImportClientCredentialsController();
18     assertEquals(null, importClientcredentials.
       ImportCredentials("device1", "password", "admin", "admin")
```

```
    );
}
```

Listing 7.1: Devices authentication testing

Despite the characteristic of non-duplicate username credentials, the database should not store two identical password credentials in identical hashed strings. This means, that even if the devices have identical passwords, they would be hashed to a different string, procedure described in the implementation chapter, with details of the benefits from it. This property of the system was tested, with the procedure illustrated on the snippet 7.2. The testing function `checkDuplicatePasswords()` at first creates two device credentials and stores them in the database. Later on their passwords are acquired using the method `DatabaseStatements.getClientPassword()`, which access the database and returns the passwords of the previously stored credentials. The two passwords are checked if equal, and in case they are not the testing was successful, meaning that no two passwords can be equal in the database.

```
@Test
public void checkDuplicatePasswords() {
    ImportClientCredentialsController importClientcredentials
    = new ImportClientCredentialsController();
    importClientcredentials.ImportCredentials("device2", "
    passwordSame", "admin", "admin");
    importClientcredentials.ImportCredentials("device3", "
    passwordSame", "admin", "admin");

    String client1Password = DatabaseStatements.
    getClientPassword("device2");
    String client2Password = DatabaseStatements.
    getClientPassword("device3");
    boolean equal = client1Password.equals(client2Password);
    assertEquals(false, equal);
}
```

Listing 7.2: Duplicate password testing

Another unit test was conducted on the procedure of creating an access control policies in the authentication unit. As already described in the implementation, a specific structure have to be followed when policies are sent to be stored and later enforced at the specific unit. Therefore, a corresponding tests were processed in order this insertion process to be validated, code snippet 7.3 illustrates the tests. The first test case `insertPolicy()` checks whether policy formatted according the required structure would be stored in the authentication unit, process which would return back “Policy generated” message back. On the the side, in case the policy is not formatted as it should, the message returned is ”Wrong input”, the testing of the wrong case is illustrated with the second test, `insertWrongPolicy()` . The string values of the policies were not shown in the snippet, for simplicity reasons, as the pictures would require much bigger space.

```
@Test
public void insertPolicy() {
   String policy...\\policy is defined in the correct format

   InsertPolicyController insertPolicy = new
   InsertPolicyController();
   assertEquals("Policy generated", insertPolicy.insertPolicy
   (policy));
}

@Test
public void insertWrongPolicy() {
String policy...\\policy is NOT defined in the correct format
InsertPolicyController insertPolicy = new
   InsertPolicyController();
assertEquals("Wrong input", insertPolicy.insertPolicy(policy)
   );
}
```

Listing 7.3: Insert policy testing

After testing the policy creation, it is naturally to check whether the policies are enforced properly. For that reason two tests were conducted, illustrated on the snippet 7.4. The first test, `authorisationTestPermitted()` have a XACML request in which the actions and the role defined can pass the authorisation process, on the other side the request in the second test, `authorisationTestNotPermitted()`, is crafted in a way that it can not pass the authorisation. The first would have to generate true result in case the authorisation is working properly, while on the other side, the second test needs to output false value.

```
@Test
public void authorisationTestPermitted() {
String requestXACML;\Request from device of role B,
   authorised to perform its actions
...
String clientAction = XmlParsings.getClientAction(
   requestXACML);
AuthorisatonReturnMessage result = AuthoriseRequest.authorise
   ("B" + clientAction, requestXACML);
assertEquals(true, result.isSuccess());

}

@Test
public void authorisationTestNotPermitted() {
String requestXACML;\Request from device of role A, NOT
   authorised to perform its actions

```

```
15 String clientAction = XmlParsings.getClientAction(
      requestXACML);
16 AuthorisatonReturnMessage result = AuthoriseRequest.authorise
      ("A" + clientAction, requestXACML);
17 assertEquals(false, result.isSuccess());
18
19 }
```

Listing 7.4: Authorisation policy enforcment testing

After authorising the devices actions, follows the generation of the token. This token have the property of proof of possession, meaning that the holder of it is able to perform the actions defined in it. However, before the service perform the desired actions, the token has to be validated. The validation mechanism has to be further tested, if it is working in every possible scenario. These testings can be noticed on the snippet 7.5, in which 3 tests are presented. The first one, `tokenValidation()`, checks whether a correctly generated token would pass the validation process, in that case the result of the testing should return true. The second test, `tokenValidationChangedCharacter()`, validates whether a change in just one character of the token would result in a corrupted token string, returning a false statement on the test. In the end, as the tokens have expiration date, a testing was conducted, `tokenValidationChangedCharacter()`, whether such a token would be noticed by the validation process, returning false, as same as the second test.

```
1  @Test
2  public void tokenValidation() {
3  Security.addProvider(new org.bouncycastle.jce.provider
4  .BouncyCastleProvider());
5
6  String token = "eyJ0eXAiOiJKV1...."
7
8  assertEquals(true, TokensHolder.checkTokenValidity(token));
9  }
10
11 @Test
12 public void tokenValidationChangedCharacter() {
13 Security.addProvider(new org.bouncycastle.jce.provider
14 .BouncyCastleProvider());
15
16 String token = "ayJ0eXAiOiJKV1...."\\first character of the
      token is changed
17
18 assertEquals(false, TokensHolder.checkTokenValidity(token));
19 }
20
21 @Test
22 public void expiredTokenValidation() {
23 Security.addProvider(new org.bouncycastle.jce.provider
```

```
24 .BouncyCastleProvider());
25
26 String token = "eyJhbGciOiJIUzI1NiJ9..."\\token with expired
      date
27
28
29 assertEquals(false, TokensHolder.checkTokenValidity(token));
30 }
```

Listing 7.5: Token validation testing

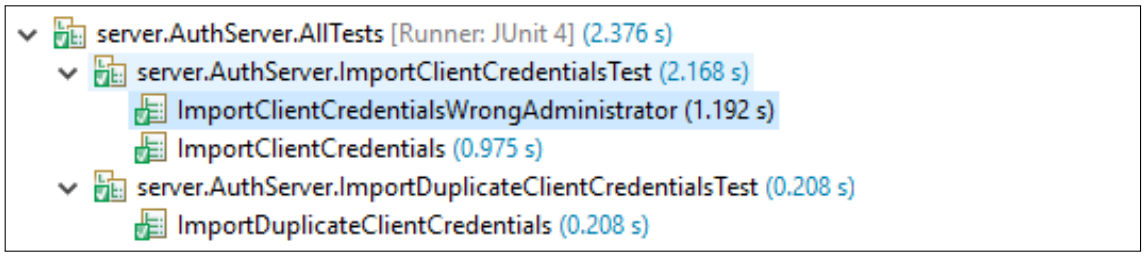
All of the testing conducted ensure that the software developed have the qualities and the main bugs fixed and maintained. Giving an good basis for further employment in the environment. However, as the nature of the software engineering is constantly maintaining and correcting the mistakes, as the complexity of the systems developed are hard to grasp by the developers, the testing process would continue during the lifetime of the project. The results of the testings are illustrated on figure 7.1, in the same order as they were described in the chapter.

7.3 Performance Testing

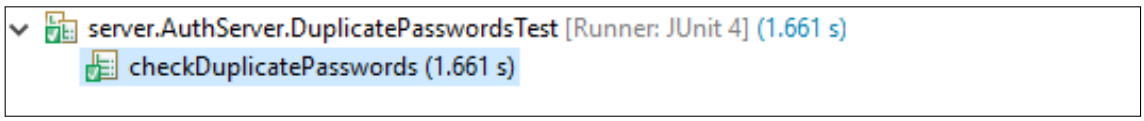
Performance testing is another form of software testing, in which the main focus is how the system performs under a particular workload [102]. In opposite of unit testing, this kind of test does not uncover the errors in the system but, it measures the performance according to a specific benchmark and standards. The results of the testing process gives valuable diagnostic information needed to eliminate the bottlenecks. It can also provide validate and measure the quality, non-functional requirements of the system, such as resource usage, scalability, reliability, throughput and responsiveness. In addition to the immense contributions from the results of the testing, it is commonly conducted in accomplishing specific objectives. Determining production readiness, crucial before releasing the product, finding of the source of performance problems, essential in the maintaining process and in occasions of low performance, comparison with similar projects, boosting the competitiveness, and in the end utilising this testing can accomplish further performance system tunings.

In order to fully gain the benefits provided by this kind of testing, many the different types of performance testings have to be defined and understood well. However, before getting deeper in the different kinds of performance checks, the parameters which are measured during those testings has to be mentioned. The following are the main hardware parameters which are monitored:

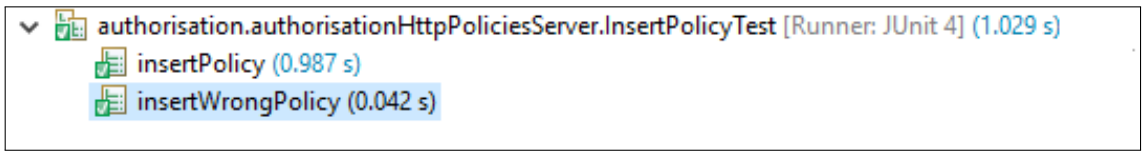
- CPU Utilisation, the time needed by the CPU to process a request
- Memory Utilisation, the amount of memory needed to process a request
- Disk utilisation, the usage of the disk in a processing a request



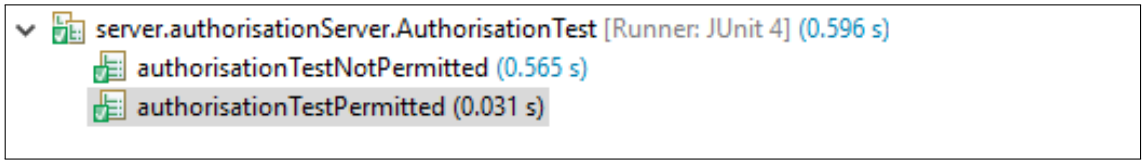
(a) Devices credentials testing



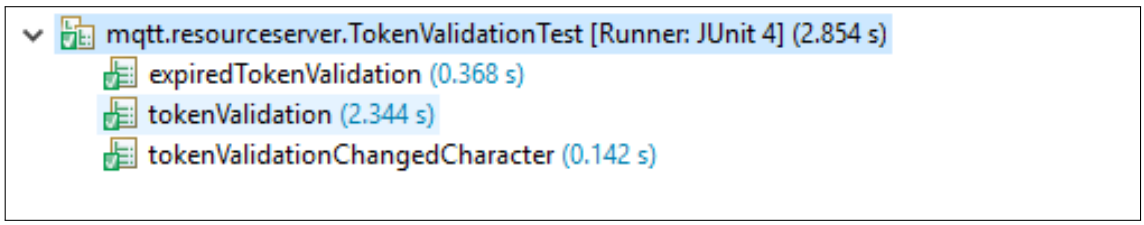
(b) Duplicate device password testing



(c) Policy insertion test



(d) Authorisation testig



(e) Token validity testing

Figure 7.1: Unit testing results

- Throughput, measured usually by kilobytes per second, showing the amount of network bandwidth corrupted during the test
- Concurrent users, how many active users can be served at a particular moment
- Request pers second, the number of requests handled

Hence, when different testings are conducted the following parameters, are monitored providing an indication on where the bottlenecks can or already appear. Therefore, after defining the monitoring criteria, the types of the testings can be further defined:

- **Load testing** In these tests the performance of the system is verified under normal conditions, further increasing the workload reaching the peak conditions. They are crucial for determining whether the system can meet the desired performance objectives. Load testing enables measuring the response time, throughput rate and level of resource utilisation, and most importantly it identifies the application breaking point.
- **Stress testing** As opposite of load, stress testing validates the system's behaviour under conditions which are beyond normal or peak conditions. In these scenarios, the system is given more users transaction than it can handle, with a goal of defining the software stability, the point when it fails and the manner in which it recovers. Additionally, stress testing provides a mechanism of revealing bugs under high workload conditions, usually uncovering synchronous issues, memory leaks and race conditions. A subset of stress testing is the **spike testing** which validates the performance in scenarios when the workloads are increased quickly and repeatedly, beyond the normal threshold for a short period of time.
- **Endurance testing** Known as soak testing, is an evaluation of the work of the system, under normal workload for an extended period of time. The main goal of this test is validating the system problem which can occur over time, such as memory leaks (a scenario which occurs if the system fails to release the discarded memory). Endurance testing provides an calculations of Mean Time Between failure (MTBF), Mean Time To Failure (MTTF) and other similar metrics.
- **Capacity testing** This kind of testing determines the number of users or transactions that the system can support and still meet the performance goals. This information can be crucial when planning on increasing the growth of the system transactions, as it would give an detailed information of the current capacities, and the future tunings which have to be performed.
- **Volume testing** also known as flood testing, as this test determines the efficiency of the system under a scenarios of large amounts of data to be further managed. It is called flood testing as the system is constantly flooded with with data.

As it can be seen many different testings can be conducted, measuring and validating the performance capabilities of any system. The nature of the developed project, requires efficient service of vast amounts of IoT devices, producing large volume datasets which have to be further managed. Therefore, there is not one type of performance testing which have to be highlighted, neither one to be abandoned. All of them, measure a different and crucial aspect of the system.With the load tests, it can be determined whether the project would satisfy its main purpose, of serving big amounts of devices requests. The results of the stress testings would give valuable information whether the system would be capable of enduring big amount of requests, which is a possible scenario in the IoT world, as serving many devices is one of the main objectives. Another test, which is related with this property, of numerous devices, is the capacity testing. Because it was already described in the literature review, that the IoT paradigm, along with its devices, is growing in immerse rates, this test is crucial in estimating whether the whole system would survive in the future. Another aspect which is tested is the endurance of the system, which is important for any kind of project released in the environment. In the end, as the data generated from the IoT is enormous, proper volume testing is needed to ensure that data flooding would not occur in the system, which is highly possible to happen when working with big datasets.

Despite the big benefits from the performance testing, appropriate testings were not feasible to be conducted. The main reason is the need of large number of IoT devices, in order the testings to be properly done, which amounts need to reach up to thousands of devices. In addition, as the access control units, the CA authority and the services, providing data processing and storing, are all separate components, in which the latter is a distributed network of machines, it was also infeasible to be tested as there was a need of numerous computer machines. However, as already defined in the analysis and the design of the system, performance was one of the main characteristics considered during the development of the project. For that reason, a lightweight and highly efficient message protocol MQTT, was utilised in the communication with the IoT devices. Additionally, the public key encryption was handled using the ECC encryption algorithm, which ensures efficient encryption using a smaller sized keys. In the end, the services provide a mechanisms of integrating different big data processing tools, which enable easy managing of the distributed processing power, but also providing a mechanisms for adding a new machines, widening the distributed system even more.

7.4 Security Testing

As already described in the literature review, security has a high importance today, that would be even more significant in the future. Many attacks and vulnerabilities have been shown with immense cost. Therefore, one of the largest non-profit project, related to security, OWASP project [103], was established at the beginning of the century. Their main goal is providing an unbiased information of the best practices in improving the security of software. From their resources, highly useful and critical information can be gathered, in understanding the security needs in any project.

Despite the different research fields, one of the areas this project emphasizes is the security testing.

Security testing, mainly defines a variety of software testings which ensure that the whole system is free of an loopholes that may cause a malicious breaches and loss of information. These testings, provide the developers of the system with an insight of the possible loopholes which exist. However, as OWASP suggests, the system is not only the software that is implemented, it is the whole organisation of the people, processes between the people and the software, and the processes between the components. Therefore, their main suggestion is taking a holistic approach when considering the testing. This rule also applies in any other security activity. So, whenever the security is tested, a proper mechanisms should be defined for validating the technologies, the people, how they use the software and whether they follow the security policies defined, and in the end how well these policies are defined.

When it comes to defining the stage in the project in which security testing should be conducted, OWASP suggests including tests in each phase of the project. One of the first approaches was patch-and-detect, in which security vulnerability was fixed after reporting it from the community, the users which are interacting with the system. However, this approach is highly unsuccessful, as the time needed to find the proper solution of the vulnerability and thus releasing the patch, many malicious breaches can occur. Additionally, many of the users would not install the security patch, leaving them open to the threats. Therefore, the approach of conducting a security tests on each step of development is found to be more suitable. The main disadvantage is the higher cost, in comparison with patch-and-detect, however the vulnerability that can occur later can overshadow the initial cost of the first approach.

When conducting the testings couple of principles need to be known and followed in order the testings to cover as much of the vulnerabilities in the system:

- **No silver bullet** There is not an out-of-the box ready to plug-in security solution which will solve all the vulnerabilities. Therefore, thinking that a security defender or a firewall is enough protection can only additionally harm the project. For that reason, an in-depth tests which will cover the whole system are needed.
- **Strategic thinking** This principle indicates the already described need of testing at each stage of the development, replacing the patch-and-detect manner.
- **Test early and often** The price of fixing a software vulnerability can be addressed faster and at lower cost if it is detect in early stage of development. The main suggestion in making this possible is educating the individuals responsible for the development and testing of the product, about common security issues and the ways of detect and prevent them.

- **Understand the scope of security** First of all it is essential to know the things that are needed to be protected and the possible threats. Next, a well understanding of the security requirements of the environment is needed(e.g. the state in which the software is use). Additionally, protection goals have to be defined, and proper testing should be conducted of the manners in which they are implemented.

- **Right mindset** Even though the software would work without any security in a scenario of a normal use, the attackers usually go beyond this, trying to find a way of breaching the protection. Therefore, when it comes to developing and testing of the security, thinking "outside of the box" is required, in order the unexpected behaviour of the attackers to be understood and simulated.

- **Use the Right Tools** Although it was already mentioned that there is not one tool solution for the security issues, tools still play a critical role. Understanding the capabilities and the benefits from these tools and the way they should be properly implemented would lead to a secure project.

Despite the principles which have to be followed, there are couple of testing techniques which can be followed. These techniques are suggested as an efficient way of finding the most of the vulnerabilities. Therefore, better understanding of them would highlight the advantages and disadvantages of each one indicating which one should be considered.

- **Manual inspections** This type of testing is typically conducted by a human, testing the security implications of people, processes and policies. Manual inspections are the most simple reviews, but also can be among the most powerful and effective one. Inspection is also done on the whole architecture, checking how the components are connected indicating if an obvious design vulnerability exist. Additionally, the policies are also validated by these inspections. Couple of advantages exist in this type: requires no support by technology, flexibility, and can be applied at any stage. On the other side the main disadvantages are the time consumption and the requirement of significant amount of though skill.

- **Threat modeling** It is a highly new and popular technique which help developers think about the possible vulnerabilities the system could face. It gives a chance mitigation strategies to be developed for potential threats, limiting the threats at the development phase. OWASP outlines that threat modeling can be used as a reference for the security testing, leaving a big freedom of creating the threat models as there is no right or wrong way of doing it. The main advantage is the practical attacker view of the system and the flexibility of doing it at any time. On the other side, it is relatively new technique without a strict guidelines, additionally it is not yet widely tested, meaning that good threat model does not automatically mean good security.

- **Source code review** This kind of testing suggest manual checking of the source code with an intention of discovering a security issues. Despite the fact

that many automatic testings exist which are more efficient, this kind of testing is proven to be irreplaceable by any other testing method. Many security experts are suggesting this approach, indicating that significant security problems are extremely difficult to be discovered with other from of testing and analysis. Issues that are found through a source code testing include concurrency problems, access control problems, flawed business logic, cryptographic weaknesses, backdoors, etc. One of the main advantages of source code review is its completeness and effectiveness, moreover accuracy is another benefit, also it can be conducted fast, by competent reviewers. On the other side it requires highly skilled security reviewers, issues in external libraries could be missed and error detection during run-time is impossible. Additionally this type of testing can not be automated, making it infeasible to be conducted frequently.

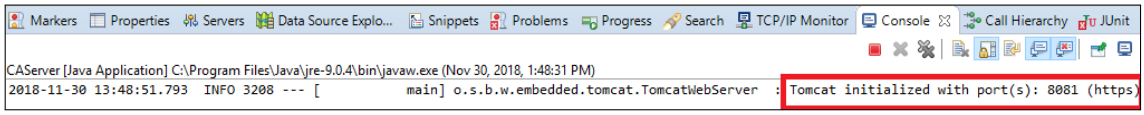
- **Penetration testing** It is been the most common security testing in the community, which is also known as black box testing or ethical hacking. The main goal of penetration testing is finding the security vulnerabilities of a specific application which is already deployed and running in the environment. Typically the testers have access to the application as regular users, with a role of an attacker, trying to exploit the vulnerabilities of the system. Many penetration tools are available which can automate the testing process, however some of the application require more unique approach. Despite the fact that this testing is frequently used and discovers many of the security issues, it should not be the only type of testing. As many of the experts are saying, penetration tests discover how bad is the application after it is deployed. Therefore, proper testing mechanisms during the development of it are more needed. The advantages of this kind of testing is that it can be fast and cheap, require a lower skill than source code review and the testing is done on a product which is actually exposed to the users. However, couple of disadvantages exist, as already mentioned it can be too late, as the vulnerabilities and flaws are already part of the working application, and it is front impact testing only.

In the project there were conducted two types of security testings, manual inspections and source code review. In the first, only the overall architecture was tested, as the people process and policy testing is not applicable to a project not released in the working environment. As the architecture is divided in couple of processing units and IoT devices, one of the most important issues is the secure communication. The testings conducted to validate whether the communication is protected are illustrated on figure 7.2. In which the figures `a)` and `b)` present the communication protocol integrated in the communication with the certificate authority and the authentication unit, which is the encrypted HTTPS. Additionally, the illustration shows the communication with the MQTT broker. Under `c)` it can be seen the configuration file, indicating that tls would be used as a communication protocol, but also the brokers port. In the end, the illustration `d)` presents a screenshot of a package transmission between the IoT device and the broker, on the specific port shown in the last column, as it can be seen the communication is established over

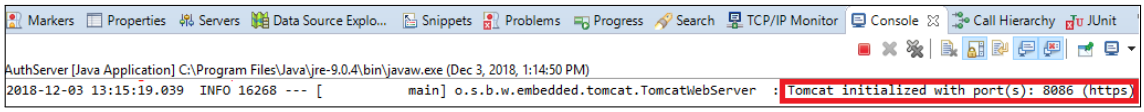
the secure TLS protocol. mn, as it can be seen the communication is established over the secure TLS protocol.

Moreover, the unit testings presented on chapter 7.2 validate the security properties of the system. The password salting, implemented for a more secure credential storage is checked and validated under the test illustrated on figure 7.2. Further, the unit testings present the token validation procedure, checking the scenarios of corrupted and expired token on figure 6.13, with a positive results that such a scenarios are prevented of occuring in the system.

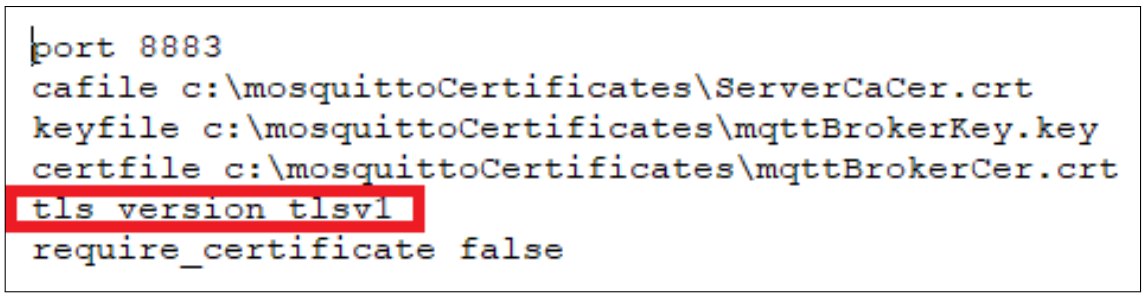
When it comes to the source code review, this type of testing resulted in minor changes in the authorisation protocol, without any further big security flaws noticed. It is important mentioning why the other two types of security testings were not conducted, the penetration testing and the threat model. One of the main reasons was that this the project developed is not implemented in an distributed system in a real case scenario, which is essential for the two mentioned tests, especially for the penetration testing. Additionally, as it is just an proposal of a specific architecture, the threat model and the attackers can not be defined until the software is integrated in a specific real life project.



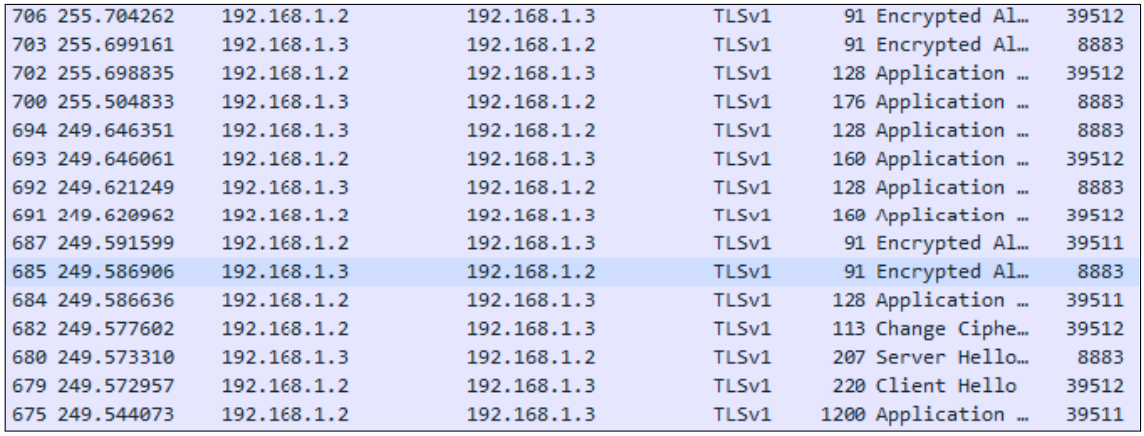
(a) Certificate authority secure communication



(b) Authenctication unit secure communication



(c) MQTT broker security configuration



(d) Secure communication with the MQTT broker

Figure 7.2: Secure communication

Chapter 8

Evaluation

This chapter will illustrate the results of the evaluation of the system. The main intention is to conclude whether the developed system satisfies the initial objectives and requirements of the project. In case the results are positive, it can be said that the project successfully fulfilled its main aim.

8.1 Evaluation Methodology

In order to validate the problem solving process and the proposed system significant attention was given on the process of evaluating the project. For more accurate evaluation, the evaluation process was divided in two parts. First, the main functionalities of the system were evaluated, defined in the analysis phase,. Giving a deeper validation whether the system developed has the required qualities. Second, the main objectives, defined at the beginning of the project, were evaluated, in order to verify whether the main aim of the project is reached.

8.2 System Functionalities Evaluation

This chapter would demonstrate the evaluation of the functional and non-functional requirements defined in the analysis of the project. The evaluation would start with the functional requirements, following the non-functional. Each of them would be described in details in the following bullets. Starting with the functional, the following evaluation was conducted:

- **Big data processing and storing**
  As the main aim of the project was developing a service which would enable processing and storing data generated from IoT devices, it is crucial to validate whether this functionality is achieved. The section 5.3 illustrates the implementation of the storing and processing features. Therefore, it can be concluded that one of the main objectives and risks in the project have been successfully fulfilled.

- **Lightweight data transmission**
  One of the main requirements was finding an efficient protocol in transmitting

the data from the devices to the service. For this purpose, in the project the lightweight MQTT protocol is used, as described in section 5. Additionally, the literature review, to be more precise the section 2.4, illustrates the efficiency and lightweightness of the communication protocol. This can ensure that the specific functional requirement was achieved.

- **Secure data transmission** The other crucial feature of the data transmission is its transfer throughout a secure channels. For this purpose the the systems architecture contains a certificate authority enabling a communication through the secure SSL protocol. The implementation of this security aspect can be reviewed on section 5.1. The corresponding testing of the implementation is illustrated on section 7.4, providing a proof that this requirement is also achieved.

- **Unique identity of each device**
  One of the requirements essential for further efficient communication with the devices is the unique identification of each device. Additionally, this requirement is the ground for the further access control which is implemented. Without unique Id there is not a possibility of authorising the actions of specific devices. The section 5.2.1 in the design chapter illustrates this property, while this property is checked whether implemented correctly in the unit testing section 7.2.

- **Access control on devices actions**
  One of the crucial security tasks is the access control on the devices actions. For this reason in the systems architecture was integrated a sophisticated token based authorisation and authentication mechanism. The design decision of these two units is presented in section 5.2. However the policy language of deciding which users actions would be permitted or denied was defined previously in the literature review, in the authorisation subsection 2.3.2. In the end the access control was tested and thus proven to be implemented and working, the results of these tests can be found in the unit testing section 7.2.

- **Secure credentials storing**
  The last functional requirement ensures that the credentials are stored on a secure way. Prevention mechanisms were integrated, ensuring that stealing the database in which these credentials are stored, would not uncover the usernames and passwords of the devices. The technique which was used is described in details in the section 6.3. Validation of the secure credential storing can be found in the unit testings 7.2, which ensures the implementation of this mechanism.

Despite the implementation of the functional requirements, which would satisfy the main feature of the system, the non-functional requirements are also an important part. These requirements can not be visually seen, however, they define the quality attributes of the project, which are of big importance. The specific non-functional qualities in the project along with the manners in which they were implemented are presented in the following bullets:

- **Efficiency**
  As already mentioned in the description of this requirement, efficiency applies on many different aspects of the project. First, it is applicable to the communication protocol, which implementation in the system is already described in the previous evaluations, under the lightweight data transmission bullet. Second, the encryption of the data has to be efficient, as the IoT devices have limited resources. The cryptographic algorithm utilized in achieving the required efficiency is described on section 5.1. In the end, the authorisation procedure has to be conducted in quick fashion. The literature review, section 2.3.2, illustrates a policy language which is abled in authorising device requests in fast manner.

- **Maintainability**
  The importance of maintainable code is already highlighted in the analysis part. However, achieving such a requirement is not an easy task. Therefore, a various architectural and design patterns were researched, with the most suitable ones utilised in the project. These patterns, described in the design section 5, give the code a property to be easily grasped and further managed and changed. Additionally, maintainability of the code is increased with an documentation, presented with this document

- **Availability**
  One of the main characteristics of any system working over the web is availability. However, proper evaluation of this requirement can not be given without testing the system after developing it in its environment. Therefore, the performance testing section 7.3 elaborates in more details how these testings can be conducted, thus evaluating the availability of the project.

- **Extensibility**
  As already described, extensibility provides a quality of further growth of the system. However, this growth can be applicable in different areas. At first, it means that the system should encourage easy integration of new devices. This property is achieved through the interfaces of the authentication unit, described in section 5.2. On the other side, extensibility should be achieved in integrating different processing and storing tools in straightforward fashion. For this reason a specific design patterns were followed, section 5.4.

- **Adaptability**
  The architecture of the project is designed with the microservice architecture pattern in mind. This means that each unit of the system is decoupled and individual. Therefore, this property is making the integration of the different units in a different scenarios achievable.

Through these bullets the functional and non-functional requirements of the system were evaluated. The results are clearly showing that the requirements were successfully implemented. In addition, these requirements would give the basis of further evaluation of the projects objectives.

8.3 Project Objectives Evaluation

This section would provide evaluation of the objectives of the project. It would be followed the same bullet structure, as the requirements evaluation, including a project references to the document. Additionally, some of the functional requirements may be a proper evaluation for a specific objective, therefore some of them would be additionally mentioned.

O1 Creating a secure and lightweight two way data transmission service interface, for communication with the IoT devices.
In order any further manipulation on the IoT generated data to be conducted, the service should provide a proper interface. As it is one of the main characteristics the service should posses, this objective is defined at the inception of the project, in the section 3.1, in which the vision of the project is described. Further on, as security is one of the main goals, it is further elaborated and evaluated in the following objectives. When it comes to the lightweightness of the data transmission, MQTT protocol was utilised. The literature review, section 2.4, illustrates the efficiency of the protocol. Later on, in the chapter 5, it is described the reason of choosing the specific protocol.

O2 Creating the main functionalities of the service: processing, storing and interface for data retrieving.
As already described, this objective fulfills the main functionalities of the service, and thus the aim of the project. Therefore its implementation forms the basis for the further development of the system. The section 3.1, describing the vision of the project, incepts these functionalities. Furthermore, in sections 5.3 and 6 these functionalities are further elaborated, defining their design and the way they are implemented.

O3 Implementing the service to act as a broker for different big data processing tools, meaning that the clients can define the tool they want (Apache Hadoop, Spark, Hive etc.) for processing their data.
As it was already described in the non-functional requirement evaluation, extensibility was one of the integrated qualities in the system. The strategy design pattern, section 5.4 gives an easy and straightforward way of adding a different big data processing tools, with completely different interfaces. This gives the opportunity, the metadata to enable defining a different tool, thus, giving the service the desired property to act as an broker for the specific tools.

O4 Creating a functionality on the broker, which will enable the IoT devices to define the model describing the manner in which the data will be processed.
As the service is not responsible for providing a different processing models, this objective is crucial for further processing of the data. Its definition can be found in the section 5.3, Design, and the way it is implemented in the section 6.4. Additionally, different big data processing tools accept different formats for the processing model. This part of the objective is achieved through the strategy design pattern, described in section 5.4.

O5 Defining a structure for the data accompanied by a metadata. The metadata will contain the information for the IoT data manipulation actions: storing, processing and retrieving, big data processing tools and the data processing model.
The success of the previous three objective are directly defining the completion of this one, as the metadata consist of the procedures and functionalities described in them. Additionally this objective resolves the interoperability issue, as there is one strict communication structure, which any IoT device has to follow.

O6 Implementing data transmission protection mechanism.
One of the main security challenges is the protection of the data transmission. Therefore this challenge was initially mentioned in the chapter 3.1, Vision of The Project. Further on, this issue was described in more details in the functional requirements, and its implementation is referenced in the evaluation of the Secure data transmission. The section 7.4, illustrates the results of the testings conducted in evaluating whether the communication is secured.

O7 Implementing security level allowing only authorised IoT devices to access the service.
This objective presents another important security aspect of the system, therefore its proper implementation is important. In the section 5.2, it is illustrated how this objective is implemented on a design level, introducing the XACML access control policy language. The way the access control is implemented is presented in section 6.3. In the end, the UnitTesting, section 7.2, evaluates the access control mechanism implemented.

From the evaluation of the objectives of the project, provided by the above bullets, it can be confirmed all of them were reached and the system consist of the properties defined in them. Moreover, as the main aim of the project, depends on the implementation of the objectives, it can be further concluded that the main aim of the project was successfully met.

Chapter 9

Conclusion and Future Work

From the inception of the project, the main idea was developing a service capable of storing and processing data generated from various IoT devices. Additionally, in order this aim to be achieved couple of objectives were defined, further describing the steps needed to develop the service required.

At the beginning, in order to better understand the needs and challenges of the IoT network, wide research was done in that field. This lead to a realisation that IoT and Big data are highly related topics, which would become even more coupled in the future. Therefore, an insight of the Big data subject was given. Further on, it was realised that the security is one of the biggest challenges that have to be addressed, in order the IoT to fulfill its high predictions in the future. However, as the devices have limited resources and capabilities, adding a security features in them is not an easy task. With that in mind, a deep review in the literature was conducted, with a purpose of finding an efficient ways of integrating the security in the IoT network.

However, before starting to implement the specified objectives, there is a need of a proper planning and development methodology. This is essential for any kind of project, as without proper strategy there is a high possibility of missing the time frame required, but also missing the main objectives. The first stage of the methodology was defining the functional and non-functional requirements of the system. They represent a further and more detailed description of the objectives, written in a more technical way. The requirements are also giving a guidance for further development of the project, defining the tasks which need to be completed in order to fulfill the aim.

With the strictly defined software methodology and well described requirements, the next step of designing and implementing the system are not difficult to grasp. Moreover, the results from the literature review performed at the beginning was crucial in determining the mechanisms and processes which are needed in developing the end product. The system is consisting of an architecture composed of different units, each responsible in fulfilling a different objective of the project. One of the units is obviously the service responsible for data processing and storing. However,

for an efficient enabling of the security properties, there were implemented three other units, certificate authority, authorisation and authentication unit. In the end the architecture suggest utilising a tokens for the access control purposes. These tokens are holding the authorised actions of each IoT device, which can later simply present it to the service for further processing and storing. The suggested approach does not require any computational power by the devices in generating the specified token, just an insignificant amount of memory. Additionally, the decoupling of the units in the architecture provides a possibility some parts of it to be integrated in other projects, e.g. improving the security of an existing IoT system.

In the end the testing and evaluation phase proved the viability of the presented work. It demonstrated that the main challenges of the IoT can be overcomed if they are tackled appropriately. However, as this project is just suggesting an architecture of a efficient IoT processing and storing service, further work is required.

First, as it was already stated in the testing phase, a deep performance testing was not conducted. Performing these testings requires couple of more powerful machines along with thousands of IoT devices. However, a strategy of performing these tests was described. Additionally, some of the security testings defined were not conducted, from the same reasons as the performance testings. Therefore, in order this architecture to be better tested it is required to be implemented in a working environment, which is a bit costly for a project of this type.

Second, while conducting the literature review, couple similar products, developed by the big software companies were found. Although they provide highly sophisticated solutions, there is a big risk of breaking the confidentiality of the data stored on their side. Therefore, if possible it is highly suggested of constructing an own architecture as suggested in the project. However, it was noticed that all of the implemented projects are following an Service Level Agreement [104] (SLA). This agreement defines the level of service expected from the service providers, responsible for the data storing and processing. Therefore, if the further development is in mind with employing the project in the working environment, it is highly recommended of creating a mechanism to implement the SLA. With this agreement, the customers can be ensured of the reliability, responsiveness, availability and many other qualities. It provides a ground for mutual agreement between the client and the service, which the service provider have to guarantee during the work cycle of the service.

In the end, the security properties of the system have to be regularly tested and improved, as everyday new and unique malicious attacks and breaches are appearing. One highly interesting and usable secret sharing mechanism was discovered during the research of the project, zero knowledge bases cryptography [105]. This cryptographic algorithm, provides a way of proving that an entity is holding a piece of information without discovering it to any other entity. The algorithm can be highly usable in proving the credentials and identity of any device in the network, without

showing their credentials. As it can be seen, this mechanism makes it impossible of any stealing of usernames or passwords during their transmission.

Bibliography

[1] D.-L. Yang, F. Liu, and Y.-D. Liang, "A Survey of the Internet of Things," *Proceedings of the 1st International Conference on E-Business Intelligence (ICEBI2010)*, no. May, pp. 358–366, 2010.

[2] "Internet of things (iot) connected devices installed base worldwide from 2015 to 2025 (in billions)." `https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/`. Accessed: 2018-06-08.

[3] V. Gazis, M. Goertz, M. Huber, A. Leonardi, K. Mathioudakis, A. Wiesmaier, and F. Zeiger, "Short Paper : IoT : Challenges , Projects , Architectures," pp. 145–147, 2015.

[4] M. Antonakakis, T. April, M. Bailey, M. Bernhard, A. Arbor, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, A. Arbor, L. Invernizzi, M. Kallitsis, M. Network, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, Y. Zhou, M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai Botnet This paper is included in the Proceedings of the Understanding the Mirai Botnet," *USENIX Security*, 2017.

[5] Y. Ma, J. Rao, W. Hu, and X. Meng, "An Efficient Index for Massive IOT Data in Cloud Environment," 2012.

[6] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, 2014.

[7] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. Ullah Khan, "The rise of "big data" on cloud computing: Review and open research issues," *Information Systems*, vol. 47, pp. 98–115, 2015.

[8] I. of Things Global Standards Initiative, "Iot definition." `https://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx`. Accessed: 2018-12-06.

[9] Z.-k. Zhang, M. Cheng, Y. Cho, C.-w. Wang, C.-w. Hsu, C.-k. Chen, S. Shieh, and I. Fellow, "IoT Security : Ongoing Challenges and Research Opportunities," 2014.

[10] J. Liu, Y. Xiao, and C. L. P. Chen, "Authentication and Access Control in the Internet of Things," pp. 588–592, 2012.

[11] W. Shang, Q. Ding, A. Marianantoni, J. Burke, and L. Zhang, "Securing Building Management Systems Using Named Data Networking," no. June, pp. 50–56, 2014.

[12] Statista, "Worldwide data created statistics." https://www.statista.com/statistics/871513/worldwide-data-created/. Accessed: 2018-12-06.

[13] S. Kaisler, F. Armour, J. A. Espinosa, and W. Money, "Big Data: Issues and Challenges Moving Forward," *2013 46th Hawaii International Conference on System Sciences*, pp. 995–1004, 2013.

[14] Statista, "Alibaba income." https://www.statista.com/statistics/298844/net-income-alibaba/. Accessed: 2018-12-06.

[15] Varonis, "60 must-know cybersecurity statistics for 2018." https://www.varonis.com/blog/cybersecurity-statistics/. Accessed: 2018-12-06.

[16] Nortron, "10 cyber security facts and statistics for 2018." https://us.norton.com/internetsecurity-emerging-threats-10-facts-about-todays-cybersecu
html. Accessed: 2018-12-06.

[17] B. Schneier, "Schneier questions need for security industry." https://www.cnet.com/news/schneier-questions-need-for-security-industry/. Accessed: 2018-12-06.

[18] R. J. Anderson, *Security Engineering*, vol. 2. 2008.

[19] Y. Cherdantseva, J. Hilton, O. Rana, and W. Ivins, "A multifaceted evaluation of the reference model of information assurance & security," *Computers & Security*, vol. 63, pp. 45–66, 2016.

[20] A. M. Nia, S. Member, and N. K. Jha, "A Comprehensive Study of Security of Internet-of-Things," vol. 6750, no. c, pp. 1–19, 2016.

[21] R. L. Rivest, "Handbook of theoretical computer science (vol. a)," pp. 617–755, 1990.

[22] B. Schneier, "The value of encryption." https://www.schneier.com/essays/archives/2016/04/the_value_of_encrypt.html. Accessed: 2018-12-06.

[23] W. Diffie and M. E. Hellman, "Multiuser cryptographic techniques," pp. 109–112, 1976.

[24] S. Ant, R. Chaves, and L. Sousa, "AES and ECC Cryptography Processor with Runtime Configuration,"

[25] P. Rogaway and T. Shrimpton, "Cryptographic Hash-Function Basics : Definitions , Implications , and Separations for Preimage Resistance , and Collision Resistance," pp. 371–388.

[26] B. Schneier, "Cryptanalysis of md5 and sha: Time for a new standard." https://www.schneier.com/essays/archives/2004/08/cryptanalysis_of_md5.html. Accessed: 2018-12-06.

[27] Crackstation, "Salted password hashing - doing it right." https://crackstation.net/hashing-security.htm. Accessed: 2018-12-06.

[28] S. Boonkrong, "Security of Passwords," vol. 8, no. 2, pp. 112–117, 2012.

[29] C.A. Ardagna, E. Damiani, S. De Capitani di Vimercati, and P. Samarati, "XML-based Access Control Languages," *Information Security Technical Report*, vol. 9, pp. 35–46, 2004.

[30] X. Yao, X. Han, X. Du, S. M. Ieee, and X. Zhou, "A Lightweight Multicast Authentication Mechanism for Small Scale IoT Applications," vol. 13, no. 10, pp. 3693–3701, 2013.

[31] V. S. Subrahmanian, "Flexible Support for Multiple Access Control Policies Universit a," vol. 26, no. 2, pp. 214–260, 2001.

[32] S. D. C. Vimercatil, S. Foresti, S. Jajodia, and P. Samarati, "Access Control Policies and Languages in Open Environments,"

[33] "WS-Policy." https://www.w3.org/Submission/WS-Policy/. Accessed: 2018-11-10.

[34] Oasis, "OASIS eXtensible Access Control Markup Language (XACML) version 1.1.."

[35] T. Yokotani, "Comparison with HTTP and MQTT on Required Network Resources for IoT," pp. 0–5, 2016.

[36] Flespi, "Http v mqtt performance test." https://flespi.com/blog/http-vs-mqtt-performance-tests. Accessed: 2018-12-06.

[37] IBM, "Getting to know mqtt." https://developer.ibm.com/articles/iot-mqtt-why-good-for-iot/. Accessed: 2018-12-06.

[38] MQTT.org, "Mqtt documentation." http://mqtt.org/documentation. Accessed: 2018-12-06.

[39] HiveMq, "Mqtt qos." https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels. Accessed: 2018-12-06.

[40] HiveMq, "Mqtt persistent session." https://www.hivemq.com/blog/mqtt-essentials-part-7-persistent-session-queuing-messages. Accessed: 2018-12-06.

[41] Microsoft, "Microsoft azure cloud iot solutions." https://azure.microsoft.com/en-us/services/iot-central/. Accessed: 2018-12-06.

[42] Google, "Google cloud iot solutions." `https://cloud.google.com/solutions/iot/`. Accessed: 2018-12-06.

[43] Amazon, "Amazon aws cloud iot solutions." `https://aws.amazon.com/iot/`. Accessed: 2018-12-06.

[44] B. Schneier, "New data privacy regulations." `https://www.schneier.com/blog/archives/2018/06/new_data_privac.html`. Accessed: 2018-12-06.

[45] G. Suciu, V. Suciu, A. Martian, R. Craciunescu, A. Vulpe, I. Marcu, S. Halunga, and O. Fratu, "Big Data , Internet of Things and Cloud Convergence – An Architecture for Secure E-Health Applications," 2015.

[46] S. Pirbhulal, H. Zhang, E. E. Alahi, and H. Ghayvat, "A Novel Secure IoT-Based Smart Home Automation," pp. 1–19, 2017.

[47] "Mendeley tool." `https://www.mendeley.com/?interaction_required=true`. Accessed: 2018-06-08.

[48] W. Royce, *Software project management*. Pearson Education India.

[49] J. Henry and H. Joel, *Software project management: A real-world guide to success*. Pearson/Addison Wesley, 2004.

[50] B. Hughes and M. Cotterell, *Software project management*. Tata McGraw-Hill Education, 1968.

[51] I. Wescon, "MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS Dr. Winston W. Rovce INTRODUCTION," no. August, pp. 1–9, 1970.

[52] M. A. Cusumano and S. Smith, "Beyond the Waterfall : Software Development at Microsoft," 1995.

[53] Y. H. Kwak and J. Stoddard, "Project risk management : lessons learned from software development environment," vol. 24, pp. 915–920, 2004.

[54] "Software requirements analysis and specifications," in *Software Requirements Engineering*, pp. 137–139, John Wiley and Sons Inc, 2011.

[55] A. Balalaie, "Microservices Architecture Enables DevOps : An Experience Report on Migration to a Cloud-Native Architecture The Architectural Concerns for Microservices Migration The Architecture of Backtory Before the Migration," pp. 1–13.

[56] D. Garlan and M. Shaw, "An Introduction to Software Architecture," *Knowledge Creation Diffusion Utilization*, vol. 1, no. January, pp. 1–40, 1994.

[57] "Definitions of architecture." `https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=513817`. Accessed: 2018-11-10.

[58] D. Notkin, "Formalizing Design Spaces : Implicit Invocation Mechanisms,"

[59] P. Avgeriou, "Architectural Patterns Revisited – A Pattern Language," pp. 1–39.

[60] Y. Chen and T. Kunz, "Performance Evaluation of IoT Protocols under a Constrained Wireless Access Network," 2016.

[61] M. A. Jan, P. Nanda, X. He, Z. Tan, and R. P. Liu, "A robust authentication scheme for observing resources in the internet of things environment," *Proceedings - 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2014*, pp. 205–211, 2015.

[62] A. Kumar, "Security for IoT," *2015 International Conference on Advances in Computer Engineering and Applications (ICACEA)*, pp. 163–166, 2015.

[63] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol," pp. 1–104, 2008.

[64] G. Apostolopoulos, V. Peris, and D. Saha, "Transport Layer Security : How much does it really cost ?,"

[65] "MQTT protocol." `http://mqtt.org/documentation`. Accessed: 2018-10-18.

[66] "CoAP protocol." `https://datatracker.ietf.org/doc/rfc8323`. Accessed: 2018-10-18.

[67] "XMPP protocol." `https://xmpp.org/about/technology-overview.html`. Accessed: 2018-10-18.

[68] "ietf: X.509 certificate." `https://www.ietf.org/rfc/rfc3280.txt`. Accessed: 2018-11-10.

[69] "Telecommunications Union's Standardization sector." `https://www.itu.int/en/Pages/default.aspx`. Accessed: 2018-11-10.

[70] A. I. Guide, "Elliptic Curve Cryptography," pp. 1–11.

[71] K. Lauter, "The Advantages of Eliptic Curve Cryptography for Wireless SEcurity," no. February, pp. 2–7, 2004.

[72] V. Hu, T. Grance, D. Ferraiolo, and D. Kuhn, "An Access Control Scheme for Big Data Processing," *Proceedings of the 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2014.

[73] "OAtuh." `https://oauth.net/`. Accessed: 2018-10-18.

[74] "Statista: Annual number of data breaches ." `https://www.statista.com/statistics/273550/data-breaches-recorded-in-the-united-states-by-number-of-breaches-and-recor` Accessed: 2018-11-10.

[75] "Crackstation: Salted Password Hashing." `https://crackstation.net/hashing-security.htm`. Accessed: 2018-11-10.

[76] S. Cirani, M. Picone, P. Gonizzi, L. Veltri, and G. Ferrari, "IoT-OAS: An oauth-based authorization service architecture for secure services in IoT scenarios," *IEEE Sensors Journal*, vol. 15, no. 2, pp. 1224–1234, 2015.

[77] S. Sciancalepore, G. Piro, D. Caldarola, G. Boggia, and G. Bianchi, "OAuth-IoT: An access control framework for the Internet of Things based on open standards," *2017 IEEE Symposium on Computers and Communications (ISCC)*, pp. 676–681, 2017.

[78] "Ouath2.0: proof of possesion." `https://tools.ietf.org/html/draft-ietf-oauth-pop-architecture-08`. Accessed: 2018-11-10.

[79] B. J. D, U. S. A, and M. D. G, "Analysis of DAC MAC RBAC Access Control based Models for Security," vol. 104, no. 5, pp. 6–13, 2014.

[80] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models," vol. 29, no. 2, pp. 38–47, 1996.

[81] X. Jin, R. Krishnan, and R. Sandhu, "A Unified Attribute-Based Access Control Model Covering DAC , MAC and RBAC," pp. 41–55, 2012.

[82] I. Mavridis, C. Georgiadis, G. Pangalos, and M. Khair, "Access Control based on Attribute Certificates for Medical Intranet Applications," *Journal of Medical Internet Research*, vol. 3, 2001.

[83] "ietf: JSON Web Token." `https://tools.ietf.org/html/rfc7519`. Accessed: 2018-11-10.

[84] C. Larman, "Applying UML and patterns: an introduction to object oriented analysis and design and interative development," 2002.

[85] E. Agerbo and A. Cornils, "How to preserve the benefits of Design Patterns," pp. 134–143.

[86] J. Vlissides, R. Helm, R. Johnson, and E. Gamma, "Design patterns: Elements of reusable object-oriented software," *Reading: Addison-Wesley*, vol. 49, no. 120, p. 11, 1995.

[87] K. Stencel, "Detection of Diverse Design Pattern Variants," pp. 25–32, 2008.

[88] J. E. Mcdonough, *Adapter Design Pattern. In: Object-Oriented Design with ABAP*. Apress, Berkeley, CA, 2017.

[89] "IoT for all: Top 3 Programming Languages for IoT Development In 2018." `https://www.iotforall.com/2018-top-3-programming-languages-iot-development/`. Accessed: 2018-11-10.

[90] "Bouncy castle." https://www.bouncycastle.org/. Accessed: 2018-11-10.

[91] "Balana XACML engine." https://github.com/wso2/balana. Accessed: 2018-11-10.

[92] "Mosquitto MQTT Broker." https://mosquitto.org/. Accessed: 2018-11-10.

[93] "Paho MQTT client." https://www.hivemq.com/. Accessed: 2018-11-10.

[94] "Android studio." https://developer.android.com/. Accessed: 2018-11-10.

[95] "Android studio." http://www.eclipse.org/. Accessed: 2018-11-10.

[96] "JSON data format." https://json.org/. Accessed: 2018-11-10.

[97] "XML data format." https://www.w3.org/TR/REC-xml/. Accessed: 2018-11-10.

[98] "MYSql database." https://www.mysql.com/. Accessed: 2018-11-10.

[99] "Apache hadoop." https://hadoop.apache.org/. Accessed: 2018-11-10.

[100] "Microsoft shell." https://msdn.microsoft.com/en-us/library/windows/desktop/bb773177(v=vs.85).aspx. Accessed: 2018-11-10.

[101] P. Ammann and J. Offutt, *Introduction to software testing*. Cambridge University Press, 2016.

[102] "Performance testing guidance for web applications." https://docs.microsoft.com/en-us/previous-versions/msp-n-p/bb924375(v=pandp.10). Accessed: 2018-06-08.

[103] "Owasp testing guidance." https://www.owasp.org/index.php/Main_Page. Accessed: 2018-06-08.

[104] P. Patel, A. H. Ranabahu, and A. P. Sheth, "Service Level Agreement in Cloud Computing," 2009.

[105] A. Rice, P. Wong, and D. Cohen, "Zero-Knowledge Cryptography,"