

ENABLING ATTRIBUTE BASED ACCESS CONTROL WITHIN THE INTERNET OF THINGS (IoT)

A Thesis Submitted to the College of
Graduate Studies and Research
In Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
In the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Marwah Hemdi

PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science

176 Thorvaldson Building

110 Science Place

University of Saskatchewan

Saskatoon, Saskatchewan

Canada

S7N 5C9

ABSTRACT

With the wide-scale development of the Internet of Things (IoT) and the usage of low-powered devices (sensors) together with smart devices, numerous people are using IoT systems in their homes and businesses to have more control over their technology. Unfortunately, some users of IoT systems that are controlled by a mobile application do not have a high level of data protection to respond in case the device is lost, stolen, or used by one of the owner's friends or family members. The problem studied in this research is how to apply one of access control methods in an IoT system whether they are stored locally on a sensor or on a cloud. To solve the problem, an attribute-based access control (ABAC) mechanism is applied to give the system the ability to apply policies to detect any unauthorized entry by evaluating some of the users' attributes: the accessed time, the device media access control address (MAC address), the username, and password. Finally, a prototype was built to test the proposed solution in two ways; one is locally on a low-powered device, the second using a cloud platform for the data storage. To evaluate both the prototype implementations, this research had an evaluation plan to mimic the real-world interactions by obtaining the response times when different numbers of requests were sent from diverse numbers of users in different delays. The evaluation results showed that the first implementation was noticeably faster than the second implementation.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor, Dr. Ralph Deters. Your supervision, advice, and consideration are highly appreciated. I am sincerely grateful to you for your kindness and positive feedback during the whole journey of this thesis. Thank you.

In addition, I would like to thank my committee members, Dr. Julita Vassileva, Dr. Jim Greer, and Dr. Daniel Chen for their guidance and advice. I greatly appreciate your expertise and all the feedback you were able to share.

I would also like to thank the King Abdullah Scholarship Program, Saudi Arabia, for helping to fund this research. Additionally, I would like to thank the Saudi Cultural Bureau in Canada and the staff from the Department of Computer Science; especially, Ms. Gwen Lancaster for her help and support during my study at the University of Saskatchewan.

DEDICATION

This thesis is dedicated with deep love to my mother (Karimah) for her continuous love, prayers, and encouragement throughout my journey. Mom, your support was the biggest motivation for me and without any doubt you are the person who I owe most for pursuing my graduate studies.

I also dedicate this work to my amazing husband (Oday), who gave me a great support throughout my studies. He has been supportive of my work and shared with me all my challenging and happiness moments to complete this thesis. His love and encouragement gave me strength and patient to complete my master studies.

I also dedicate this thesis to my great brother (Mohammed) who was always behind me with his support and encouragement. Finally, to all my family members and my great friends back home and here in Saskatoon, thank you so much. I love you all, and I have not enough words to express my gratitude for your love, support, and encouragement.

TABLE OF CONTENTS

Permission to Use	i
Abstract.....	ii
Acknowledgements	iii
Dedication.....	iv
Table of Contents	v
List of Tables	vii
List of Figures.....	viii
List of Abbreviations	x
Introduction	1
Problem Definition	8
Literature Review	12
3.1 Attributes Based Access Control (ABAC).....	12
3.2 Security challenges.....	16
3.3 RESTful web services	20
System Architecture and Implementation	23
4.1 First implementation	25
4.1.1 Client Side	26
4.1.2 Server Side	27
4.1.3 Web services	31
4.2 Second implementation.....	33
4.2.1 Client Side	33
4.2.3 Web services	36
4.2.2 Server Side	37

Evaluation.....	40
5.1 First implementation evaluation results	40
5.2 Second implementation evaluation results	43
Summary and Contribution	48
Future Work.....	52
References	55

LIST OF TABLES

<u>Table</u>	<u>page</u>
3-1. Four of the worst data security breaches in the 21st century.....	19
6-1. Evaluation.....	40

LIST OF FIGURES

<u>Figure</u>	<u>page</u>
1-1. Internet of Things (IoT).....	1
1-2. List of countries by Internet of Things devices online per 100 inhabitants as Published by the OECD in 2015.....	2
1-3. Forecast of the number of devices that connected to the IoT Worldwide (billions).....	3
1-4. The increasing number of smartphones from 2007-2014 (millions).....	5
2-1. Problem definition.....	9
3-2-1. Types of cyber-attacks experienced by companies in the United States as of August 2015.....	16
4-1. First proposed solution.....	24
4-2. Second proposed solution.....	24
4-1-1. Single Board Computer (Raspberry Pi 2)	25
4-1-2. First implementation.....	25
4-1-3. Copper (Cu) CoAP user-agent for Firefox web browser.....	27
4-1-4. Abstract Layering of CoAP.....	32
4-2-1. Second implementation.....	33
4-2-2. The Interface of the Application.....	34

5-1. The evaluation setup for the first implementation.....	41
5-1-1. The response times when 100 requests sent per five users.....	41
5-1-2. The response times when 50 requests sent per 10 users.....	42
5-1-3. The response times when 25 requests sent per 15 use.....	42
5-2. The evaluation setup for the second implementation.....	44
5-2-1. The response times when 100 requests sent per five users.....	44
5-2-2. The response times when 50 requests sent per 10 users.....	45
5-2-3. The response times when 25 requests sent per 15 use.....	46
7-1. HTTP vs. HTTPS.....	53

LIST OF ABBREVIATIONS

<u>Abbreviation</u>	<u>Description</u>
ABAC	Attribute Based Access Control
ABE	Attributes-Based Encryption
AC	Access Control
ACLs	Access Control Lists
CC	Cloud Computing
CoAP	Constrained Application Protocol
DAC	Discretionary Access Control
DoD	Department of Defense
GAE	Google App Engine
Golang	Go language
GPU	Graphics Processing unit
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IaaS	Infrastructure as a Service

IBAC	Identity-Based Access Control
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IT	Information Technology
JSON	JavaScript Object Notation
M2M	Machine to Machine
MAC	Mandatory Access Control
MAC address	Media Access Control address
OrBAC	Organization-Based Access Control
P2P	Point to Point
PaaS	Platform as a Service
RBAC	Role Based Access Control
REST	REpresentative State Transfer
SaaS	Software as a Service
SBC	Single Board Computer
SOC	System On Chip
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

URI

Uniform Resource Identifier

CHAPTER 1

INTRODUCTION

The Internet of Things (IoT) can connect different devices such as sensors (small devices that can be programmed to complete a specific task) or smart devices together, which each will complete a task. IoT concept is like a networked interconnection of devices; it can connect a large number of devices, sensors, or buildings (see Fig. 1-1).

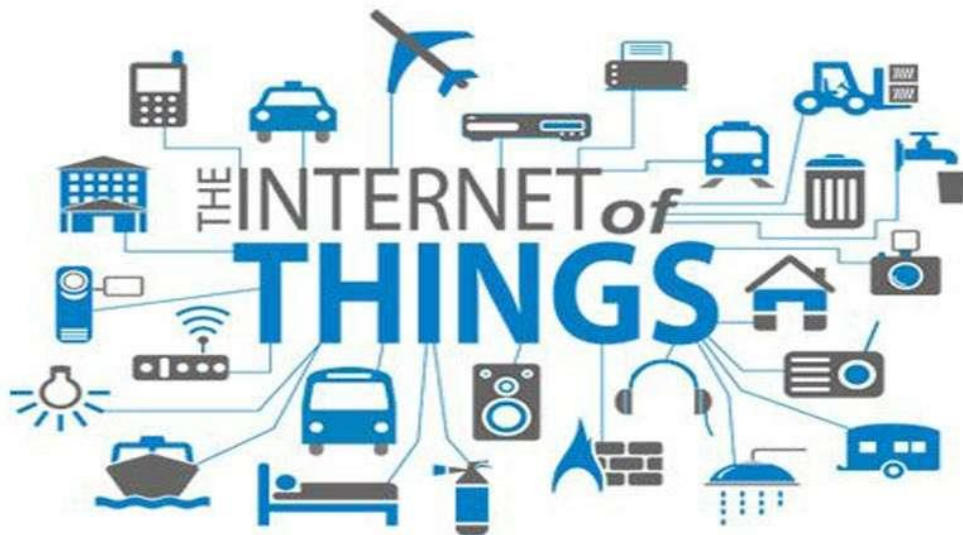


Figure 1-1. Internet of things (IoT)

IoT was defined in the Recommendation ITU-T Y.2060 (06/2012) as “a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies.”¹ IoT allows objects (sensors and effectors) to detect and control remotely across existing network structure, producing chances for more direct incorporation between the physical world and computer-based systems. For example, customers use Philips hue LED bulbs to control their lights in homes through an application on their tablets or smartphones. Moreover, in Philips

¹ <http://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx>

hue LED bulbs “There’s also an API that allows Netflix to dynamically change the lighting based on a movie scene’s color palette, extending the movie experience beyond the screen” [1]. Another example is the service that Hilton hotel is offering to their guests, which are smartphone-based check-ins and room key functionality [1].

Consequently, IoT is becoming incredibly powerful and popular around the world, and each IoT application will have a big number of users. Fig. 1-2 shows a list of countries and the amount of IoT devices per 100 people.

The following is a list of countries by IoT devices online per 100 inhabitants as published by the OECD in 2015.^[34]


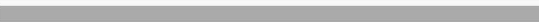









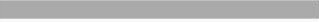

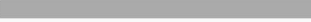

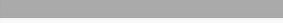

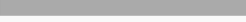

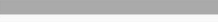




Rank ↕	Country ↕	Devices online ↕	Relative size ↕
1	 South Korea	37.9	
2	 Denmark	32.7	
3	 Switzerland	29.0	
4	 United States	24.9	
5	 Netherlands	24.7	
6	 Germany	22.4	
7	 Sweden	21.9	
8	 Spain	19.9	
9	 France	17.6	
10	 Portugal	16.2	
11	 Belgium	15.6	
12	 United Kingdom	13.0	

Figure 1-2. List of countries by Internet of Things devices online per 100 inhabitants as published by the OECD in 2015²

² <https://www.linkedin.com/pulse/iot-ebrahim-dashty>

Also, many studies predicted that the number of devices that connected to the IoT would increase more worldwide in the forthcoming years as Fig. 1-3 shows next:

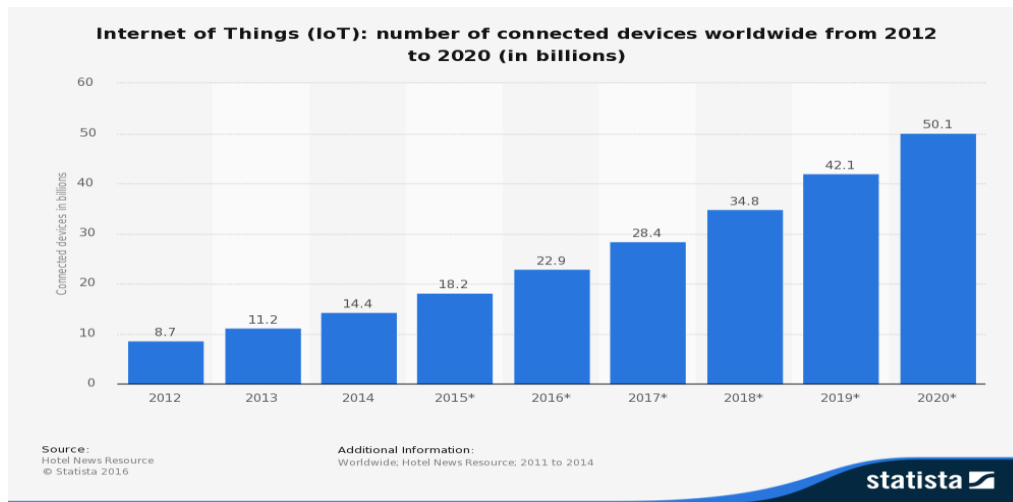


Figure 1-3. Forecast of the number of devices that connected to the IoT worldwide (billions) ³

Although probably the concept of IoT became popular recently, the terminology goes back to 1999. Kevin Ashton, co-founder of MIT's Auto-ID Center, is attributed by most sources with inventing the term "Internet of Things." However, the acronym, IoT, appears to be a noticeably later innovation. Worth mentioning here is the fact that Ashton's idea of IoT-focused on using radio frequency identification (RFID) technology to connect devices together [2]. That was similar to today's IoT, but still, there are many differences. IoT today depends on IP networking mainly to let devices exchange a broad range of information; however, Ashton's concept of an RFID-based IoT in 1999 relied on wireless networking (it was still in the beginning), and cellular networks had not yet converted to a fully IP-based configuration. Device manufacturers put small stock in an RFID-based IoT. As an outcome, the world's first Internet-connected refrigerator, the LG Internet Digital DIOS, was introduced featuring a LAN port for IP connectivity by June 2000. "(The fridge had been under development since 1997, showing that the idea if not the name for IoT existed well before Ashton introduced the term in 1999)" [2].

Furthermore, the sensors became a valuable tool to complete simple tasks in the real world. Sensors were defined as "Sensors are used to measure physical quantities such as temperature,

³ <https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/>

light, pressure, sound, and humidity.”⁴ In general, developers can program the sensors to predict a particular change in an environment or recover physical information. In addition, sensors became powerful devices in computer science; they gained popularity in the developers’ community because:

- They are simple devices to process.
- Their small sizes make them portable (and they become smaller).
- Their procedure like a computer, it takes data, process it, and produce an output.
- They can be programmed to complete simple everyday tasks such as click a button in an elevator, and turn on or turn off a lamp.

Besides, the usage of Cloud Computing solves the problem of having a huge amount of storing data in servers due to the cloud has unlimited capacity.

Additionally, the number of smart devices such as tablets and mobile phones has increased significantly in recent decades. They have an essential role in humans’ lives; people now use their mobile devices in everyday tasks, such as making calls, sending texts, and doing online operations that require secretive information. Examples include buying products, making bank transfers, or checking banking accounts. The number of smartphone users has grown in recent years, and experts estimate that it will continue to rise. Wang, Wei, and Vangury stated that 821 million mobile devices, including tablets and smartphones, were sold in 2012; in 2013, the number of devices sold increased by 46%. Also, 2.5 billion smart devices were sold in 2015 [33]. Fig. 1-3 shows the increasing number of smartphones from 2007 to 2014 in millions.

⁴ <http://www.bbc.co.uk/schools/gcsebitesize/ict/measurecontrol/0computercontrolrev2.shtml>

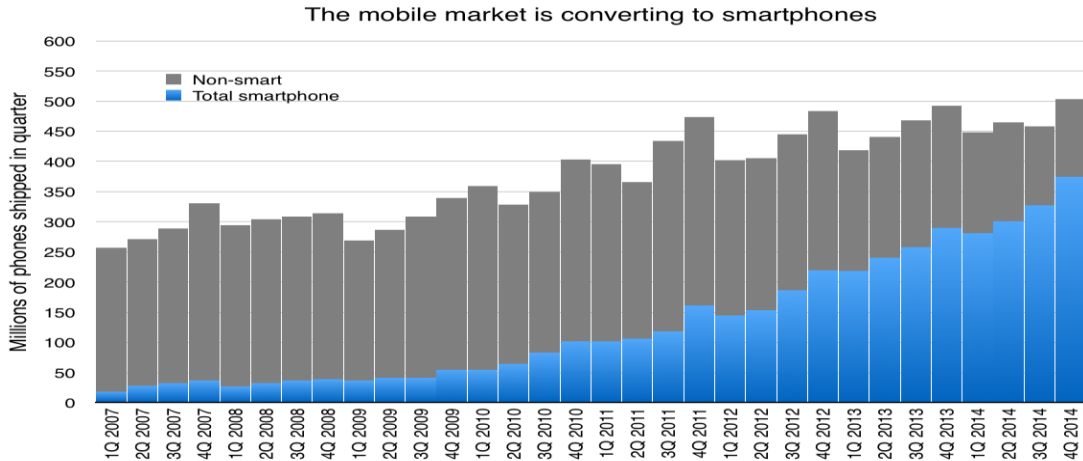


Figure 1-4. The increasing number of smartphones from 2007-2014 (millions) ⁵

For the last decades, the feature phones that had only some features developed to smartphones that have an operating system, like a small PCs. As an outcome, those devices became more popular, and more people obtained them.

The term cloud computing (CC) gives flexibility to mobile devices' and sensors' users; everything they used to access using PCs or laptops, can be accessed using only their personal smartphones. For example, CC opens the opportunity for users to interact with business services. Users can download applications such as iPhone apps and Google apps from Apple and Google cloud space. These applications offer entertainment and all kinds of help for users' personal and professional lives.

With web services like Hypertext Transfer Protocol (HTTP) and Constrained Application Protocol (CoAP), both based on REpresentative State Transfer (REST), sending and retrieving data has become easier. This research will explore both concepts.

IoT applications are a promising idea because they help people to interact faster with the services they need from the application provider, and gives the users a smart, convenient, and comfortable experience in their homes or businesses. Also, most of IoT systems use low cost and power computers to achieve a particular task to the users. As well, they build a mobile application for them, so they could communicate with IoT system and control the provided services in the

⁵ <https://theoverspill.wordpress.com/2015/02/09/android-oem-pro-fitability-and-the-most-surprising-number-from-q4s-smartphone-market/>

system like changing the temperature in their homes or offices. However, the idea of devices, which can access the system's data, that is stored locally or on a cloud consider being dangerous due to the device could, be lost or stolen anytime. In addition, any application has one layer of authentication like username and password mechanism is not secure enough because nowadays there are many ways to crack the password.

This thesis investigates the problem of how to apply access control in an IoT environment by implementing an attribute-based access control (ABAC). By focusing on this issue, the system will protect the data in the application from unauthorized entry, an entry that could allow access to private data like the application owner's personal information.

It is important for a homeowner's, business owner's, or IoT system's provider to have a high level of security, both because of the increase in harmful software such as spyware and malware, and the danger from cyber-attack or data leakage. Any of the previous software could monitor, record, steal, transfer, or worst of all, destroy the data on the attacked device. Therefore, if an unauthorized person could access that information in IoT system, the owner of the system could lose its reputation and, in a commercial situation, could lose money as well. If the system was in a house, the personal and private information of the homeowners would be hacked, which could lead to safety threats.

This research focuses on solving this problem by applying one access control mechanism, Attributes Bases Access Control (ABAC), to users' attributes specifically time, device, username, and password. The built system is applying some policies on attributes to detect any suspicious behavior on the part of the user. With time, the system will recognize each 'user context' depending on the comparison between the entered attributes to the pre-defined policies; the system would deny access for users whose attributes do not match the policies. The proposed solution could be applied to an existing applications' infrastructure. Finally, because one of the concerns in IoT system is the large number of users and how the application will work in an overload situations, the built proposed solution was evaluated by testing it in an overwork different scenarios (diverse numbers of delays, requests, and users) to compare the response time results between the two implementations.

The remaining sections of the thesis are structured as follows:

- Chapter 2 discuss the definition of problems
- Chapter 3 reviews the related technologies and research
- Chapter 4 explains the solution's architecture, design, and implementations
- Chapter 5 describes the evaluation experiment and represent the results
- Chapter 6 states the summary and the contribution of this thesis
- Chapter 7 discusses the future work

CHAPTER 2

PROBLEM DEFINITION

The importance of IoT systems has been growing in the recent years in many domains like (medicine, manufacturing, environment, and so on). To the level where people start installing IoT systems in their homes for different purposes such as entertainments, or security. On the other hand, like any technology in our lives, IoT systems developers have done lots of research and contributions to design and build IoT systems that can perform useful tasks to the users. Thus, IoT systems gained popularity, and an enormous number of users with their data are stored on those systems.

With having a rising number of IoT systems and users using them, some concerns about the systems' security and users' privacy were raised.

Some of the security area concerns [3]:

- Securing IoT structure.
- Having a level of protection in IoT to prevent random attacks.
- Having a level of protection in IoT to prevent attacks by malware.

Some of the privacy area concerns [3]:

- Control of personal data (privacy of data).
- Enhance the privacy knowledge.
- Creating methods and tools to handle the identity of users and objects.

These concerns are some of main challenges in IoT; this research will focus more on them in chapter 3 (the literature review).

Relating to the challenges were mentioned above, afterward is a description of the problem definition of this thesis.

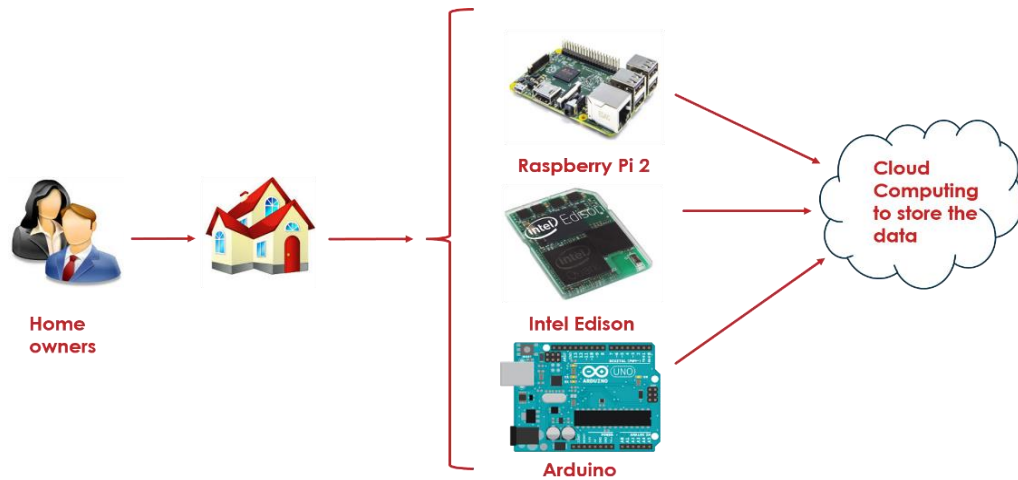


Figure 2-1. Problem definition

Fig.2-1 is demonstrating the problem description of this research, people building IoT systems in their houses by installing sensors in them. Three different sensors were put as examples :

- Single Board Computer (SBC); for instance, Raspberry Pi 2 (the one is used in this project) ⁶
- System On Chip (SOC); for instance, Intel Edison⁷
- Arduino⁸

Sensors grow into many of our real-world life aspects; especially, after using them in IoT systems. People now have sensors in their homes; they want to use the technology for their own benefits such as reduce their energy costs. For instance, a homeowner has an IoT system, which uses sensors to achieve the next:

- Recover the temperature in the house.
- Turn off or turn on the cooling or the heating system.
- Modify the system settings.

⁶ <https://www.canakit.com/raspberry-pi-starter-ultimate-kit.html>

⁷ <http://www.bit-tech.net/news/hardware/2014/01/07/intel-edison/1>

⁸ <https://www.arduino.cc/en/Guide/Introduction>

In the previous situation, the system will allow the users three operations:

- READ to retrieve the temperature.
- WRITE to switch the system on or off.
- UPDATE to alter the settings.

Therefore, such a scenario could be applied also to a commercial environment like a hotel offering IoT application to the guests to change the room temperature while they are out of the room, so the hotel saves more energy.

Either IoT system was in a house or a business, it means that the sensors take some data as an input, process it, and store the output in a database (locally or in the cloud). Hence, offering an IoT services and provide a mobile application to the users and give them the permission to use their own personal devices to access and alter the data anytime is risky in case the wrong person accessed the database.

Some of the main challenges that related to IoT systems are:

- The security of the information: allowing the users to access and alter the data anytime and anywhere using any of their personal devices is risky. Because anyone of the user's partner, kids, or friends (who knows the signing in credentials of the application) can access, read, write, or update important information.
- The irresponsibility of the users: giving the users the permission to access and use the data could lead to complications if the user was careless. For instance, in the example of allowing the user to turn on/off the heating or cooling system, what if it was in a classroom and the students have an application to edit the temperature as they longing, the problem that would happen here is one student is cold and another is warm. In this case, they will keep changing the temperature, and this type of behavior could cause damage. In another way, the users' behaviors could minimize the security of the device that is used to access IoT services; like if they want to download forbidden software, they will change the security setting of the device, and that could cause problems.

This research is trying to solve the problem of helping to prevent an illegal entry to an IoT system by applying the proposed solution mainly in a low power device, which is Raspberry Pi2. By evaluating some of the users' attributes; specifically, accessed time, device's MAC address, username, and password to detect if the user who is trying to access the services is a permitted or not. Another issue is testing the system in an overload cases that includes different (delays, users, requests) to detect the response times (is the amount of time from the moment that a client sends a request to server including the time that the client receives the response and the entire request has completed).

CHAPTER 3

LITERATURE REVIEW

To build the proposed solution of this research, this chapter presents previous research work that related to the following concepts:

- Attributes Based Access Control (ABAC).
- Security challenges.
- RESTful services.

3.1 Attributes Based Access Control (ABAC)

Before focusing on ABAC, the general idea about Access control (AC) is introduced to have a better understanding. After a great deal of research, Hu et al. defined Access Control as “The logical component that serves to receive the access request from the subject, to decide, and to enforce the access decision” [4]. Another definition stated by Focardi and Gorrieri in [5] describing AC as the process that controls every entry to a system and its resources and allowing only the authorized requests to access the data. Additionally, Musa explained in [6] that AC approaches to guarantee the protection of the data from unauthorized expose or alteration. Besides, access control methods control how users work together with data and other network properties. AC has different types of models each has its advantages and disadvantages; still, many developers use them and realize their flexibility, especially in IoT domain. For instance, [7] and [8] combine two types of AC methods like Role-Based Access Control (RBAC) and ABAC methods to get the best of the two models.

Following, is an overview of some of the Access control models:

Mandatory Access Control (MAC): In MAC, the system only decides which subjects can access particular data objects. The MAC model based on security categories. For example, each subject has a security classification such as (secret, top secret, confidential, and so on); also, each object has a security classification (secret, top secret, confidential, and so on). All security categories stored in the system, so every time a subject request access to an object, the system

compares between the subjects' and objects' classifications. If they are equivalent, the subject will be granted the access. MAC based application used mainly in military systems and Department of Defense (DoD) since the 1960s.

Discretionary Access Control (DAC): In DAC, the access to the object is specified by the object's creator. In this model, the control of access is based on the discretion of the object's owner; hence, it is called discretionary Access Control. Some examples that based on DAC are Most of the operating systems like all versions of Windows, Linux, and Macintosh. In these operating systems, when a user creates a file, he or she chooses the access privileges that allowed to other users. Therefore, when a user accesses a file, the operating system will permit or decline the access based on the access privileges that the creator of the file indicated.

Role-Based Access Control (RBAC): Robert Kuhn and David Ferraiolo first formalized RBAC in 1992. RBAC is a model that makes the decision of permitting the users the access to a computer or a network resource based on the roles of a particular user within an enterprise (users' job titles). For instance, an employee from the natural resources' department in a company cannot access, create, or modify a file in the Information Technology (IT) department. Also, a worker in the management department has more access privileges than an operative in the analysis department has. Furthermore, Hu et al. explained more about the RBAC by saying that the "access is implicitly predetermined by the person assigning the roles to each individual and explicitly by the object owner when determining the privilege associated with each role" [4].

The previous models are the oldest and most known of AC models; however, there are more types of models used in different domains. For instance, Organization-Based Access control (OrBAC): it is a model that "allow [s] the policy designer to define a security policy independently of the implementation" ⁹. As well, Identity-Based Access Control (IBAC) based on approaches like access control lists (ACLs); so if a user inserts a credential that matches the one stored in the ACL, the system will grant the user the access to the requested object.

After having an idea about the Access Control and some of its models, now focusing on ABAC because it is the one which is applied in this research. Many developers defined ABAC based on their researching and understanding of it such as what Yuan and Tong stated that ABAC

⁹ http://orbac.org/?page_id=21

is a model “based on subject, object, and environment attributes and supports both mandatory and discretionary access control needs” [9]. Another definition said that ABAC “grants accesses to services based on the attributes possessed by the requester” [10]. Also, Hu et al. had their own definition of ABAC, which is “An access control method where subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions” [4]. To summarize ABAC previous definitions and more, when a user requesting the access to a specific data, the system must apply policies and conditions on the users and/or the object and/or the environment attributes. If the attributes meet those conditions, the system will allow the user the access to the data.

Attributes in ABAC could be considered as characteristics of anything (in the subject or object) that might be defined and to which a value might be assigned. ABAC depends on the evaluation of attributes of the subject, attributes of the object, environment conditions, and pre-defined policies (by the system originator) to allow actions for the subject. All ABAC applications include the abilities to evaluate the attributes of the subject, object, and environment. Furthermore, the subject should not be able to change his or her own authorization attribute value because only security authorities (system creator) should be able to provide and declare the attributes, and attribute values based on authoritative permissions of the object [4].

Some of ABAC advantages, which make it one of most important AC models and an appropriate model for this research:

- In ABAC, the entities such as the user’s name, password, or location are called attributes, and there are policies must be applied to those entities. The attributes are changeable, yet the policies are static; thus, ABAC is different from other models because it has evaluating polices to be applied on the subjects’ attributes; depending on the results the subject will grant or denial the access to the object.
- ABAC is best used in situations where users are dynamically changing.
- ABAC is a flexible model since it does not need to define a prior relationship between the subject and the object. In case the object owner wants to change the access decision, simply he or she needs to modify the access policies or the attributes’ values.

- ABAC provides the object owner or the organization manager a great ability that he or she can create an access control policies without knowing the subjects. For example, if a hotel manager offers an application to their customers, he or she does not need to know all the guests to apply ABAC policies. Consequently, any new guest checks in, the access control rules or objects do not need to be altered. And that consider a great and unique ability for ABAC “Because all the existing access control models are for known user access except ABAC, and only ABAC can realize the unknown user access” [7].

Next, some researchers suggested ABAC models for an IoT system. For example, Kaiwen and Lihua introduced in [7] a model that combines two AC methods, RBAC and ABAC because they wanted to assign roles for each user. They created the term Attribute-Role-Based Hybrid Access Control to describe their system; the idea for IoT system is to evaluate a number of users’ attributes every time the user sign in, then depending on the evaluation result the system will apply a specific role to each user with particular permissions.

The second model is called CollRBAC: IoTCollab Role-Based Access Control Model presented by Adda et al. in [8], like the previous work, it has the idea of joining two AC mechanisms in one system the main different is that in CollRBAC the system uses RBAC as an authentication step. After that, they introduce ABAC as a way to assign the permissions for each role.

Other developers used ABAC as an encryption technique, and it is known as Attributes-Based Encryption (ABE). For instance, Wang et al. research work in [11], Their system implementing the basic ABE process from generating keys to encryption to lastly a decryption. In details, “An ABE system usually consists of a key authority, publishers [senders] and subscribers [recipients]. The key authority authenticates publishers and subscribers (verifies they are who they say they are, as well as their attributes), generates public/private keys, and issues the keys to publishers and subscribers,” [11].

To sum up, IoT systems give the users a smart experience, and that why they are becoming popular to the people. Also, to achieve that IoT systems rely on several sensors, and a server to store, process, and retrieve the data. Hence, to protect these data from unauthorized access, IoT systems developers need an AC model that flexible and dynamic with a vast number of subjects

and does not require a prior relationship between the subject and the object. As shown in this section ABAC has most the potentials to apply an Access Control mechanism on IoT system generally and on this research specifically.

3.2 Security challenges

Because of the improved technologies like blending IoT systems with smartphones and the use of the cloud computing as a sufficient way to store data and information, the hackers becoming more knowledgeable and building more complicated software to attack. Therefore, the traditional threats such as malicious software and viruses are very different from the past; they are invisible and more dangerous, and they could cause significant damaging outcomes.

The next statistic displays the categories of cyber-crime attacks that frequently experienced by corporations in the United States. Through a 2015 study of 58 U.S. businesses, it was found that 97 percent of the participants had experienced malware attacks. The most common kind of attacks were viruses, worms, and Trojans.

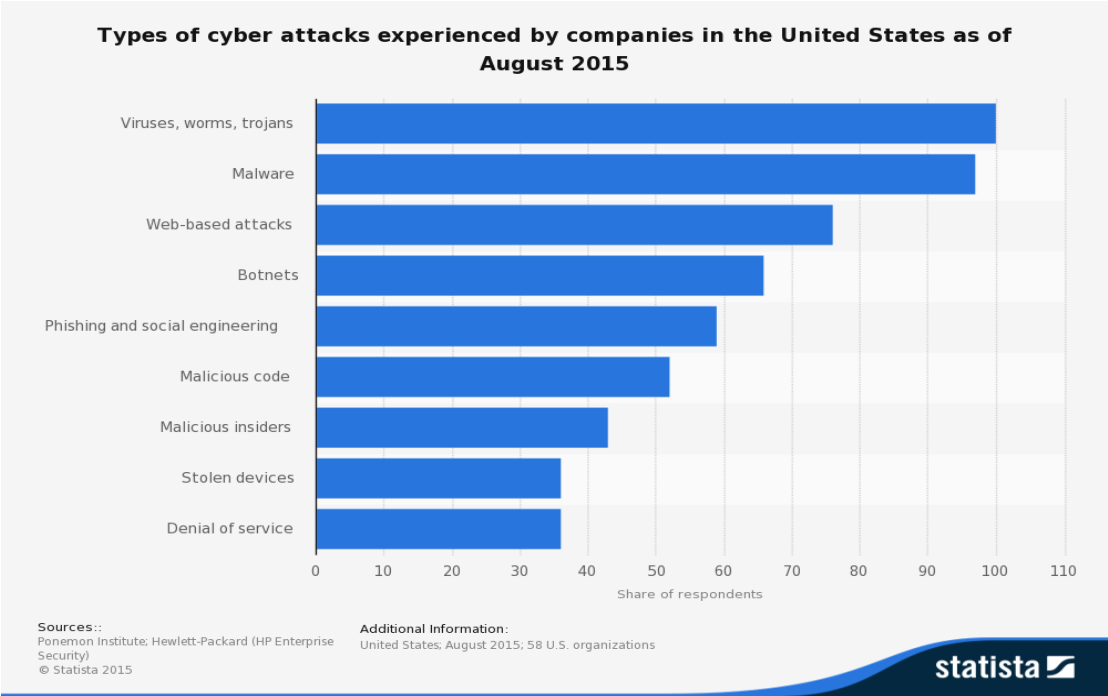


Figure 3-2-1. Types of cyber-attacks experienced by companies in the United States as of August 2015¹⁰

In this section of the literature review, some of the security concerns that other developers declared or concluded from their research will be presented.

To begin with, [12] revealed some of the security risks that could happen in each layer of IoT structure:

1. **Perception Layer:** With all the available technology, hackers could spy on a communication using software like malware. The attackers aim at this layer because they know it is the fundamental part of IoT (this layer's basic function is to perceive and collect information). Therefore, attackers can easily obtain and explore the data from this layer to capture the users' information, and that might cause excessive damages.
2. **Network Layer (transport layer):** One of the most known features of IoT is its vast amount of data. Thus, this layer is primarily responsible for transferring data between the Internet and the mobile telecommunication network and so on. Because an enormous amount of data is transmitted between the perception layer and the application layer, the network layer will unavoidably generate enormous numbers of redundant data. Consequently, the developers must add the filtration devices between the network layer and the application layer to avoid service attacks and leave the network unblocked.
3. **Application Layer:** This part achieves the main goal of developing IoT system, which makes the users' lives smarter and decreases the workload. Specifically, it processes the data logically so the users can use it to provide real-time information. The biggest security challenge in this part is protecting the users' data, which includes their confidential information.

Furthermore, the second implementation in this research used a cloud-computing platform to store the data and the rules of ABAC because nowadays most the applications including IoT store the data in the cloud. Cloud Computing (CC) generally means the procedure of storing or managing

¹⁰ <https://www.statista.com/statistics/293256/cyber-crime-attacks-experienced-by-us-companies/>

data in remote servers that is hosted on the Internet. Bahar et al. mentioned in [33] some of the services that provided by the Cloud Computing which helps in the growth of CC:

- Software as a Service (SaaS) it is a service that allows the users to use diverse applications running on the cloud infrastructure using client interface like web browsers, yet the customers do not have any control on the underlying cloud infrastructure, servers, operating systems, storage, or networks.
- Platform as a Service (PaaS), and it gives the consumers a platform to work with. It allows them to create applications using programming languages; however, the users cannot control or change the underlying cloud infrastructure, servers, operating systems, storage, or networks.
- The third service is giving the customers computer infrastructure, and it is known as Infrastructure as a Service (IaaS); with this service the users have full control over the applications, servers, operating systems, storage, and networks, but they cannot control the underlying cloud infrastructure.

With these services, Cloud Computing has significant benefits to all users with different needs. For example, extending battery lifetime of the mobile device (in case it was used as client) which considered being one of the obstacles of mobile devices. Dinh et al. did some experiments in [29] and the results showed that there is a different in the battery lifetime when the mobile devices were connected to the CC due to most of the computation operation were done in resourceful machines like servers in the clouds. An additional advantage was mentioned in [29] is improving another concern in IoT applications which is data storing capacity. CC offers the users the ability to store data in the cloud over wireless network. For instance, image sharing services give the user the option to upload their photos to the cloud and access it from any device that connected to the Internet to save the memory storage on the user's device; Flickr and Facebook are great examples of applications using CC to store images.

However, there are some risks that related to (CC), which could be possible threats for the data security:

- Cloud Computing offers enterprises a way to move openly available information out of their private computing structure because cloud computing is internet-based service [13]. Having all kind of information such as personal information obtainable

to the customers, employees, or other partners consider dangerous because of the hacking knowledge and malware.

- With the massive development of CC and IoT, hackers are also growing more knowledgeable and have a bigger motive to breach other people’s devices. “They could also intercept passwords, steal credentials, collect sensitive personal or corporate data, or install malware to take over devices,” as Leavitt noted in [14].

Furthermore, Armerding mentioned in [15] 15 of the worst data security breaches in the 21st century in which customers registered their personal information so they could benefit from the business’s services (See Table 3-1).

Table 3-1. Four of the worst data security breaches in the 21st century.

Affected Company	Date	Impact
Sony's PlayStation Network	April 20, 2011	<ul style="list-style-type: none"> • More than 77 million accounts were hacked, and 12 million accounts had unencrypted credit card numbers. • Sony said that they lost millions while the site was down for a month. • Cost was 14 billion yen = \$171 million
Target Stores	December 2013	<ul style="list-style-type: none"> • Credit/debit card information and/or contact information of up to 110 million people were compromised. • The cost of the breach was \$162 million.
Home Depot	September 2014	<ul style="list-style-type: none"> • Theft of credit/debit card information of 56 million customers. • The breach had cost it \$33 million.
Anthem (The second-largest health insurer in the U.S.)	February 2015	<ul style="list-style-type: none"> • Theft of personal information on up to 78.8 million current and former customers. • The total cost of the breach is predicted to be over \$100 million.

The conclusion from this section is that CC and IoT systems are targets for the hackers to acquire imperative and sensitive information. Moreover, reading about IoT systems' layers leads to the understanding that the application layer often has the intelligent and private information which could have a lot of risks in case of unauthorized access.

3.3 RESTful web services

The Web is a universal ecosystem to apply all kind of operations on applications and services; it makes us able to search, transfer, cache, replicate, and much more. One of the biggest factors that made the Web essential is people; "human users are the direct consumers of the services offered by the majority of today's web applications" [16]. Applications including the ones in IoT or Cloud Computing are a type of distributed systems because they have the ability to connect numerous devices in a network. Moreover, there are enterprise applications; Webber et al. defined them in [16] as a computer written software to fulfill the necessary requirements of an institute rather than the discrete users. Whatever a domain that the distributed system is a part of, the system's developer needs to send and receive data between two sides the client such as a web page or a mobile application and the server whether it was locally or in a cloud, and he or she could accomplish that by using web services. According to Dospinescu and Perca, "Web services are a solution for the integration of distributed information systems, autonomous, heterogeneous and self-adaptable to the context" [17]. There are different types of web services like SOAP, XML, and REST; each has strength and weakness points.

Moreover, [17] declared the components of the web services eco-system:

- **Discovery:** which is defining the location of the web service that would attach.
- **Description:** the client accepts a description of how the connection would be accomplished.
- **The format of the messages:** this component is important for the encoding stage.
- **Encoding:** permit processing by any language, one of the most known is XML.
- **Transport:** the way that the data will be conveyed between the client and the server. For instance, HTTP, FTP, or CoAP.

In this research, the focus will be essentially on RESTful-based protocols because it is one of the most popular web services for its structure that can construct large-scale distributed hypermedia systems. The REST architectural is grounded on four principles [18]:

- **Resource identification:** Uniform Resource Identifier (URI) is the resource identifier “which provide [s] a global addressing space for resource and service discovery” [18]. Each RESTful Web service represents a set of resources that classify the requests between the servers and the clients in the interaction.
- **Uniform interface:** the systems operate any resources are using a static set of CRUD operations (Create, Read, Update, and Delete).
- **Self-descriptive messages:** resources are separated from their representation so that their content can be retrieved in a diversity of formats (e.g. XML, plain text, PDF, and so on). Besides, “Metadata about the resource is available and used, for example, to control caching, detect transmission errors, negotiate the appropriate representation format, and perform authentication or access control” [18].
- **Stateful interactions:** every interaction with an item is stateless; in other word, requests are self-contained. Stateful interactions are founded on the impression of obvious state transfer, which hyperlinks could achieve.

Christensen concluded in [19] that RESTful services improve the created application by surpassing the abilities of old-style smart devices. Over time, developers created protocols based on RESTful services, and they have great potential to build scalable connections due to the support for caching, clustering, and load balancing that are built into REST. All of this leads to a new generation of developing IoT applications with extraordinary potential.

This project is using two protocols based on RESTful structure:

1. CoAP in the first implementation which uses Single Board Computer (Raspberry Pi 2).
2. HTTP in the second implementation which is in a cloud platform.

➤ **HTTP vs. CoAP**

As HTTP is a long-standing successful protocol, it can use a small written-code to join several resources and services. Interoperation delivered by HTTP is the crucial point

of IoT, for this, HTTP is applied at the application level. However, HTTP is based on Transmission Control Protocol (TCP) using Point-to-Point (P2P) communication model that not appropriate for constrained devices, HTTP is considered complex for those devices [20].

CoAP is a network-oriented protocol; it permits low overhead, parsing complexity, and multicast. Unlike HTTP-based protocols, CoAP operates over UDP (User Datagram Protocol) instead of using complex congestion control. CoAP offers REST based operations like GET, POST, PUT, and DELETE. On the other hand, it is founded on lightweight UDP protocol, CoAP permits Internet Protocol (IP) multicast that fulfills the communication for IoT [20].

In summary, RESTful-based protocols considered being one of the most well-known and used protocols among other web services. CoAP is appropriate more in a compact environment such as SBC, the one used in the first implantation of the proposed solution. CoAP maintains M2M connections and sends the data in a binary form which makes it faster than HTTP that sends the content in human-readable form. On the other hand, HTTP is suitable more for web applications; it maintains P2P connections and confirms that a request is sent and a response is received. Which makes it a better choice in the second implementation of this research.

CHAPTER 4

SYSTEM ARCHITECTURE AND IMPLEMENTATION

With the growth and increasing usage of technology, the risks and threats are also growing. Today's systems including IoT systems store an enormous amount of private information about the users. Moreover, the process of storing data has changed; we now have cloud computing to do the job without needing paper or servers to maintain records. Moreover, IoT is now used everywhere, thanks to all the capabilities it offers such as sensors or Machine-to-Machine (M2M) services. Hence, the private information in organizations has become a target for malicious people. As a result, the number of security incidents and malware is rapidly increasing.

Nielsen believed that "Threats today are designed to be largely invisible, blending in with background noise and traffic, requiring a form of contextual-based security analytics to detect and identify them" [21]. Because the threats that target enterprises can be dangerous and subtle, developers need to think of new ways to protect and secure private information; one of these new ways is analyzing user context.

The main objective of the proposed solution is to prevent any unauthorized access to an IoT system. This research is working on the ability to apply one of the access control types—ABAC. By evaluating some of the users' attributes such as access time, device, username, and password, the system will apply ABAC policies to make sure that the user who is trying to access the data is authorized to do so. To use a metaphor, the system is compared to a castle, and I want to protect it from risky people trying to get inside it. Consequently, the first thought is to put guards on the front gates, but that will not be enough; the castle needs more guards on each gate. Therefore, a system using only a password as a security technique is like the castle with guards only at the front gates. Hence, this project is implementing ABAC to an IoT system to evaluate the attributes of the user. For instance, when the user accesses the data which device is used to sign in? The system applies some policies on each of those users' attributes, and if they do not fit, the system denies access.

The main idea of the proposed solution is to add a layer of security, which can evaluate some of the users attributes to indicate if the user trying to access the database is authorized or not. Fig. 4-1 and Fig. 4-2 present the main idea of the proposed solution:

1. With an SBC (Raspberry Pi 2), which accesses the database locally.

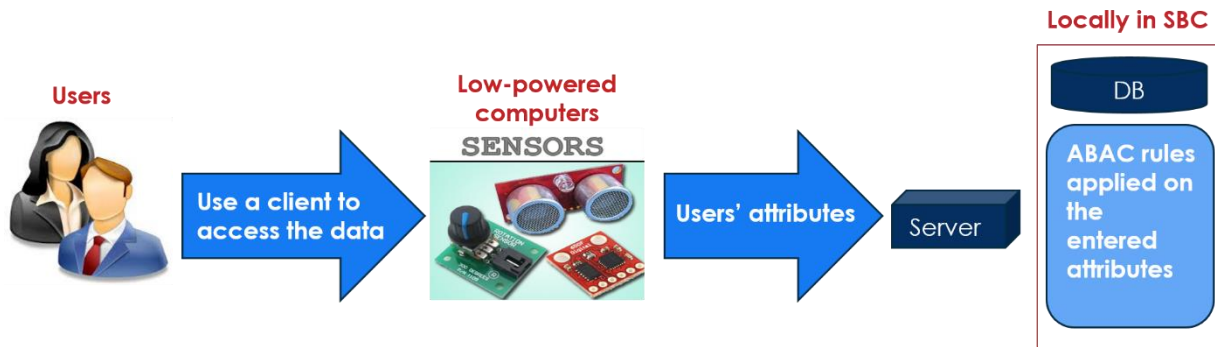


Figure 4-1. First proposed solution

2. With an SBC (Raspberry Pi 2), acting as a proxy which connected to cloud computing.

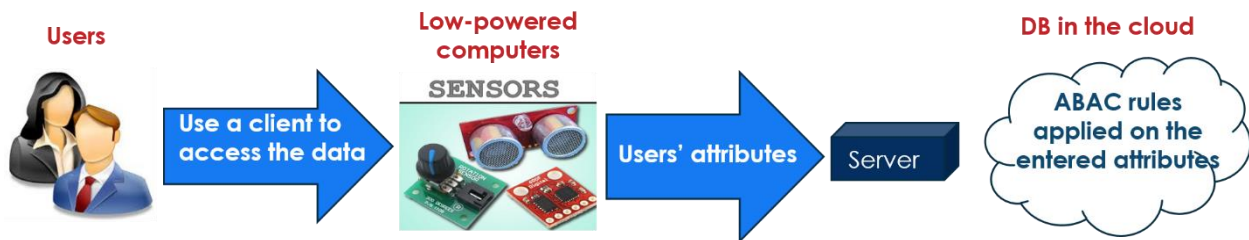


Figure 4-2. Second proposed solution

In both the scenarios, the implementation will be the same. It will apply ABAC policies on some of the user attributes, including the credentials in the first layer of authentication (username and password).

Two levels of security were built to achieve the proposed solution:

1. The first level is the client side. It is retrieving some of the user context, such as username and password, from the database in the cloud.
2. The second level is created on the database which creates the functions to store and evaluate some of the users' attributes, such as location and time.

When a user who has the application signs in with his or her username and password, these attributes, along with another context, will be sent to the server side, where all the information is

stored on a database to determine if the user should access the company data in the cloud or not. The system applies some policies on the users' context, depending on each user's attributes stored in the database.

To create the proposed solution, this research needs two implementations, and in each, there are two steps. The first step is to create the client side; in the first part is a Web page and in the second part is a mobile application. The second step is to build the server side of the system. In both parts, one will be local and the other is in the cloud. Furthermore, the system connects the client side with the server side in both parts using RESTful-based protocol.

4.1 First implementation

This is where the real contribution of this research is applied. The client, the server, the web services, and the database were built in a low-power device, which is a type of SBC (see figure 4-1-1).



Figure 4-1-1. Single Board Computer (Raspberry Pi 2)

In this case, Raspberry Pi 2 represents an IoT low-power device such as a sensor. Figure 4-1-2 explains the first implementation.

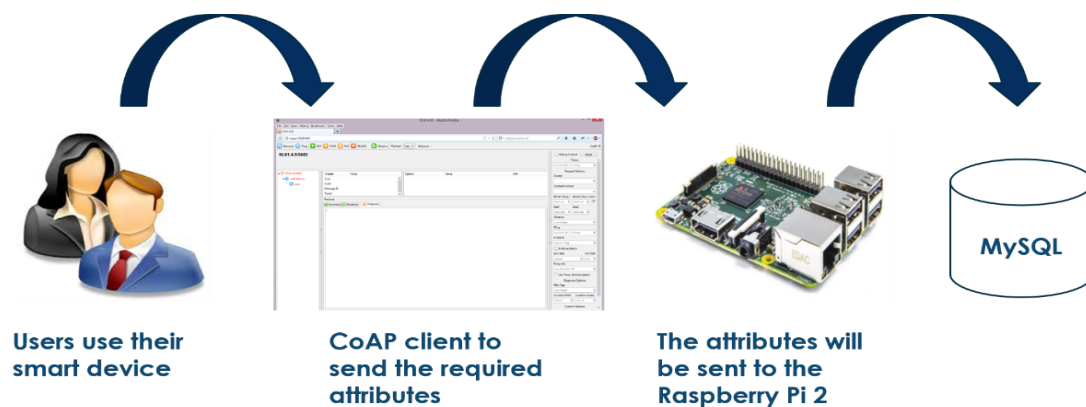


Figure 4-1-2. First implementation

4.1.1 Client Side

First, to build a client prototype for the SBC implementation, a Web page was created using HyperText Markup Language (HTML), a foundation technology used to build Web pages as well as to construct user interfaces and Web applications. A markup language is a set of markup tags, and each HTML tag implements different functions; Web browsers can read and execute HTML files. HTML was chosen to write the client prototype in this implementation for following reasons:

1. The easy syntax of the language.
2. It is faster to run the system locally.
3. HTML uses friendly and clear interface.

HTML was used to take some attributes from the users: username, password, and the device's MAC address. After the user enters the required credentials, he or she will click an input button to submit the entered context. The form method was assigned to POST to send the information to the database where ABAC policies are stored to be applied on any entered user's attributes.

In the former approach, to connect HTML to the database on the server side, HTTP was used as a Web server. However, because a Single Board Computer (Raspberry Pi 2) is used, the solution was built as compactly as it could be; this is why system replaced the Web service from HTTP to CoAP. Shelby, Hartke, & Bormann stated, "One of the main goals of CoAP is to design a generic web protocol for the special requirements of this constrained environment, especially considering energy, building automation, and other machine-to-machine (M2M) applications" [22]. In order to use CoAP, the system had to change the client (the Web page) to CoAP client because HTML does not have a CoAP library¹¹ (see figure 4-1-3).

¹¹ <http://coap.technology/impls.html>

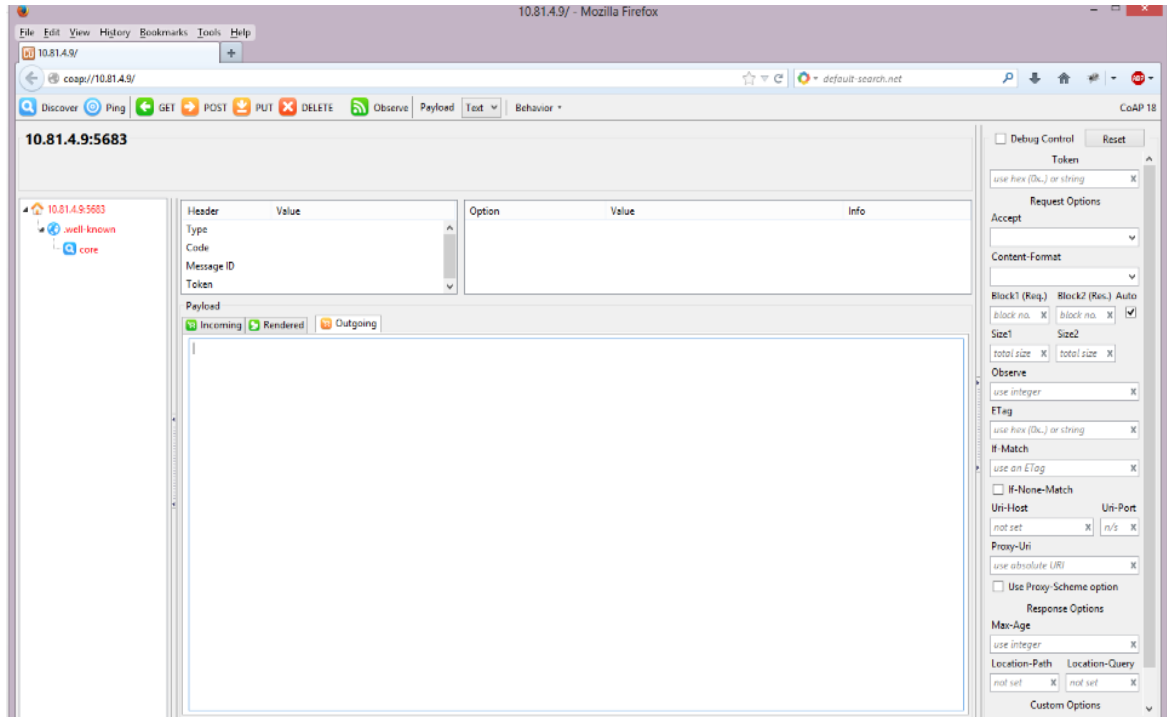


Figure 4-1-3. Copper (Cu) CoAP user-agent for Firefox web browser

Another thing the system executed just to try it in this research is building a web page in the SBC to send the required attributes to the cloud via a proxy. The proxy is a Go language (Golang) written code proxy that contains the cloud URL and it uses HTTP as web service to get the user context from the web page and send to the cloud. The objective of this thesis is to demonstrate that the system could have a client in Single Board Computer like Raspberry Pi 2, which connected to a server which exists in the cloud through a proxy.

4.1.2 Server Side

The server side in the SBC is divided into two sections:

- The proxy that handles sending the attributes to the database

The proxy was written using Golang because it has a good cross-platform support, concurrent and modern, and it has faster compiler than other languages like Java; also, it connects easily to MySQL¹².

¹² <https://www.quora.com/What-is-the-advantage-of-Google's-Golang-over-Java-and-Scala>

Initially, this proxy is created to get a specific user attributes (username, password, and MAC address) from the client Cu and send it to the database. Furthermore, a request handler function was created which apply CoAP protocol to get the required attributes from the users to send it to the database. To achieve that the entered attributes had to be parsed to the JSON (JavaScript Object Notation), then obtained the entered context in variables; subsequently, those attributes were sent to another function called POST. In POST, a connection with MySQL database was created. Afterward, an authentication step was written where the system applied a query to compare the username, password, and MAC address with the ones stored in the database.

Some parts of the Golang server code will be presented following:

```
// Define struct and variables that will be assigned to the
attributes of the users
    type attributes struct {
        Username string
        Password string
        Mac string
    }
    var (
        name string
        pass string
        mac string
    )
//Assign the entered attributes to the created struct above
    var user attributes
    name = user.Username
    pass = user.Password
    mac = user.Mac
```

- MySQL database that stored in the SBC (Raspberry Pi 2)

The database in this research contains number of tables:

- **Users table:** This is the main table because it is the one used for authentication, and all other tables are connected to this table using a foreign key. It includes

all the usernames and passwords of the clients of an enterprise that wants to provide an IoT application. Every time a user wants to sign in using his or her device, the username and password of this user will be compared with login credentials in this table to verify their authentication.

- **Mac_address table:** This table includes all MAC addresses of the users' devices that they are using to sign in. Worth declaring here is that a client can use more than one device. Each mobile phone or tablet has a MAC address depending on the network connection with the device. In this table, all the MAC addresses of each user's devices were stored, and they have the user_id from the users table as a foreign key. For example, if a user whose ID is 22 has two devices, and each has a MAC address, then both MAC addresses will be stored in this table with the user_id 22. This table is also used in the authentication step such as the usernames and passwords in the users table.
- **Logs table:** This table evaluates the context of each user who is trying to sign in. The login process starts in this implementation from the CoAP client. The system will send the username, password, and MAC address to the Golang proxy, which sends the attributes to the MySQL database to apply the ABAC. It starts with an authentication process, which compares these credentials to the usernames and passwords in the users table. If they exist, then the matching ID in the users table will be sent to the logs table. Correspondingly, when a client signs in, the new MAC address will be compared to the current ones in the mac_address table, and its ID will be sent and stored in the logs table to be evaluated. Moreover, this table includes a column called time, which is a timestamp to return the server time when the username, password, and MAC address entered the database. The server time was used because it does not change; even if the mobile device changed its time or location, the server time is unalterable.

After the required attributes' ID is entered, and the time stored in the logs table, the logs contains t_result, u_result, p_result, and mac_result columns that return the evaluation of the accessed time, username, password, and MAC address. The username results column has a trigger to set the word "YES" if the username ID was entered in the logs table from the users table; this

also applied to the password results and mac results column. The evaluation of the accessed time is defined in an SQL trigger, which takes the time when the username, password, and MAC address entered the server and compares it to the assigned ABAC policies to test the prototype:

- If the accessed time is between 8 a.m. and 5 p.m., the result is “Yes” because these are work hours.
- If the accessed time is between 12 p.m. and 1 p.m., the result is “Maybe” because this is a lunch hour.
- If the accessed time is after 5 p.m. and before 8 a.m., the result is “No” because these are after work hours.

A function procedural was constructed using SQL commands to return the evaluation and access control results of each user’s signing in operation to the application.

To send the attributes to the database the following function was created in the server code; it includes establishing a connection to the database and inserting the attributes after a simple authentication step.

```
// Create the POST function
func postdb () string {
// Create the connection to MySQL
    con, err := sql.Open ("mymysql", "employees/root/sweet123")
    defer con.Close ()
    if err != nil {
        log.Println (err)
    }
// Define the variables that will be send to the DB
    var uid,mid int
// Apply some of MySQL operations to send the attributes to the DB
using select and inseret.
    row := con.QueryRow ("SELECT id FROM users WHERE username=?
AND      password=?", name, pass)
    err = row.Scan (&uid)
```

```

    row = con.QueryRow ("SELECT user_id FROM mac_address WHERE
mac=?", mac)
    err = row.Scan (&mid)
    if err != nil {
    log.Fatal (err)
    } else {
    _, err = con.Exec ("INSERT logs (userid, mac_id) VALUES
(?,?) ", uid, mid)
    }

```

All the ABAC policies were implemented on the database side because having them on the application level would make IoT system vulnerable to hackers. Additionally, IoT application could be suitable on different platforms.

Finally, the POST function returns the word “Hello to you” in case the system sent the attributes to the database and the ABAC policies were applied on the entered context without any errors.

4.1.3 Webservices

In this implementation, CoAP was applied as web service; CoAP is RESTful architecture based protocol; therefore, its resources act as server-controlled constructs require an identified URIs. It is used in low-powered electronics devices in another word (nodes), to make them capable of connecting interactively together; also, it allows them the ability to connect to the Internet. The Internet Engineering Task Force (IETF) CoRE Working Group has started the regularization activity on CoAP in March 2010. “Like HTTP, CoAP is a document transfer protocol. Unlike HTTP, CoAP is designed for the needs of constrained devices” said Jaffey in [23]. Moreover, Jaffey added “CoAP runs over UDP, not TCP. Clients and servers communicate through connectionless datagrams. Retries and reordering are implemented in the application stack. Removing the need for TCP may allow full IP networking in small microcontrollers” [23].

Next, some of the CoAP features [22]:

- Support for URI and content type.
- Asynchronous message interactions.

- Low parsing complexity and header overhead.
- Security binding to Datagram Transport Layer Security.
- Web protocol accomplishing Machine-to-Machine (M2M) requirements in constrained environments.

Figure 4-1-4 presents the interactive structure model for CoAP; like an HTTP client/server model, it is a transfer protocol. However, CoAP has a bottom layer which is messages layer that has been created to handle UDP and asynchronous switching. The upper layer is request/response layer is mainly for the communication method and handles request/response messages.

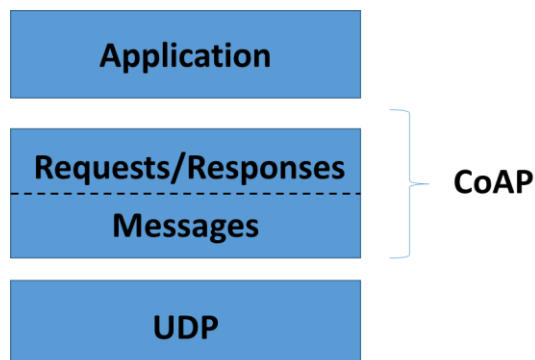


Figure 4-1-4. Abstract Layering of CoAP

The CoAP was implemented in the server code to achieve the goal of sending the attributes to the database (MySQL) installed on the SBC next is part of CoAP code:

```
// Create the connection to the CoAP port
log.Fatal (coap.ListenAndServe ("udp", ":5683",
    coap.FuncHandler (func (l*net.UDPConn, a*net.UDPAddr,
        m*coap.Message) *coap.Message
// Create the CoAP message
if m.IsConfirmable () {
    res := &coap.Message{
        Type:      coap.Acknowledgement,
        Code:      coap.Content,
        MessageID: m.MessageID,
        Token:     m.Token,
```

```

Payload:  [] byte ("hello to you!"),
}

```

4.2 Second implementation

In this part, a mobile application was created as a client, and the server-side was constructed in a cloud. Google cloud platform was used to store the database. Fig. 4-2-1 represents the main parts of this implementation; first, the users can use the mobile application (which was built and installed in Nexus 7 to complete the testing). Then, there is the interface, which requires a username and a password from the user, and they both considered being users' attributes. Lastly, all the attributes will be stored in the database (in the cloud platform) along with ABAC rules so the system applies them on each attribute.

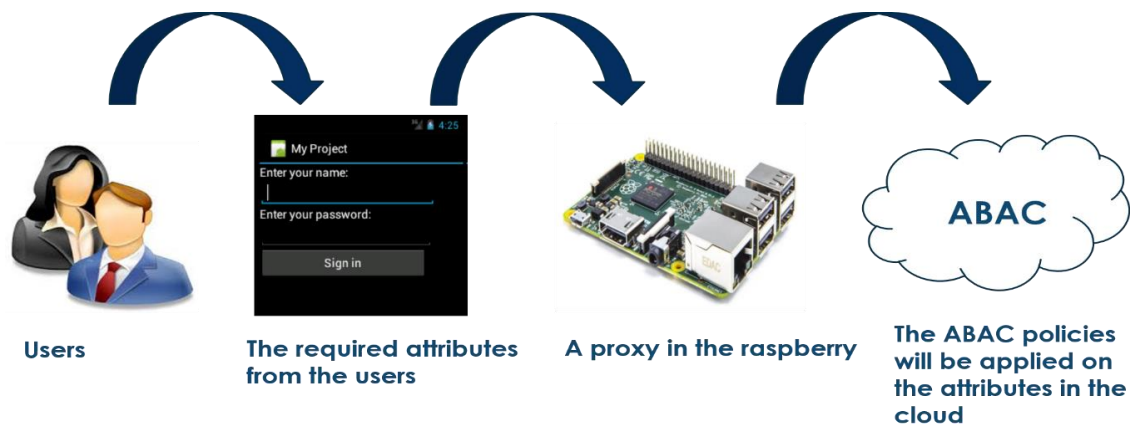


Figure 4-2-1. Second implementation

4.2.1 Client Side

The client side is the mobile application prototype; the system was tested on the local host first to detect the errors that are not considered connection errors, such as syntax mistakes. Next, the mobile application was built using Xamarin, which is a platform to create mobile applications. Xamarin was chosen due to following reasons:

- Its simplicity (all codes are written in C#).
- The developers can write codes for different platforms.

This research built the application for Android platform. The application had an interface (see figure 4-2-2) that contains:

1. A text field to enter the username.
2. A password field to enter the password.
3. A Sign in button (when it is clicked the username and password along with another context will be sent to the database in the cloud platform).

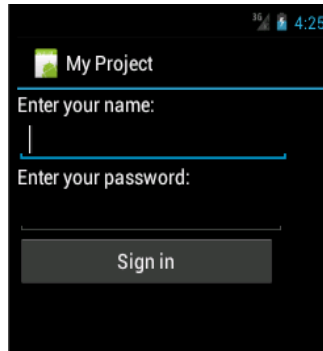


Figure 4-2-2. The Interface of the Application

The POST function code was written that sent the entered user attributes to the database in the cloud. To do so a public class was created to take the attributes to send it to the database table.

```
// C# code in Xamarin to represent the structure of the table
to be send to the database
public class t_context
{
    public string username {get; set;}
    public string password {get; set;}
}
```

The GET function code to retrieve some data from the cloud side is written on the client-side. In addition, a code in this client was written to recover the MAC address of the user's device. The MAC address was selected as another user attribute to be evaluated in the cloud.

HTTP POST and GET methods were used to send and retrieve the data from and to the mobile application, and the MAC address was sent to the server side as a header in the URL of the POST method. Moreover, a condition had been added to the application to reject any request that is not complete, such as username without a password and vice versa. The system must have all the attributes to apply ABAC policies.

This scenario does not include any sensor; it is simply sending the data entered by the user from the mobile application to the cloud. However, because this thesis focuses on using sensors like SBC, a proxy on the SBC was written using Golang, which takes the entered attributes from a client and sends them to the cloud. The client is Postman, an online request builder that is simple and can perform all HTTP operations and retrieve valuable information such as response time.

Some parts of the Golang proxy in the Single Board Computer (Raspberry Pi 2):

```
// Define the variables that will be assigning the attributes of
the users

    type authInfo struct {
        name string
        pass string
        mac string
    }

//Create the POST function using HTTP and grabbing the posted
attributes from the client

    func auth (w http.ResponseWriter, r *http.Request)

    if r.Method == "POST"

    r.ParseForm ()

    name: = r.FormValue ("name")

    pass: = r.FormValue ("pass")

    mac: = r.FormValue ("mac")

//Build a POST request for the Cloud

    form.Add ("name", name)
    form.Add ("pass", pass)
    form.Add ("mac", mac)
    req, _:= http.NewRequest ("POST", cloudaddr,
strings.NewReader (form.Encode))
    req.PostForm = form
```


4.2.3 Webservices

After creating the server and client sides, HTTP was used to connect the client to the server so they could share the data. HTTP structure is based on the principles of RESTful architectural style, and that clarifies the outstanding scalability of the protocol HTTP1.0 and HTTP1.1.

RESTful services have great qualities which make it easier to be used with the mobile platform [19]. Some of those abilities are:

- It diminishes the effect of network instability because REST is stateless.
- It is easy to invoke because it is URL based.
- It is distinct because its replies are frequently HTTP based.
- Its distribution can be completed concisely.

HTTP main methods:

- GET to retrieve the current state of an item.
- PUT to create or update an item.
- POST to transfers a new state onto an item.
- DELETE to erase an item.

This project is URL-based, so it is suitable to apply the HTTP operations. Moreover, it is more convenient when changing and editing the data and the syntax because each operation has its own URL. On the client side, an HTTP Web request sends and retrieves the data because each HTTP Web request needs a URL, content type, content length, the method to be applied, and some headers when needed.

In this prototype, to send the entered username, and the entered password, the HTTP Web request was assigned to the POST URL that was established in the POST function on the server side. In addition, a header was added to the request for the MAC address of the user's device to be sent along with the username and the password in the URL. Another HTTP Web request was created for the GET function; the GET URL to retrieve the data was also created on the server side.

PUT and DELETE do not apply in this design because there is no need to destroy any users' data or modify them.

CHAPTER 5

4.2.2 Server Side

The server side is the cloud side where the system stores all the information about the application or the user. The cloud includes the company's delivered services to the users, and each user must access the service he or she needs to use their IoT application. In this research, after the user signs in using the client side and some of his or her attributes, those attributes will be sent to the database to get evaluated (applying ABAC policies) on the server side.

In this design the server is using Google Cloud Platform, it is (PaaS) Cloud Computing platform developers use to host their Web applications. It gives four language options (Python, Java, PHP, and Go) to build the server side. This project used Python because of its simplicity and because Python is considered a strong language with a readable syntax.

There are two parts in the server:

1. **Google App Engine¹³**: This is the local side of the server. In this side, the Python code is to connect to the database in the cloud. This part acts as the middleware between the client and the database; here is where the URLs was created to transfer the user context (username, password, and MAC address). The Python code for this research contains GET function to retrieve the data that is stored in the database, and a POST function to Transfer the user context from the client side (HTML Web page or mobile application) and save it in the database. Each function has its own URL to use in the Web service.

In the POST function, the username and password authentication step were written. In details, when a user logs in using his or her username and password, this information will be sent to the POST function in the POST URL. However, before granting the user the access, the system makes sure those login credentials are stored in the database. If these data were in the database, they will be evaluated along with

¹³ <https://cloud.google.com/appengine/docs>

other attributes. Moreover, an authentication code was built for the MAC address in the POST function to make sure that the MAC address exists in the database.

The GET function acquires specific data from the database after the evaluation step of the user attributes is finalized. In this system, applying the ABAC policies occurs in cloud side (database) using MySQL triggers; to get some information after that, a MySQL stored procedure was constructed to return just a simple sentence to the user. For instance, if the user attributes authentication and evaluation results were acceptable by the system, the stored procedure will print, "Access allowed". The stored procedure is called in the GET function, and this GET function URL is called in the HTTP GET method on the client side (mobile application).

GAE has an interface that allows the user to test the application on a local host port or deploy the application to the cloud, and the application developer can access it using a given URL from the cloud.

```
//Create the class and the connection to the database instance in the cloud using Python programming language.
```

```
class Context (webapp2.RequestHandler):
    def post (self):
        user = self.request.POST ['name']
        passwd = self.request.POST ['pass']
        mac_address = self.request.POST ['mac']
        if (os.getenv ('SERVER_SOFTWARE') and os.getenv
            ('SERVER_SOFTWARE').startswith('GoogleAppEngine/'
            )):
            db = MySQLdb.connect (unix_socket='/cloudsql/' +
                _INSTANCE_NAME, db='', user='', passwd='',
                charset='utf8')
        else:
            db = MySQLdb.connect (host='173.194.81.103',
                port=3306, db='', user='', passwd='')
```

2. **The cloud SQL¹⁴:** This is the part of operating in the Google cloud. Worth mentioning here is that the Google cloud platform offers different options to store the data, such as cloud storage for formless data storage, cloud data store for NoSQL data storage, and Google Drive for users to store their personal files. For the proposed solution, Cloud SQL was a good choice because the data in this research work is related, so a relational database is an appropriate choice. Moreover, to access the database and apply the operations on the data, the system only uses the tools, which already existed locally in the MySQL. All the tables, ABAC policies, and evaluation steps were explained in the server side in the first implementation (figure 4.1.2).

```
// create the connection to MySQL

    cursor = db.cursor ()

    cursor1 = db.cursor ()

//Use Python code to insert the data to the database by
matching the id of the users

    cursor.execute ('SELECT id FROM users where username=%s
AND password=%s', (user, passwd))
    cursor1.execute ('SELECT user_id FROM mac_address
where mac=%s', (mac_address))
    id = cursor.fetchone ()
    u_id = cursor1.fetchone ()
    if not (id or u_id):
        return "Not Found"

    else:

        cursor1.execute ("INSERT logs (userid, mac_id) VALUES
(%d, %d)" % (id[0], u_id[0],))
        db.commit ()
        db.close ()
```

¹⁴ <https://cloud.google.com/sql/docs/introduction>

EVALUATION

The evaluation main goal was to test both implementations applying ABAC in case of a request heavy load. This experiment does not include any security attacks scenario. It is only to obtain the response times when different numbers of requests sent from several numbers of users in different delays (table 6-1).

Table 6-1. Evaluation plan

Delays	125ms	250ms	500ms
Requests (r) per users (u)	100r per 5u	100r per 5u	100r per 5u
	50r per 10u	50r per 10u	50r per 10u
	25r per 15u	25r per 15u	25r per 15u

This evaluation was applied on both implementations to obtain the response times according to the previous table. It is worth noting that all the X-axis in all the charts represents the requests number, and all the Y-axis represents the response time.

5.1 First implementation evaluation results

To evaluate the first implementation of this research, an SBC was used, which is Raspberry Pi 2 (RPi2) Model B Quad-Core 900 MHz 1 GB RAM. The kit contains CanaKit 2.5A Micro USB Power Supply with Noise Filter (UL Listed) specially designed for the Raspberry Pi 2. Additionally, 8 GB MicroSD Card and CanaKit Wi-Fi Adapter are included.

In this implementation, the client, server, and the database all exist in the SBC.

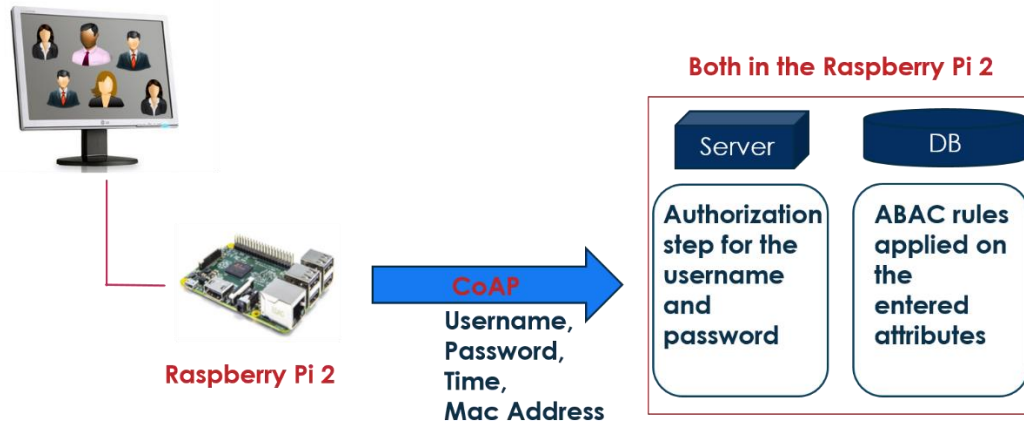


Figure 5-1. The evaluation setup for the first implementation

1. When sending 100 requests per five users in three different delays (125ms, 250ms, and 500ms)

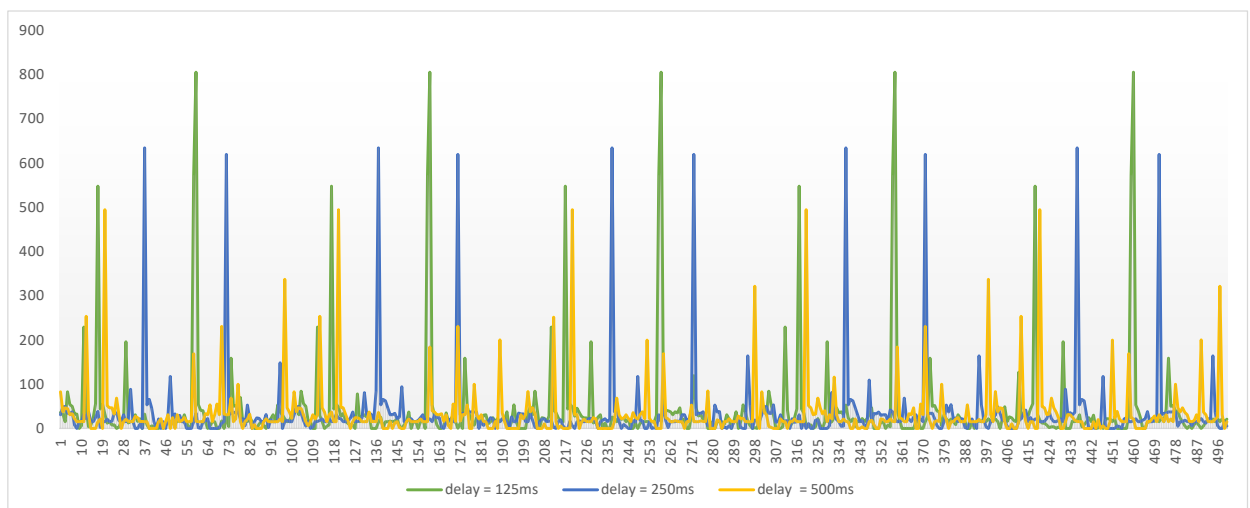


Figure 5-1-1. The response times when 100 requests sent per five users

The fastest scenario was when the delay was set to 500ms (the yellow line), its highest response time reached about 500ms. The slowest response time was when the delay was set to 125ms (the green line), its response time reached more than 800ms. Also, the last scenario's highest response time was a bit more than 600ms (the blue line) when the delay was 250ms.

2. When sending 50 requests per 10 users in three different delays (125ms, 250ms, and 500ms)

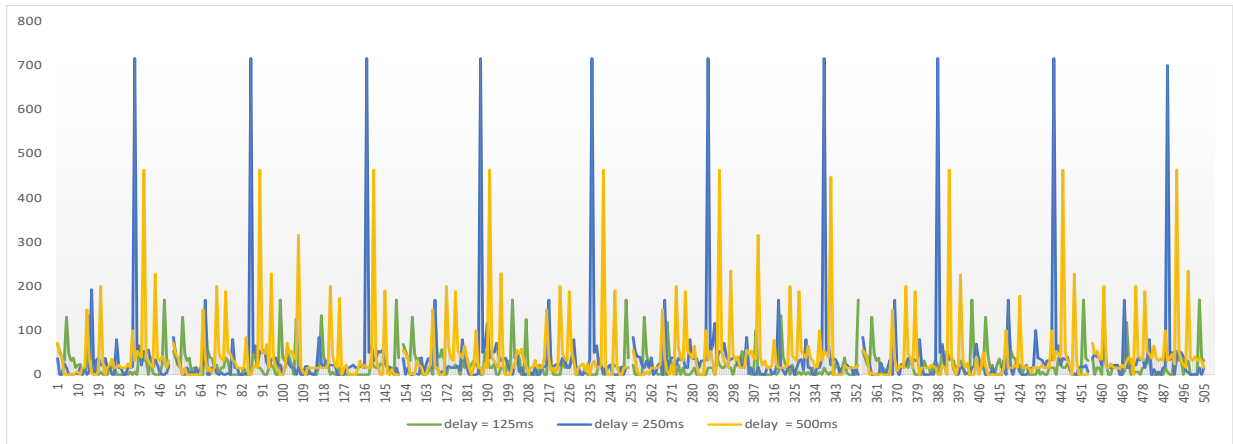


Figure 5-1-2. The response times when 50 requests sent per 10 users

The slowest response time was when the delay was set to 250 milliseconds (the blue line); its response time reached more than 700ms. However, the fastest scenario was when the delay was set to 125ms (the green line), its highest pick was in less than 200ms. While the third scenario’s response time (the yellow line) was in-between with a little more than 400ms as the highest response time.

- When sending 25 requests per 15 users in three different delays (125ms, 250ms, and 500ms)

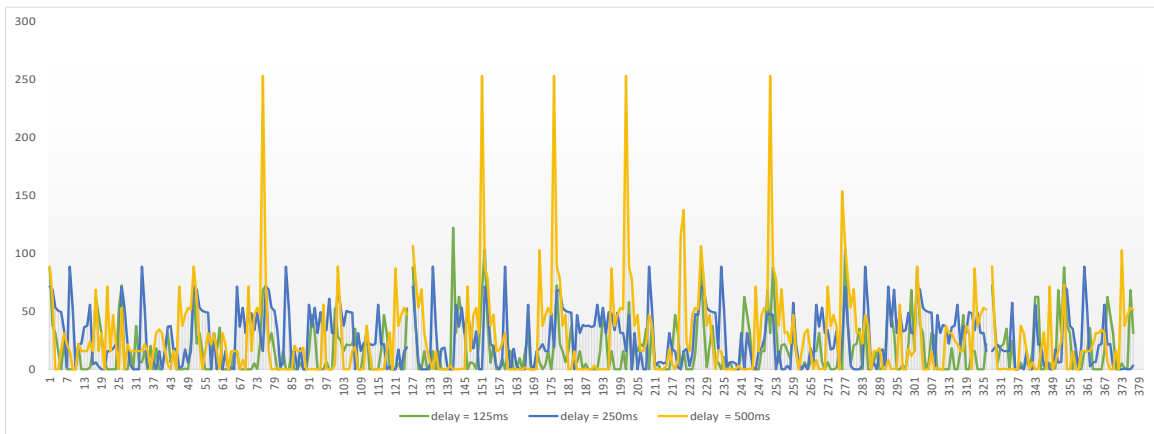


Figure 5-1-3. The response times when 25 requests sent per 15 users

The slowest response time was when the delay was set to 500ms (the yellow line), its response time reached 250ms. The fastest scenario was when the delay was set to 250ms (the blue line), its highest response time reached less than 100ms. Moreover, the last scenario’s highest response time was a little more than 100ms (the green line) when the delay was 125ms.

In general, (1) and (2) charts in all the assigned delays of the previous evaluation results, the total number of requests is 500 requests because in (1) we have five users, each sends 100 requests, and in (2) 10 users each sending 50 requests. In chart number (3), there are 15 users each sends 25 requests so the total of requests is 375. Hence, chart (3) in all the delays has the fastest response times (round-trip times). On the other hand, (1) and (2) have slower response times because they have more number of requests.

Sending the requests in all the delays was relatively fast; however, we can see in the charts that the response times have an increase suddenly. In my opinion, the reason is that the database and the ABAC policies held in the SBC along with the server. Thus, the users sending the requests and the systems applies ABAC rules simultaneously, and that takes some time and power from the CPU of the sensor.

Moreover, when analyzing the response time charts in each delay it was clear that each user got the same amount of time, which makes sense because there is no network delay due to the whole solution (first implementation) was held locally in the Raspberry Pi. Also, the fact that this project used Raspberry Pi 2 not an older model (like Raspberry Pi Model B+) helped enhance the performance with all the changes in the new model; for example:

1. The Raspberry Pi 2 has 4 processors in one chip (the B+ has only one)
2. The Raspberry Pi 2 has ARMv7 core (the B+ has an ARMv6 Core)
3. The Raspberry Pi 2 has 1 Gig of RAM (the B+ has 512 MB of RAM)

Reference [24] predicted the improvement of the performance in Raspberry Pi 2 by stating that “While it strongly depends on what you're doing, you should see at least 85% improvement” based on the three improvements were mentioned earlier.

In addition, the X-axis in all the charts represents the requests number, and the Y-axis represents the response time.

5.2 Second implementation evaluation results

To test the second implementation, a simulation of the proposed solution was built and implemented on a mobile device as a client. An Android Nexus 7 was used with the following features: Android version 5.0.2 operating system, internal storage capacity of 32 GB, the memory capacity of 2 GB RAM, and its processor 1.5GHz Quad-Core Qualcomm Snapdragon S4 Pro, and

400MHz Adreno 320 GPU. However, to apply the evaluation plan a Java based tool was used to create the different numbers of users sending different numbers of requests parallel in a particular delay. Another reason is the Android version of the device which used here is 5.0.2; it has depended greatly on Graphics Processing unit (GPU)-accelerated performance since Android 4.0. However, the Raspberry pi 2 GPU was the same as the old version of Raspberry Pi (250MHz) [25].

In this implementation, the client sends the attributes to Golang proxy in the SBC (Raspberry Pi 2) to the database that exist in the Google storage platform.

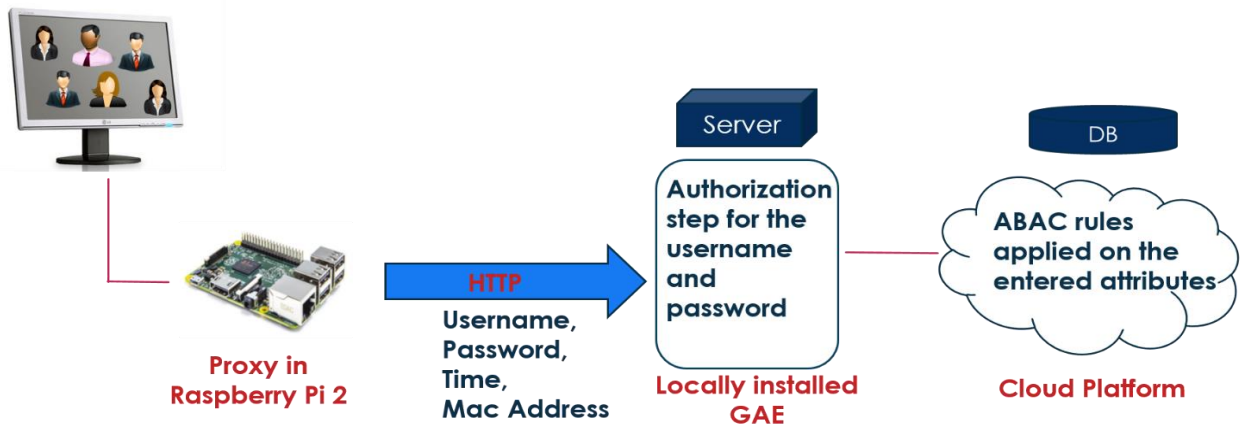


Figure 5-2. The evaluation setup for the second implementation

1. When sending 100 requests per five users in three different delays (125ms, 250ms, and 500ms)

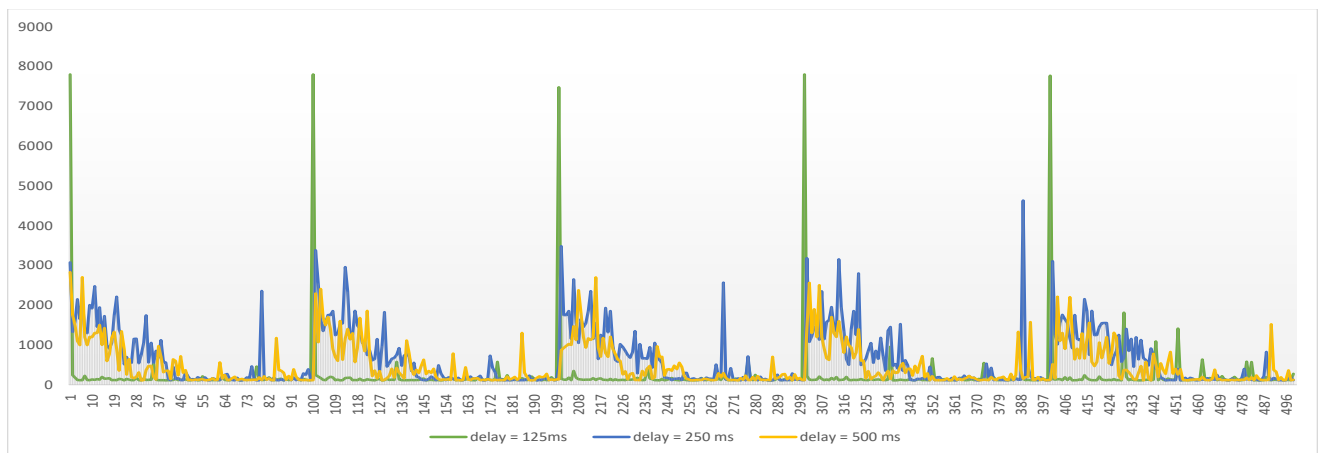


Figure 5-2-1. The response times when 100 requests sent per five users

The fastest scenario was when the delay was set to 500ms (the yellow line), its highest response time reached a little less than 3000ms. The slowest response time was when the delay was set to 125ms (the green line), its response time reached almost 8000ms. In addition, the last scenario's highest response time was about 5000ms (the blue line) when the delay was 250ms.

- When sending 50 requests per 10 users in three different delays (125ms, 250ms, and 500ms)

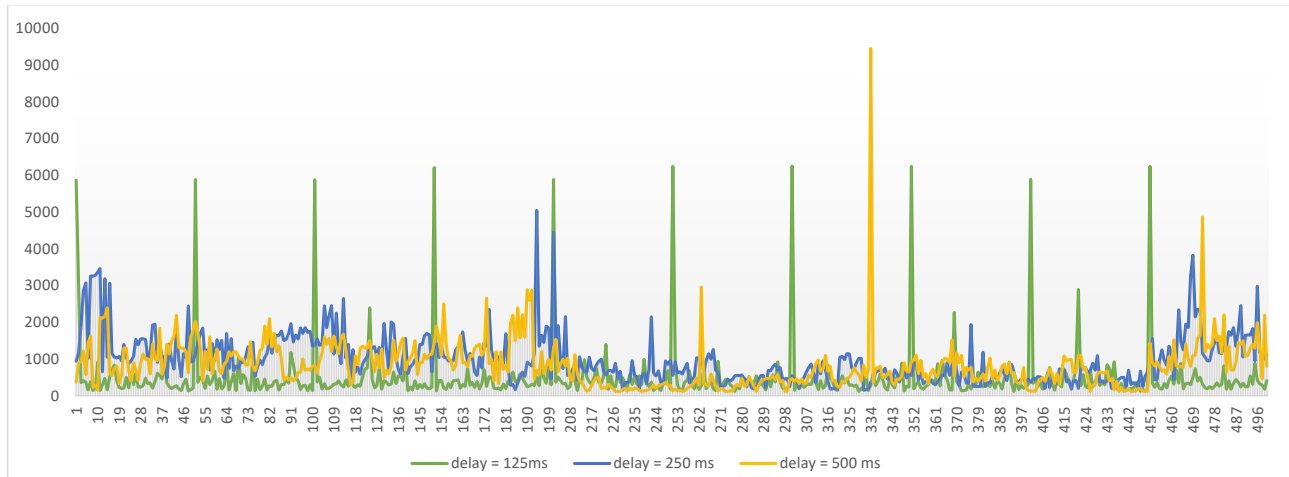


Figure 5-2-2. The response times when 50 requests sent per 10 users

The slowest response time was when the delay was set to 125ms (the green line); its response time reached less than 6000ms; this delay also has the most steady line. However, the fastest scenario was when the delay was set to 250ms (the blue line), its highest pick was about 5000ms. While the third scenario (the yellow line) interactions were relatively fast, but it had the highest response time which was more than 9000ms.

- When sending 25 requests per 15 users in three different delays (125ms, 250ms, and 500ms)

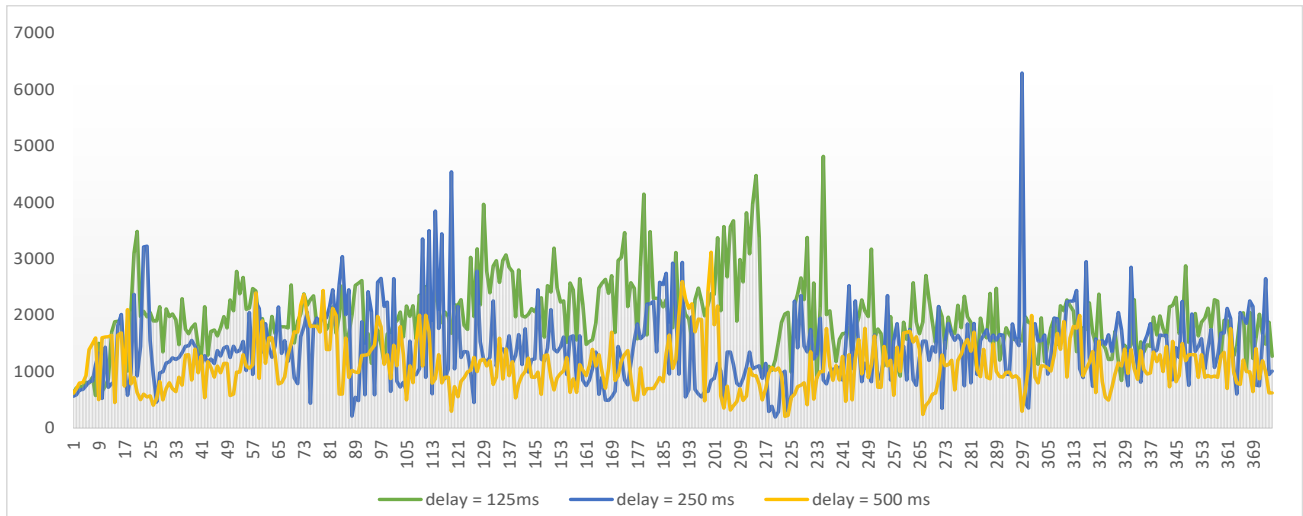


Figure 5-2-3. The response times when 25 requests sent per 15 users

The slowest response time was when the delay was set to 250ms (the blue line), its response time reached more than 6000ms. The fastest scenario was when the delay was set to 500ms (the yellow line), its highest response time reached less than 3000ms. Moreover, the last scenario's highest response time was almost 5000ms (the green line) when the delay was 125ms.

In all the pre-defined delays (1) and (2) charts of the previous evaluation results, the total number of requests is 500 because in (1) we have five users each sends 100 requests, and in (2) 10 users each sending 50 requests. In all (3) charts, there are 15 users each sends 25 requests, so the total of requests is 375. Hence, the third chart has the slightly faster round-trip times (response times).

Unlike the 5.1 evaluation results, in some of these charts the response time for each user is a little different. For the reason that the client was held in the PC along with the server (GAE). So, the data (attributes) were sent from the client to the proxy in the Raspberry Pi2 and from there to the database in the cloud. Furthermore, in all the charts there were peaks due to the network delay, backup processes were running, and the time used in sending the data to the proxy.

An additional main cause of the various fluctuations in all the charts is using Google cloud platform and storing the data and ABAC policies in Google SQL storage. While searching to understand better the evaluation results of the second implementation in this research, one point was noted which is the price of the plan. It is a privately owned platform; there are different pricing plans and amounts; as much as a user pay, he or she will get better services. This point is essential

because it affects the obtainable CPU slice and the storage; for instance, when sending 100 requests from 5 users, another request could come to the platform from a user who has a more expensive plan than the plan was selected in this project. In this case, the first request speed will decrease and the response time will increase accordingly. The same thing will happen for storing the data; there will be a huge number of users affected by the same physical storage.

Also, the X-axis in all the charts represents the requests number, and the Y-axis represents the response time.

CHAPTER 6

SUMMARY AND CONTRIBUTION

In conclusion, technology is growing quickly, especially in IoT systems. The growth is evidenced by the increase in the number of devices and applications that is based on the term IoT. This research included the next aspects:

- **Research problem:**

With all the technology in the world in our time, people using it to have its benefits in their business and homes. For example, IoT systems became popular, Park et al. stated that “there are more devices connected to the Internet than people on the planet, and the prediction is that there will be 50 billion IoT devices by 2020” [26]. IoT systems have the ability to be installed in low-powered devices (sensors and effectors) and to complete specific tasks for the users to make their everyday lives’ responsibilities easier and more comfortable. However, IoT still have some challenges; one of its main concerns is how to protect data in IoT systems from unauthorized access. This research is helping to introduce a solution to this concern.

- **The motivation:**

Because having an IoT system requires a budget and developing software from the system developer; also, needs much information from the users including private data such as the data that will be retrieved by the sensor and will be stored in the database. Hence, this thesis’s motivation to solve the prior problem is to prevent loss of money to the IoT system developer and avoid safety threats to the system’s users.

- **This research proposed solution:**

This project is aiming at creating a new approach (different from the traditional security methods like encryption techniques) to protect the data and deny any unauthorized access to the system. Therefore, an access control mechanism (ABAC) was applied to evaluate the users’ attributes for verifying the identity of the user.

To sum up this research work, an ABAC based system was built in two implementations:

1. In the first one, a low-powered device (Raspberry Pi 2) was used; a CoAP client tool used as the client; the server side is a written code by Go language, and the database is MySQL stored locally, and all the connections are via CoAP web services.
2. The second one consists of a client that is a mobile application tested on a smart device and a server that is created in a privately owned cloud platform along with the data storage, and the connection between them was HTTP web services.

ABAC rules were successfully applied in both implementations (local and cloud).

To evaluate the proposed solution:

Hence, one of the issues in IoT system is the large number of users and how the application will perform in an overload circumstances, the built proposed solution was evaluated by testing it in an overwork different cases (diverse numbers of delays, requests, and users) to compare the response time results between the two implementations.

To evaluate both the implementations prototypes, this research had an evaluation plan to mimic the real-world interactions. The evaluation involved different delay times in the client requests (125, 250, and 500) milliseconds to simulate different levels of concurrency. Moreover, each delay time consists of three different numbers of users (5, 10, and 15), each sending a different number of requests (100, 50, and 25). The goal of the evaluation was to retrieve the response times (round trip times) in each case.

In general, the results varied significantly between the first and second implementations. The first one was faster because the solution was built locally in SBC there was no network competition or delay. Unlike the second implementation where the client, proxy, and the server are separated in different machines, and the database was held in a cloud platform; there was a network delay and a waiting time for the data to be sent from the client to proxy in SBC to database in the cloud where the attributes were evaluated by ABAC policies.

➤ **The contribution:**

This research's domain is IoT hence it is focusing on low-powered and Internet-connected devices because they are essentials for IoT applications. Besides, in Chapter 2, some of the challenges in IoT systems; especially, in privacy area were presented.

The contributions of this research are:

- Applying an Access Control method on IoT system prototype in two scenarios:
 1. Using the low-powered device, which is Raspberry Pi 2, and a CoAP client that is Copper (Cu) CoAP user-agent for Firefox web browser (main contribution).
 2. Using Google SQL cloud platform, Google App Engine (GAE), and a mobile application as client.
- Obtain the response times by testing the system in different scenarios include dissimilar delays, users, and requests numbers, so we could prove that the built solution can work in situations similar to the applications in IoT environment.

Chapter 3 demonstrated some of the work and research that discussed some models about IoT and the security which concern the users or the application provider. This thesis applies Attribute Based Access Control policies (that can evaluate some of the user's attributes) on a low-powered device to introduce a prototype that indicates the ability to have some security approaches applied on a sensor for instance, not on the application side.

Furthermore, the same structure on a mobile cloud platform was applied for two reasons:

1. To compare the performance between the two scenarios while the number of users and requests change.
2. To take a future a step by providing a scenario that mimics the real-world applications since most of the IoT applications in the present time, offer the clients mobile applications to access the system and using a cloud-based storage to have more capacity. Worth mentioning that the second implementation is a simple prototype because the main contribution is the first implementation, so it needs more developing in the future.

While working in this research, two paper related to it were submitted to two conferences, and both were accepted. The first one was in *The 3rd International Symposium on Emerging Inter-*

Networks Communication and Mobility (EICM-2016). The second one was in The 7th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEEE IEMCON 2016).

CHAPTER 7

FUTURE WORK

For the future work, there are some features could be added to improve the system prototype that was built in this research. Such as:

- **More attributes and platforms**

More attributes like the user location, and IP address should be studied and applied to help the system make better decision. Because having more evaluated number of attributes will produce results that are more accurate about the users. As Hu et al. stated in [4] “ABAC enables precise access control, which allows for a higher number of discrete inputs into an access control decision, providing a bigger set of possible combinations of those variables to reflect a larger and more definitive set of possible rules to express policies.”

Also, testing the built solution using different platforms. For example, instead of using Raspberry Pi 2, alternative low-cost computers could be used like System On Chip (Intel Edison) or Arduino. Moreover, another cloud platform could be used as a substitute of the Google cloud platform like Amazon or Azure cloud services.

- **Adding the higher level of security**

This step is required in case of allowing an over-write in the system. For example, if the IoT system owner wants to change some of the system settings and his attributes do not fit the ABAC policies. Another type of access control might be added such as (IBAC). In IBAC, [9] stated that “permissions to access a resource is directly associated with a subject’s identifier” to grant the access to the user. A subject identifier could be a fingerprint of the system owner.

- **Using HTTPS instead of HTTP**

In the second implementation, HTTP was applied as a web service to send the data from the client to the proxy in SBC to the database in the cloud platform. The sent data is the attribute including the device MAC address, and the user username and password. HyperText Transfer Protocol Secure (HTTPS) is the secure version of HTTP. In this protocol, all the data and communications that sent between the client and the server are encrypted. HTTPS is frequently used to protect highly private online transactions like online banking and online shopping order forms.

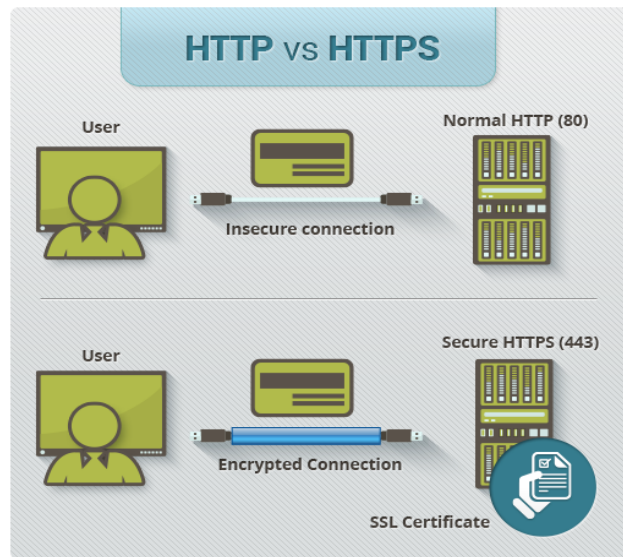


Figure 7-1. HTTP vs. HTTPS¹⁵

Moreover, there are general challenges about IoT need to be focused on enhancing the IoT systems in the future:

- **Lack of unlimited bandwidth and networking infrastructure**

The newer devices are added on the network, the more traffic the network need to deal with. As a solution, the network infrastructure could be expanded, and that is what all service providers are doing all the time. However, it is a slow and costly process. In the lack of faster ways to expand the network's capacity, this will continue to be a limiting issue for the growth rate of IoT.

¹⁵ <https://www.instantssl.com/ssl-certificate-products/https.html>.

- **Power consumption**

Power is a problematic issue in IoT systems. One of the biggest advantages of IoT is the ability to handle numerous devices that widespread without structure them into traditional infrastructure. So being capable of unchaining IoT hardware from a constant power source is a requirement for achieving IoT full potential. Nevertheless, the technology to accomplish that is not present yet. Predictions show that it will take time before batteries in IoT devices can last for years.

In conclusion, this research includes different areas of research (IoT, Access Control, Cloud computing), and they could raise some issues in the current work. Thus, this research, enabling Attribute Based Access Control within Internet of Things (IoT), could still develop.

REFERENCES

- [1] J. Koufopoulos, “9 Examples of the Internet of Things That Aren’t Nest,” *The Percolate Blog*, 23-Jan-2015. [Online]. Available: <https://blog.percolate.com/2015/01/9-examples-internet-things-arent-nest/>. [Accessed: 15-Jan-2016].
- [2] C. Tozzi, “IoT Past and Present: The History of IoT, and Where It’s Headed Today,” 25-May-2016. [Online]. Available: <http://mspmentor.net/msp-mentor/iot-past-and-present-history-iot-and-where-its-headed-today>. [Accessed: 08-Sep-2016].
- [3] M. Weber and M. Boban, “Security challenges of the internet of things,” in *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2016, pp. 638–643.
- [4] V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, “Guide to Attribute Based Access Control (ABAC) Definition and Considerations,” National Institute of Standards and Technology, NIST SP 800-162, Jan. 2014.
- [5] R. Focardi and R. Gorrieri, Eds., *Foundations of security analysis and design: tutorial lectures*. Berlin; New York: Springer, 2001.
- [6] S. Musa, “Cybersecurity: Access Control,” *The EvoLLLution*, 04-Feb-2014. .
- [7] S. Kaiwen and Y. Lihua, “Attribute-Role-Based Hybrid Access Control in the Internet of Things,” in *Web Technologies and Applications*, Springer, 2014, pp. 333–343.
- [8] M. Adda, J. Abdelaziz, H. Mcheick, and R. Saad, “Toward an Access Control Model for IOTCollab,” *6th Int. Conf. Ambient Syst. Netw. Technol. ANT 2015*, vol. 52, no. 2015, pp. 428 – 435, 2015.
- [9] E. Yuan and J. Tong, “Attributed based access control (ABAC) for Web services,” in *2005 IEEE International Conference on Web Services, 2005. ICWS 2005. Proceedings*, 2005, p. 569.
- [10] L. Wang, D. Wijesekera, and S. Jajodia, “A Logic-based Framework for Attribute Based Access Control,” in *Proceedings of the 2004 ACM Workshop on Formal Methods in Security Engineering*, New York, NY, USA, 2004, pp. 45–55.
- [11] X. Wang, J. Zhang, E. M. Schooler, and M. Ion, “Performance evaluation of attribute-based encryption: Toward data privacy in the IoT,” in *Communications (ICC), 2014 IEEE International Conference on*, 2014, pp. 725–730.
- [12] X. Yang, Z. Li, Z. Geng, and H. Zhang, “A Multi-layer Security Model for Internet of Things,” *IOT Workshop 2012*, pp. 388–393, 2012.
- [13] B. Cummings, “10 IT Risk and Security Trends to Watch | Risk Management,” 01-Feb-2011. .
- [14] N. Leavitt, “Today’s Mobile Security Requires a New Approach,” *Computer*, vol. 46, no. 11, pp. 16–19, Nov. 2013.
- [15] T. Armerding, “The 15 worst data security breaches of the 21st Century,” *CSO Online*, 15-Feb-2012. [Online]. Available: <http://www.csoonline.com/article/2130877/data->

protection/data-protection-the-15-worst-data-security-breaches-of-the-21st-century.html.
[Accessed: 17-Feb-2016].

- [16] J. Webber, S. Parastatidis, and I. Robinson, *REST in Practice: Hypermedia and Systems Architecture*. O'Reilly Media, Inc., 2010.
- [17] O. DOSPINESCU and M. PERCA, "Web Services in Mobile Applications," *Inform. Econ.*, vol. 17, no. 2, pp. 17–26, 2013.
- [18] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful web services vs. big'web services: making the right architectural decision," in *Proceedings of the 17th international conference on World Wide Web*, 2008, pp. 805–814.
- [19] J. H. Christensen, "Using RESTful Web-services and Cloud Computing to Create Next Generation Mobile Applications," in *Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications*, New York, NY, USA, 2009, pp. 627–634.
- [20] X. Chen, "Constrained Application Protocol for Internet of Things," Apr. 2014.
- [21] P. Nielsen, "The importance of context in keeping end users secure," *Netw. Secur.*, vol. 2015, no. 2, pp. 10–13, Feb. 2015.
- [22] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," Jun-2014. [Online]. Available: <https://tools.ietf.org/html/rfc7252>. [Accessed: 21-Mar-2016].
- [23] T. Jaffey, "MQTT and CoAP, IoT Protocols," Feb-2014. [Online]. Available: http://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php. [Accessed: 24-Mar-2016].
- [24] Lady Ada, "Benchmarks & Performance Improvements | Introducing the Raspberry Pi 2 - Model B | Adafruit Learning System," 04-May-2015. [Online]. Available: <https://learn.adafruit.com/introducing-the-raspberry-pi-2-model-b/performance-improvements>. [Accessed: 19-Sep-2016].
- [25] A. Williams, "Raspberry Pi 2: Power and Performance," *TrustedReviews*, 17-Nov-2015. [Online]. Available: <http://www.trustedreviews.com/raspberry-pi-2-review>. [Accessed: 19-Sep-2016].
- [26] N. Park, H. Hu, and Q. Jin, "Security and Privacy Mechanisms for Sensor Middleware and Application in Internet of Things (IoT)," *Int. J. Distrib. Sens. Netw.*, vol. 12, no. 1, p. 2965438, Jan. 2016.
- [27] I. M. Abbadì and C. J. Mitchell, "Digital Rights Management Using a Mobile Phone," in *Proceedings of the Ninth International Conference on Electronic Commerce*, New York, NY, USA, 2007, pp. 185–194.
- [28] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," *Commun ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.
- [29] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wirel. Commun. Mob. Comput.*, vol. 13, no. 18, pp. 1587–1611, Dec. 2013.

- [30] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Gener. Comput. Syst.*, vol. 29, no. 1, pp. 84–106, Jan. 2013.
- [31] D. Jana and D. Bandyopadhyay, "Management of identity and credentials in mobile cloud environment," in *2013 International Conference on Advanced Computer Science and Information Systems (ICACISIS)*, 2013, pp. 113–118.
- [32] R. K. Lomotey and R. Deters, "Mobile-Based Medical Data Accessibility in mHealth," in *2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, 2014, pp. 91–100.
- [33] A. N. BAHAR, M. A. HABIB, and M. M. ISLAM, "Security architecture for mobile cloud computing," *Int. J.*, vol. 3, no. 3, pp. 2305–1493, 2013.
- [34] Y. Lin, C. Huang, M. Wright, and G. Kambourakis, "Mobile Application Security," *Computer*, vol. 47, no. 6, pp. 21–23, 2014.
- [35] D. Kang, J. Oh, and C. Im, "Context Based Smart Access Control on BYOD Environments," in *Information Security Applications*, K.-H. Rhee and J. H. Yi, Eds. Springer International Publishing, 2014, pp. 165–176.
- [36] I. Suominen, "Access Control for Internet of Things," *Intopalo*, 25-May-2015. .
- [37] B. Cha, "A Beginner's Guide to Understanding the Internet of Things," *Re/code*, 15-Jan-2015. .
- [38] J. Shi, Y. Li, and R. H. Deng, "A secure and efficient discovery service system in EPCglobal network," *COSE Comput. Secur.*, vol. 31, no. 8, pp. 870–885, 2012.
- [39] W. Colitti, K. Steenhaut, and N. De Caro, "Integrating wireless sensor networks with the web," *Extending Internet Low Power Lossy Netw. IP SN 2011*, 2011.
- [40] A. Keränen and C. Bormann, "Internet of Things: Standards and Guidance from the IETF | Internet Society," Apr-2016. [Online]. Available: <http://www.internetsociety.org/publications/ietf-journal-april-2016/internet-things-standards-and-guidance-ietf>. [Accessed: 24-Mar-2016].
- [41] G. Halfacree, "Intel unveils Quark-based Edison microcomputer," *bit-tech*. [Online]. Available: <http://www.bit-tech.net/news/hardware/2014/01/07/intel-edison/1>. [Accessed: 22-Apr-2016].
- [42] B. Singh, "How Internet of Things (IoT) Are Going To Impact Your Business?," *Business 2 Community*, 15-Jun-2016. [Online]. Available: <http://www.business2community.com/big-data/internet-things-iot-going-impact-business-01572401>. [Accessed: 02-Aug-2016].
- [43] I. T. S. Sector, "ITU-T Recommendation Z. 120," *Message Seq. Charts MSC96*, 1996.
- [44] P. Wayner, "Ultimate cloud speed tests: Amazon vs. Google vs. Windows Azure," *InfoWorld*, 26-Feb-2014. [Online]. Available: <http://www.infoworld.com/article/2610403/cloud-computing/ultimate-cloud-speed-tests--amazon-vs--google-vs--windows-azure.html>. [Accessed: 19-Sep-2016].
- [44] "Attribute Based Access Control (ABAC) Overview," 06-May-2015. [Online]. Available: <http://csrc.nist.gov/projects/abac/>. [Accessed: 09-Sep-2016].

- [45] “Copper (Cu) CoAP user-agent - JavaScript CoAP Implementation.” [Online]. Available: <http://people.inf.ethz.ch/mkovatsc/copper.php>. [Accessed: 23-Mar-2016].
- [46] “Gartner Says Tablet Sales Continue to Be Slow in 2015.” [Online]. Available: <http://www.gartner.com/newsroom/id/2954317>. [Accessed: 08-Jan-2016].
- [47] “HTTP to HTTPS | What is a HTTPS Certificate.” [Online]. Available: <https://www.instantssl.com/ssl-certificate-products/https.html>. [Accessed: 08-Sep-2016].
- [48] “MySQL :: MySQL Documentation.” [Online]. Available: <http://dev.mysql.com/doc/>. [Accessed: 05-Apr-2016].
- [49] “Postman | Supercharge your API workflow.” [Online]. Available: <https://www.getpostman.com/>. [Accessed: 17-May-2016].
- [50] “The Go Programming Language.” [Online]. Available: <https://golang.org/>. [Accessed: 06-Apr-2016].
- [51] “Welcome to Python.org,” Python.org. [Online]. Available: <https://www.python.org/>. [Accessed: 05-Apr-2016].
- [52] “World’s first Internet of Things has successful trial,” NewsX, 22-Feb-2016. [Online]. Available: <http://www.newsx.com/tech/21394-worlds-first-internet-of-things-has-successful-trial>. [Accessed: 15-Jul-2016].