THREE-TIERED ARCHITECTURE FOR ACCESS CONTROL IN INTERNET OF MEDICAL THINGS

MAHI UDDIN

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITI MALAYA
KUALA LUMPUR

2022

THREE-TIERED ARCHITECTURE FOR ACCESS CONTROL IN INTERNET OF MEDICAL THINGS

MAHI UDDIN

DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE (APPLIED COMPUTING)

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITI MALAYA
KUALA LUMPUR

2022

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: **Mahi Uddin**

Matric No: **17198918/1** (new); **WOA180015** (old)

Name of Degree: **Master of Computer Science (Applied Computing)**

Title of Dissertation ("this Work"):

THREE-TIERED ARCHITECTURE FOR ACCESS CONTROL IN INTERNET OF MEDICAL THINGS

Field of Study: **Security on IoT**

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date: 01/02/2022

Subscribed and solemnly declared before,

Witness's Signature Date: 05.02.2022

Name:

Designation:

THREE-TIERED ARCHITECTURE FOR ACCESS CONTROL IN INTERNET OF MEDICAL THINGS

ABSTRACT

The Internet of Things (IoT) has made a successful impact on our personal and professional life. More specifically, the internet of medical things (IoMT) is capturing a crucial role in improving healthcare and providing medical facilities to human-being worldwide. Different smart sensors are connected to patients and transfer important data to the healthcare system to monitor and provide better patient treatment. As a result, the number of users is increasing drastically. Connected Internet of Things devices will be 52.2 billion by 2022. Although IoMT is counted as a promising apart in healthcare, the security of patients' information is still a challenging issue that should be thoroughly addressed. Access control is one of the leading security mechanisms introduced to healthcare to limit access to patients' information to only authorized individuals. In healthcare, access to the medical record of patients within a minimal time is also extremely important because less time maybe save a life. In this research, designed a three-tiered architecture for access control (T-TAFAC) on IoMT. That uses a hybrid approach by combining a fine-grained attribute-based access control model (ACM), role-based access control model (RBAC), capability-based access control model (CBAC), and using fog computing. Fog computing is a new era of cloud computing that provides computation, storage, and application services to end-users. This implemented schema is eXtensible Access Control Markup Language (XACML) driven. This T-TAFAC reduces the existing processing time. T-TAFAC is emulated as a testbed, provide a performance evaluation of this architecture, and making it compatible for practical use.

Keywords: Access Control, Internet of Medical Things (IoMT), Access Control Model, Sensors, Fog Computing.

SENIBINA TIGA-LAPISAN UNTUK KAWALAN AKSES DI INTERNET BENDA PERUBATAN

ABSTRAK

Internet Benda (IoT) telah berjaya memberi kesan yang baik kepada kehidupan peribadi dan profesional kami. Lebih khusus lagi, Internet Benda Perubatan (IoMT) memainkan peranan penting dalam meningkatkan penjagaan kesihatan dan menyediakan kemudahan perubatan kepada manusia di seluruh dunia. Sensor pintar yang berbeza dihubungkan dengan pesakit dan memindahkan data penting ke sistem penjagaan kesihatan untuk memantau dan memberikan rawatan pesakit yang lebih baik. Akibatnya, jumlah pengguna meningkat secara drastik. Peranti Internet Benda yang disambungkan akan menjadi 52.2 bilion menjelang 2022. Walaupun IoMT dianggap sebagai menjanjikan selain dalam penjagaan kesihatan, keselamatan maklumat pesakit masih merupakan masalah yang mencabar yang harus ditangani secara menyeluruh. Kawalan akses adalah salah satu mekanisme keselamatan terkemuka yang diperkenalkan kepada penjagaan kesihatan untuk membatasi akses maklumat pesakit hanya kepada individu yang diberi kuasa. Dalam penjagaan kesihatan, akses ke rekod perubatan pesakit dalam waktu minimum juga sangat penting kerana kurang masa mungkin menyelamatkan nyawa. Dalam penyelidikan ini, kami merancang seni bina kawalan akses (ACA). Itu menggunakan pendekatan hibrid dengan menggabungkan model kawalan akses berasaskan atribut halus (ACM), model kawalan akses berasaskan peranan (RBAC), model kawalan akses berasaskan keupayaan (CBAC), dan menggunakan pengkomputeran kabut. Fog computing adalah era baru pengkomputeran awan yang menyediakan perkhidmatan pengiraan, penyimpanan, dan aplikasi kepada pengguna akhir. Skema yang dilaksanakan ini didorong oleh Bahasa Markah Kawalan Akses eXtensible (XACML). Senibina kami yang dirancang mengurangkan masa pemprosesan yang ada. Kami telah meniru testbed, memberikan penilaian prestasi seni bina kami, dan menjadikannya serasi untuk penggunaan praktikal.

Kata kunci: Kawalan Akses, Internet Perkara Perubatan (IoMT), Model Kawalan Akses, Sensor, Komputasi Kabus.

ACKNOWLEDGEMENTS

First of all, all praises be to Allah SWT, the most Merciful, the Gracious and the Supreme powerful, who has given me the capability and strength to accomplish this work. May Allah (SWT) give the highest rank to Prophet Mohammad (peace be upon him) and his companions.

I want to express my deepest respect and gratitude to my supervisor Associate Professor Dr. Rafidah Binti Md Noor, for her academic support, extraordinary guidance, and encouragement throughout my research.

I am also thankful to Multimedia Lab, Wisma R&D, University Malaya for allowing me to use their instrument. My compliments go to my fellow lab mates Muhammad Umair Munir, Mohammad Mustaneer Rahman, and Mujibur Rahman, who all shared their experiences to make my work more accessible. I also extended thanks to all FSKTM staff members for their cooperation and input according to requirements.

I am also grateful to Luva Nahid Choudhury, the Director-General of Bengal Foundation, to accept my study leave and give me scope to continue my job part-time by providing remote support.

Finally, I find my words inadequate to express my heartfelt gratitude to my beloved wife Shabicun Nahar Nawme for her continues help, support and encouragement in each stage of my study period. I can’t forget Md Nazrul Islam, Tazul Islam, Saiful Islam and my other school friends, who supported me mentally and financially. Last but not least, I would like to thank my father, father and mother-in-law, children and other family members for their encouragement, support and love.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ABAC	Attribute-based access control
ACA	Access control architecture
ACM	Access control model
ACL	Access control lists
CBAC	Capability-based Access Control
EC2	Elastic compute cloud
HIT	Healthcare Information Technology
IBM	International Business Machines
IP	Internet protocol
IoT	Internet of things
IoMT	Internet of medical things
LAC	Logical access control
LDAP	Lightweight directory access protocol
MIMT	Man-in-middle-attack
NIST	National Institute of Standards and Technology
PII	Personally identifiable information
RBAC	Role-based access control
SAML	Security Assertion Markup Language
T-TAFAC	Three-tiered architecture for access control
XACML	eXtensible Access Control Markup Language

CHAPTER 1: INTRODUCTION

1.1 Research Background

Internet of Things (IoT) is an emerging technology. Nowadays, every industry is using IoT because of its adjuvant benefits. IoT devices reduce time, cost, and human interactions. As a result, its span of use is expanding fast. Connected IoT devices will be $52.2 billion by 2022 (Karen et al., 2018). The health industry also grabs this enormous technology. The Internet of Medical Things (IoMT) is the collection of medical devices and applications that connect to Healthcare Information Technology (HIT) systems through online computer networks. About 70% of healthcare equipment is using the Internet of Medical Things (IoMT) (Kayla, 2019).

Generally, HIT collects the patient's data basically from the hospital territory and anonymous places and process those data on the cloud, monitoring the patient's condition, and take the appropriate treatments by the healthcare personnel. Patient's personal and medical information is crucial to protect from unauthorised access. Security is the main threat to this system. These security risks may include applications, steal data from an enterprise, or using them, as seen in the Mirai botnet (Jerkins, 2017). Human lives can be under threat due to a little wrong data or misinformation. Hence, healthcare-related data require more consciousness than other data because life safety relies on it.

Cloud was introduced in the medical sector to give the best patient data availability to access by doctors from any part of the world at any time. But in the medical sector data processing time is very crucial. To reduce data processing time fog technology offers faster processes, which is crucial in medical sector.

Various types of sensors have been growing in health facilities in recent years as the cost of sensors has decreased significantly, and electronic equipment have continued to miniaturise. Patients are also wearing an increasing number of tiny sensors to monitor a variety of vital indicators related to their health. These sensors produce massive amounts of data.

Several systems in a healthcare institution are intended to operate with real-time data from sensors, meters, and a variety of other devices needed to ensure the facility's functioning. Furthermore, sensor signals are used by electronic control systems in medical equipment and other probes for accurate diagnosis and treatment. Temperature sensors, pressure sensors, position sensors, humidity sensors, air bubble detectors, force sensors, photo optic sensors, and piezo film sensors are just a few of the sensors available. They gather the information that is applied in a number of medical applications to improve the way patients are monitored and treated. They are deployed in many healthcare applications to offer reliable monitoring, diagnosis, and treatment. Many healthcare institutions are implementing smart healthcare apps leveraging sophisticated IoT technologies. Blood Pressure Monitoring, Body Temperature Measurement, Glucose Monitoring, Medical Pump Technology, Dialysis Equipment, Medical Wearable Technology, Sleep Monitoring, and Assisted Baby Delivery are just a few of numerous featured applications.

IoT solutions for healthcare hold the potential of making health organisations' operations more intelligent and more efficient. IoT has the potential to change the way equipment, technology, and people interact and connect in health care settings, resulting in improved treatment and reduced costs (Badidi & Moumane, 2019). Healthcare IoT solutions include:

• Remote monitoring solutions for patient health

- Remote care for patients
- Management of hospital assets and devices using IoT
- Preventative maintenance solutions for medical equipment
- Intelligent solutions for emergency management in hospitals

Figure 1.1 shows that the sensors devices are connecter to the cloud and all users get the sensor data from the cloud. This is a centralized cloud based IoMT networks scenario.

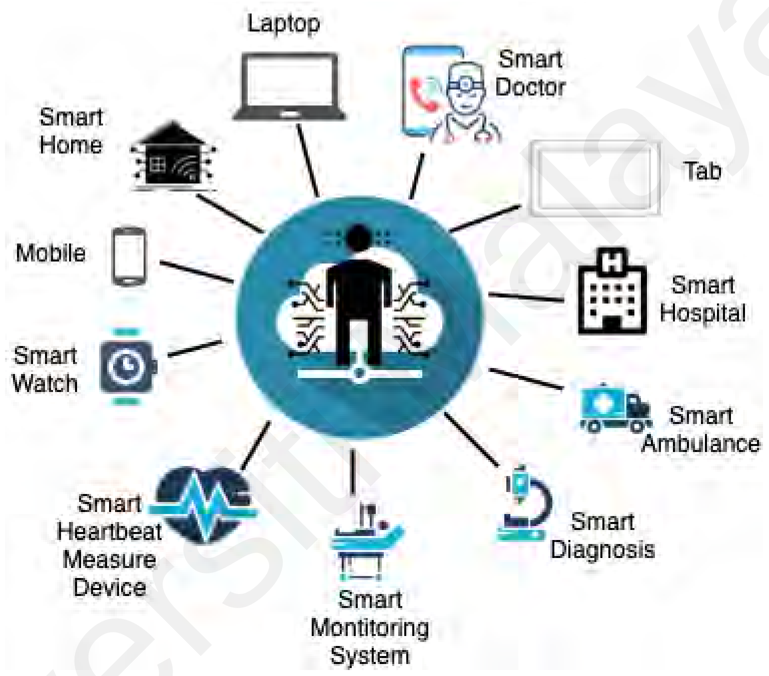


Figure 1.1: IoMT uses scenario in Hospital.

By bringing together the IoT and data analytics, healthcare companies will be able to collect and analyse data from healthcare systems and distant facilities to deliver the proper information to the appropriate person at the right moment, from any location, at any time. Data analytics will also assist healthcare facilities in improving patient satisfaction, operational efficiency, and security for all. Additionally, facility operators will gain more significant insights into how each building or piece of medical equipment is operating, allowing them to make decisions that are supported by evidence and predictable patterns.

The data contained in healthcare records including names, birth dates, policy numbers, diagnosis codes, social security numbers, and billing information has a much longer shelf life and is rich enough for identity theft. The main challenge of the Access Control Architecture is to secure unauthorized access, manage the policy and reduce the latency (Abu Bakar & Jamal, 2020). Because, if the access control takes more time, human lives will be in threat.

The access control model is one of the important components in access control architecture. There are many types of access control models, three of them are mostly in use. The pros and cons of these ACMs are discussed below:

Attribute-based Access Control Model: It is very adaptable in an extensive open system where the numbers of potential users are very high, and most users will not be known before. Infrastructure is put in place to ensure the security of data and user attributes, as well as to perform authorization and authentication tasks. It does not utilize attributes to establish permission between subjects and resource directly, but it does use them as the foundation for authorization. It gives the global agreements of the attributes so that attributes provided in one domain could be forwarded to the other domain during domain-to-domain interaction. A central repository for user attributes enables interoperability and data exchange across many service providers who may make use of the information. The main disadvantage of this model is a high level of complexity owing to policy definition and maintenance. The user's attributes may not always match those utilized by the web-based system's service provider. Low expressiveness of the attribute given by the different organizations with a standard set of standardized attributes. The variety of user data adds to the complexity, which can only be addressed by a centrally managed database with all attributes in the same format.

Role-based Access Control Model: Advantages of this model is its simplicity and static nature; the whole created model is readily understood by the administrator. Simple to use since roles and permissions are defined by the administrator. Best for the local domain because in the local domain, the users did not go for quick changes as compared to broader domains, which needs dynamicity. Statically defined roles, lower complexity for the system on runtime because of permissions to roles and roles to users assigned manually by an administrator. The main cons are that changing an entity's access privileges without changing its roles is not feasible. It does not engage in sharable semantic domain models. It does not include the ability to reason over utilities. Delegation models are not required by a dynamic environment. It does not Support justification advising and negotiation necessary for greater autonomy. It has a mobility problem. So, it is very hard to maintain the roles wise policy.

Capability-based Access Control Model: Capabilities are protected addresses. They can be easily copied, passed as parameters, and transmitted from domain to domain, but cannot be modified or forged by users. Capabilities are independent of context. Regardless of the domain or process in which they are employed, they address the same object. Access privileges and addressing information are both included in capabilities. The address or identifier in a capability is independent of the concrete base and limit information used for memory mapping. This identifier is used to locate a single physical descriptor for the addressed object. Capabilities and the objects they reference may be stored in long-term storage. Capabilities offer a standardized method for naming all kinds of objects in the environment, including both hardware and software. (Henry, 1984). CBAC is unsuitable if a collection of resources is large and often changes. It's tough to find out which people have access to a file and altering the status of a file is complicated.

Every ACM has its own merits and drawbacks as discussed above. So, to design an access control architecture selecting an apt access control model is very important. Because

1.2 Problem Statement

To save human lives, the healthcare sector now uses real-time patient’s data. Patients' information is saved in a cloud database. An authenticated user may verify his information or information to which he has access. The doctor may look up his patient's medical history and prescribe the appropriate medicine. A user's record, prescription history, and doctor's reviews may all be seen. Due to privacy issues, the patient's personal information is essential. (Siddiqui et al., 2018). As a result, such records should be accessible in a secure way with little processing time. Protecting personal information from unauthorised access is a major issue (Vijayalakshmi & Arockiam, 2018). Another major issue is to get this information in the shortest possible period; in an emergency, the shortest possible processing time to obtain a patient's history may save a human life.

To resolve these challenges and to give better management on authorizations, many researchers proposed many methods such as (Zhang & He, 2016) proposed hybrid access control model; attribute-based access control, role-based access control, and capability-based access control, however, the model comparatively takes more time than others access control models. Alnefaie et al. (2011) had reduced the time using ABAC and fog computing, contrary ABAC has a performance deficiency than the other model (Zhang & He, 2016). Moreover, some researchers proposed traditional access control approaches, such as Access Control Lists (ACL), Role-based Access Control (RBAC) and Attribute-based Access Control (ABAC), which are not considered scalable, manageable, and efficient mechanisms that shall meet the requirements of IoT systems. From literature investigations, no research has implemented fog technology by combining three ACMs in IoMT networks. Therefore, to consider the latency of access authorization, this research designs a three-tiered architecture for access control by introducing fog technology and combining three ACMs.

1.3 Research Questions

RQ1: What is the state-of-art of the access control architectures in IoMT?

RQ2: How to minimize the authorization processing time by designed a three-tiered architecture for access control in IoMT?

1.4 Research Aim

This research explores and designs a three-tiered architecture for access control in IoMT based on a hybrid of the attribute, role and capability-based access control models considering authorization processing time and performance.

1.5 Research Objectives

Following research objectives would facilitate the achievement of this aim:

a. To explore the state-of-art of access control architecture in IoMT.
b. To design a three-tiered architecture for access control in IoMT based on a hybrid of the attribute, role, and capability-based access control models with considerable authorization processing time.
c. To implement and analyze the three-tiered architecture for access control in IoMT with considerable authorization processing time.

1.6 Research Outcomes

At the end of this research, the anticipated outcomes are as follows:

- Describe a taxonomical knowledge of the existing and designed a three-tiered architecture for access control in IoMT.
- Implemented and analyzed the fine-grained three-tiered architecture for access control that considers authorization processing time in IoMT.

Table 1.1 shows the mapping of the problem statement, research questions, research objectives and research outcomes.

Table 1.1: Mapping of the problem statement, research questions, research objectives and research outcomes

Problem Statement	Research Questions	Research Objectives	Research Outcomes
The knowledge gap of different Access Control Architecture in IoMT.	What is the state-of-art of the Access Control Architecture in IoMT?	To explore the state-of-art of Access Control Architecture in IoMT.	Describe a taxonomical knowledge of existing and designed access control architecture in IoMT.
Comparatively, Access Control Architectures are consuming more time on the authorization.	How to minimize the authorization processing time by designed a Three-Tiered Architecture for Access Control in IoMT?	- To design a Three-Tiered Architecture for Access Control in IoMT based on a hybrid of the attribute, role, and capability-based access control models with considerable authorization processing time. - To implement and analyze the Three-Tiered Architecture for Access Control in IoMT with considerable authorization processing time.	Implemented and analyzed the fine-grained Three-Tiered Architecture for Access Control that considers authorization processing time in IoMT.

1.7 Chapter Organization

To document the present research work, the dissertation has been organized in the following way. Chapter one (current chapter) presents the background of the work, followed by the problem statement, research questions, and research aim and objectives.

Chapter two presents the related work on access control architecture, which include the different tier of access control architecture, literature review on existing access control architecture, and the comparison table between different access control architecture.

Chapter three provides a methodology on how the research was conducted. Moreover, the designed three-tiered architecture for access control on IoMT and how it works to reduce the authentication processing time and the simulation of the architecture is also presented in chapter three. The simulation is carried out by using NodeMCU, thermal sensor and C# programs as Central Management System. NodeMCU is used as a thing that is open-source, interactive, programmable, low cost, simple, smart Wi-Fi (ESP8266-12E microcontroller) enabled. The ESP8266 is a cheap Wi-Fi microchip with a full TCP/IP stack and microcontroller capability. Laptop and desktop are used as client and server. The thermal sensor is for gathering the information and developed three programs to collect data from patients and transfer those data to desired persons like as medical personnel, patients and guardians.

Chapter four contains the outcomes of the present research work. All the results are discussed with reasoning and a performance analysis of the developed system is given in this chapter.

Finally, in chapter five, the research conclusion is presented. In addition, the achievements of this research and future works are also outlined in that chapter.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This chapter reviews the research papers and findings from other researchers that are related to access control architecture. In the first section, brief description of the concept of the different important components like cloud computing, fog computing, IoT, XACML, access control model with access control architecture are delineated. In the third section, access control architectures that were designed by the other researchers are discussed. In section fourth, already designed architecture are compared with each other. Challenges in access control architecture are described in fifth section. And in the last section, a summary of the chapter is included.

2.2 Cloud Computing

Cloud computing refers to the provision of computer system resources on-demand, particularly data storage (cloud storage) and processing power, without the need for direct active administration on the part of the user. The following describes the concept, history, architecture, and characteristics of cloud computing.

2.2.1 Concept of Cloud Computing

The objective of cloud computing is to enable consumers to take advantage of all of these technologies without requiring extensive knowledge or experience in each one. The cloud promises to reduce expenses by allowing customers to focus on their main business rather than being hampered by IT issues. Virtualization is the main enabling technology for cloud computing. Cloud computing is often used to describe data centres that are accessible to a large number of people through the internet. Large cloud servers, which are prevalent nowadays, often contain services that are spread over many locations from a single central server. If the connection to the user is quite near, it may be appropriate to identify the server as an edge one.

2.2.2 History of Cloud Computing

The word ‘cloud’ means provider or carrier who offers services via internet, while ‘computing’ refers to processing, computations, or calculations as well as the numerous resources that are made available by computers. The idea of cloud computing was first introduced in 1961 by John McCarthy at the Massachusetts Institute of Technology (MIT). He said, “If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility. The computer utility could become the basis of a new and important industry.” (Surbiryala & Rong, 2019).

Salesforce was one of the first businesses to deal with the idea of cloud computing in the late 1990s. The company began offering Software as a Service (SaaS), which allows customers to manage their client relationships. In addition to Platform as a Service (PaaS), which offers clients development platforms such as Microsoft Azure and Google's Application Engine, the salesforce model is a common cloud computing paradigm. The other type is the Infrastructure as a Service (IaaS) model, which began in 2006 with Amazon Elastic Compute Cloud (EC2). Cloud computing became popular in 2007 as a result of the fast growth of communication channels and a rise in the geometric progression of corporate and private users' demands to extend their information systems (Arutyunov, 2012).

Many universities in the United States began collaborating with Google and IBM in 2007 to promote cloud computing programmes at their campuses. This assisted in lowering the cost of academic study by sharing resources among students and constructing a significant processing capacity or computer power to access it via internet. Many other universities across the world followed the same trend during the subsequent years (Sultan, 2010).

By 2025, there will be over 163 zettabytes of data stored in the cloud (Reinsel et al., 2017). Cloud data centers will process 94% of all workloads in 2021. In 2020, the total worth of the market was $371.4 billion. With a compound annual growth rate (CAGR) of 17.5%, it is projected that the market will amount to $832.1 billion by 2025 (Chauhan & Sood, 2021).

2.2.3 Architecture of Cloud Computing

In traditional cloud-based architecture for IoT is shown in Figure 2.1, device-to-cloud architecture is the sole communication mechanism that connects the leading cloud and the nodes in the network. Some major models of cloud computing that are used at present are Application as a Service (AaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

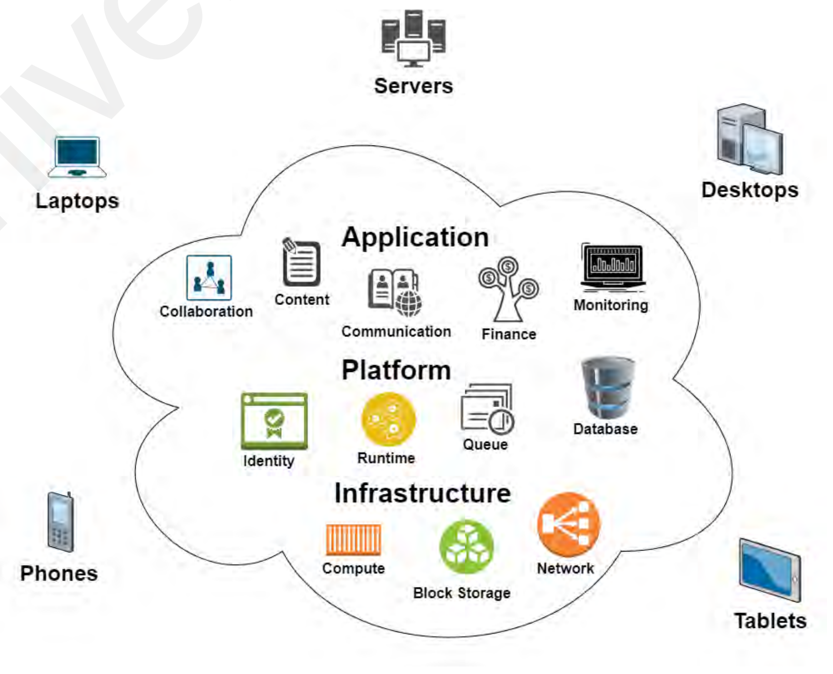


Figure 2.1: Cloud computing architecture. Wikipedia (2022)

Mobile phones, laptops, tablets, desktops, and servers' users get the service from the different models of cloud computing.

2.2.4 Characteristics of Cloud Computing

In many aspects, cloud computing differs from local computing. NIST has discovered five distinct characteristics:

On-demand self-service: No matter what exact resource is required; a user can directly get the requisite computational capabilities from the source.

Broad network access: A user is not bound to a single place and may access resources from anywhere there is a connection to a network (usually the internet).

Resource pooling: Many users share the same set of resources from a provider, consuming what they need, without having to think themselves about where those resources came from.

Rapid elasticity: Users can easily raise or reduce their use of a computing resource in response to their urgent demands.

Measured service: The provider keeps track of a customer's use, which might be used for invoicing or other purposes.

2.2.5 Cloud Computing in Healthcare

The healthcare sector, on the other hand, has found that a traditional device-to-cloud design is inefficient. If you rely only on a distant cloud server, for example, you may have congestion and latency problems. This main server room is prone to mistakes that may have a negative effect on the safety of patients (Salonikias et al., 2016; Kraemer et al., 2017). When confronted with an urgent or emergency situation that may occur at any time or in any location, one of the most important problems in health care is what to do. Therefore, emergency management is largely reliant on the dissemination of information to the public and medical professionals. The cloud computing paradigm, on the other hand, requires that programmes and data be completely processed on the cloud, which is time-consuming. Cloud methods cannot be controlled in real-time in an emergency. For this purpose, a system that can constantly and quickly monitor and report on the patient's condition is required (Kraemer et al., 2017), and one of the possible options is to use fog computing technology.

2.3 Fog Computing

Fog computing is a novel trend that provides an architectural approach for distributed systems by extending the storage, networking, computing facilities, and communication of cloud computing to the edge of the networks (Kraemer et al., 2017; Maksimović, 2018). The following describes the concept, history, architecture, and characteristics of fog computing.

2.3.1 Concept of Fog Computing

It is built on the IoT, and it allows for data processing close to data sources, as well as analysis and knowledge creation inside the data source. Fog computing is a technique for reducing the quantity of data sent between end-user devices and the cloud. Fog computing, on the other hand, differs from cloud computing in that it works at the edge of the network, resulting in a significant reduction in processing time (Mukherjee et al., 2017). Furthermore, fog computing may be implemented on a single fog node or on a group of fog nodes working together. This kind of technology may increase scalability while also providing flexibility and redundancy. The number of fog nodes is increased in the event that more computing capacity is needed (Kraemer et al., 2017).

2.3.2 History of Fog Computing

Fog computing was first developed by Cisco in 2012 to solve the difficulties IoT applications in traditional cloud computing (Bonomi et al., 2014). The goal was to promote IoT scalability, i.e., to manage a large number of IoT devices and big data volumes for real-time low-latency applications. The OpenFog Consortium was formed on November 19, 2015, by Cisco Systems, ARM Holdings, Dell, Intel, Microsoft, and Princeton University to promote interests and development in fog computing.

2.3.3 Architecture of Fog Computing

Figure 2.2 depicts the three tiers that make up the fog architecture in order to better explain the distinction between how the fog architecture extends the cloud and how the cloud expands the fog. Tier 1 is comprised of objects/end devices, which may include sensors or mobile devices, among other things. In the fog model, Tier 2 represents the infrastructure components that will process the application logic. These components include gateways, routers, and access points. Tier 2 represents the fog model. Finally, the cloud is represented by Tier 3.

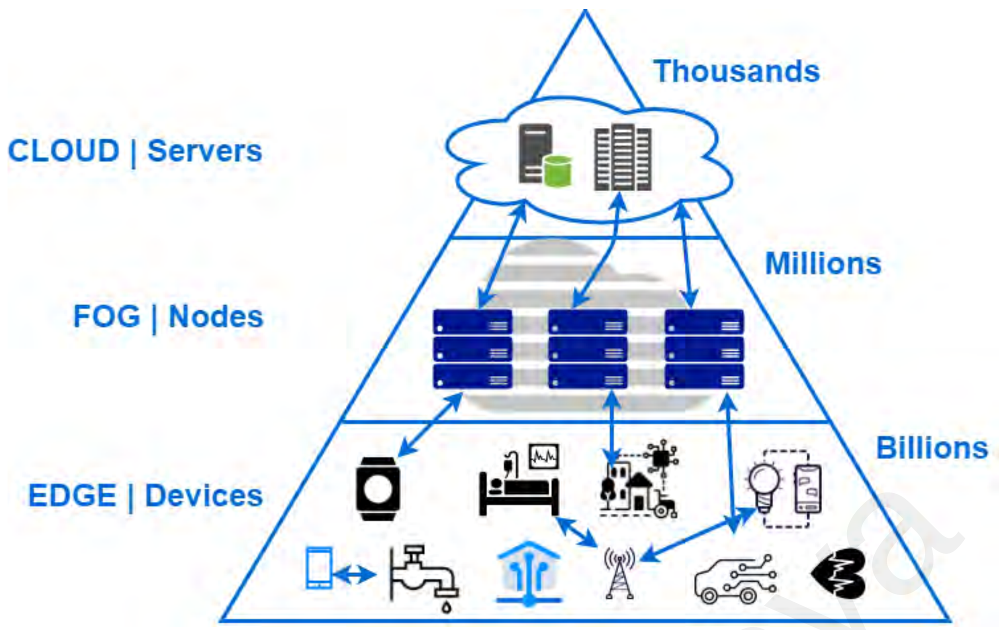


Figure 2.2: Fog based architecture

Despite the fact that today's healthcare systems are heavily reliant on the widespread use of sensors and mobile devices, the fog architecture is by far the most effective architectural style for managing the distribution of the various components involved in the process of tracking and monitoring patients' health data.

2.3.4 Characteristics of Fog Computing

Anawar et al., (2018) describe the following characteristics of fog computing.

Cognition: Cognition is the ability to respond to client-centric goals. Fog based data access and analytics provide a better warning about consumer requirements, as well as the optimum position handling for transmitting, storing, and controlling functions via the cloud to the IoT continuum.

Heterogeneity: Fog computing is a virtualized architecture that provides compute, storage, and networking services between the main cloud and the end devices. Its heterogeneity featured servers consist of hierarchical building components at distributed locations.

Geographical Environment Distribution: Fog network distributes geographically its nodes and sensors in the scenario of different phase environments, for example, temperature monitoring at chemical vats, weather monitoring sensors, and health care monitoring sensors.

Edge Location with Low Latency: The coming out smart applications services are inadequate due to the lack of support at the proximity of devices with featured QoS at the edge of the core network.

Real-time Interaction: Real-time interaction is a requirement of fog computing, like monitoring a patient at healthcare, a critical process in the oil industry with the fog edge devices or sensors, real-time transmission for traffic monitoring systems, electricity distribution monitoring system applications, and so on.

Support for Mobility: Mobility support is a main fog computational merit that can enable direct communication between mobile devices using SDN protocols that decouple host identity from location identity with a dispersed indexing system.

Large Scale Sensor Network: Fog has a capability when an environment monitoring system, in near smart grid applications, inherently extends its monitoring systems caused by hierarchical computing and storage resource requirements.

Widespread Wireless Access: Wireless access protocols (WAP) and cellular mobile gateways are famous instances of fog node closeness to end-users in this situation.

Interoperable Technology: Fog components must be able to work in an interoperating environment to provide support for a large range of services like data streaming and real-time processing for best data analyses and predictive decisions.

2.4 Comparison between Cloud and Fog Computing

When there is constant access to a cloud server capable of processing and transferring data efficiently to the end device, cloud computing is a wonderful option (Prakash et al., 2017). Fog computing is primarily a heterogeneous device architecture in which a smart device manages specific apps and services at the node, while the real administration is handled by the cloud. Mobile users are the primary target of fog computing, whereas general internet users are the primary target of cloud computing. In fog computing, the service type is localized, but in the cloud, it is worldwide. Data and applications are processed in a cloud mean remote places, which is time consuming task for large data, the problem of bandwidth, slow response time and scalability problems in cloud computing. Fog computing, consumes less time, has Less demand for bandwidth, better response time and scalability (Dhivya et al., 2017).

2.5 Internet of Things (IoT)

Internet of Things (IoT) devices enable internet connectivity to be extended beyond the normal standard devices such as PCs, laptops, smartphones, and so on. The sensors and actuators on the Internet of Things are embedded in physical items that are linked through cable and wireless networks, typically utilising the same Internet Protocol (IP) that connects to the internet.

The term ‘internet of things’ has been around for 22 years. However, the concept of linked objects had been known for a long time, at least, since the 1970s. The concept was labelled ‘embedded internet’ or ‘pervasive computing’ at that time. Kevin Ashton, who worked at Procter & Gamble at the time, created the term Internet of Things (IoT) in 1999. Working in supply chain optimization Ashton sought to bring top-level management's attention to a new and innovative technology known as radio frequency identification (RFID). He titled his presentation ‘Internet of Things’ since the internet was the trendiest new technology in 1999, and it made sense. Even though Kevin caught the curiosity of certain P&G executives, the phrase ‘Internet of Things’ didn't spot-on for the following ten years.

2.5.1 IoT Takes-off

In the summer of 2010, the IoT idea began to gain popularity. According to information released, Google's StreetView service not only took 360-degree photos but also saved a huge amount of data from people's WiFi networks. People were arguing if this was the start of a new Google plan to index the real world in addition to the internet.

In the same year, the Chinese government stated that the IoT would be a strategic focus in their Five-Year Plan. In 2011, Gartner, the market research company that invented the famous hype-cycle for emerging technologies, included IoT as a newly emerging phenomenon on their list. In the following year, IoT was the topic of Europe's most important internet conference LeWeb. At the same time, major tech publications such as Forbes, Fast Company, and Wired began adopting the term ‘Internet of Things’ to characterize the phenomena.

According to research issued by IDC in October 2013, the IoT industry would worth $8.9 trillion by 2020. In January 2014, when Google announced that it would purchase Nest for $3.2 billion, the phrase ‘Internet of Things’ became widely acknowledged. The Consumer Electronics Show (CES) in Las Vegas was also themed on IoT at the same time.

2.5.2 IoT has a Very Simple Lifecycle of Development

Deployment is followed by monitoring, service, and management, which is then followed by frequent upgrades and, finally, decommissioning. The IoT product lifecycle is depicted in Figure 2.3

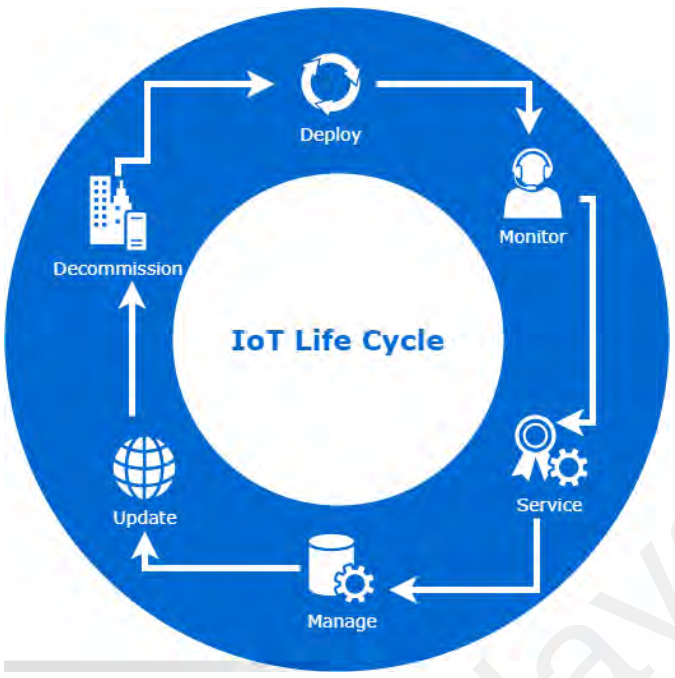


Figure 2.3: Life cycle of IoT device

Apart from these details, there are certain benefits and drawbacks of Internet of Things devices that may have a significant effect on humanity's present and future generations.

2.5.2.1 Benefits of IoT devices

These smart devices have several benefits, some of which are listed below.

- The Internet of Things promotes what is known as machine-to-machine contact between objects.

- It has a high level of automation and control.

- It is better to operate since it is integrated with more technological knowledge.

- IoT has a robust monitoring capability.

- It helps you save a lot of time.

- The IoT helps to save money by eliminating manual tasks and time.

- Automating everyday activities allows for better device monitoring.

- Improved efficiency and time management.

- Good features result in a higher quality of life.

2.5.2.2 Drawbacks of IoT devices

There are several benefits, but there are also some drawbacks. The following are the different demerits:

- There is no worldwide interoperability standard for Internet of Things devices.
- They have the potential to grow very complicated, ending in failure.
- Privacy and security breaches may impact the Internet of Things devices.
- Users' safety is compromised.
- Reduction in the use of physical labor, which results in job losses.
- With the advancement of AI technology, the Internet of Things device may eventually take control of one's life.

2.6 Internet of Medical Things (IoMT)

Internet of Medical Things (IoMT) is a network of devices that offer health-related services via internet (Dilawar et al., 2019). IoMT is a linked health system architecture that includes medical equipment, software applications, and services. More specifically, a link between devices and sensors allows healthcare organizations to improve the efficiency of their clinical operations and workflow management while also monitoring patient health from afar.

To speed up diagnosis, the IoMT combines the digital and physical worlds (Dilawar et al., 2019). And, to enhance the patient's health, the patient's therapies are more precise, and the patient's behavior and health condition are modified in real-time. Patients and doctors will be greatly impacted by the interconnection of medically relevant devices.

2.7 Access Control Model

Access control model (ACM) is the main component of access control architecture. Commonly used access control approaches, e.g., Attribute-Based Access Control (ABAC), Role-Based Access Control (RBAC) and Capability-Based Access Control (CBAC). These devices are not deployed in isolation, rather all three collectively provide a complete solution for securing access to IoT-enabled smart healthcare devices (Pal et al., 2018).

2.7.1 Attribute-based Access Control (ABAC)

It is anticipated that the ABAC approach, which is exemplified by the XACML standard (Erik, 2013), will help to solve the problem of roles explosion by providing the ability to specify access policies directly using subject properties (e.g., age, location, position in an organization, etc.), as well as resources and environmental features. ABAC still requires a standardized definition of the attributes inside a domain or across several domains in order to function properly. There are four elements in ABAC. They are subjects, objects, actions, and environments. It introduces new elements that are called environments. By using the following questions, elements can be retrieved.

Who? = Subjects.

What? = Objects.

How? = Actions.

When? Where? Why? = Environments.

2.7.2 Role-based Access Control (RBAC)

Using access control lists (ACLs), which give particular topics access privileges, is the most popular method of restricting who has access to what information. When the number of subjects and resources grows, it becomes more challenging to manage ACLs. The

RBAC method was developed in order to decrease the cost of basic access control systems (Ferraiolo & Kuhn, 1992; ANSI, 2016). It assigns access rights to roles and subjects to roles in order to reduce the load of simple access control systems. When the quantity of resources and the number of administrative domains increase, this strategy results in an explosion of roles.

2.7.3 Capability-based Access Control (CBAC)

The idea of a capability-based security model is not new (Dennis & Van Horn, 1966; Henry, 1984; Tanenbaum et al., 1995), and it was utilized to develop the RFC2693 specification (Ellison et al., 1999). A capability is a key, ticket, or token that gives the possessor authentication to access a resource or object in a computer system. An intuitive example is that a movie ticket is a capability to watch a movie. In the same way, a key is the capability to enter a house.

The XEROX Casca application (Dirk et al., 2005) and the XPOLA access control system (Skinner, 2009; Fang et al., 2005; Lackorzynski & Warg, 2009) are examples of capability-based security models that have been used in recent years to address both usability issues and rights delegation in grid or service-oriented systems.

Other Access Control Models are also available. For a healthcare medical setting, Wang and Jiang (2015) designed a task-based access control model (T-RBAC). The T-RBAC presents the idea of a task by categorizing it into four types: inherited, non-inherited, passive, and active tasks (Oh & Park, 2000). Liu et al., (2012) looked into the role of context variables in access control policy decisions and developed a context-aware fine-grained access control mode. Zerkouk et al. (2013) have proposed a new adaptive access control concept and architecture. The security policy is based on an analysis of the handicap condition based on user behavior monitoring in order to provide a service utilizing an assistive device inside an intelligent environment. Bernabe et al., (2016) presented a flexible trust-aware access control system for IoT (TACIoT), which offers an end-to-end and dependable security mechanism for IoT devices based on a lightweight authorization mechanism and a new trust model designed specifically for IoT settings. Zemmoudj et al. (2019) presented the revocation feature in the Context-Aware Access Control Model.

2.8 Access Control Architectures

Access control is essential and a prerequisite for governing and safeguarding information assets inside an organization's infrastructure. Organizations generally use web-enabled remote access in conjunction with application access distributed across multiple networks, which presents several challenges, including increased operation burden, monitoring issues due to the dynamic and complex nature of security policies for access control, and a lack of consistency across networks (Uddin et al., 2019). There are two kinds of access control: physical access control and logical/computer access control are shown in Figure 2.4.

Physical access control is the most common type. Physical access control restricts access to facilities such as campuses, buildings, rooms, and physical information technology assets. Physical access control may be accomplished by a person (a guard, bouncer, or receptionist), by mechanical means (such as locks and keys), or by technology methods (such as access control systems) all at the same time. Physical access control is a function of who, where, and when someone enters a building. An access control system decides who is permitted to enter or depart a building, where they are permitted to leave or enter, and when they are permitted to enter or exit a building. Historically, this was achieved in part via the use of keys and locksets. When a door is locked, only someone who has the key to the door can enter through it, depending on how the lock is programmed to operate. Mechanical locks and keys do not enable the key holder to limit the use of the key to particular hours or dates. Technically speaking, there is no way for mechanical locks and keys to track down which key was used on which door, and keys may be readily duplicated or handed to an unauthorized individual. Mechanical keys that have been lost or whose holders are no longer allowed to use the protected area must be re-keyed in order to prevent the protected area from being accessed by others. Electronic access control and credential-based physical access control systems are the two most common types of physical access control systems available on the market.

Computer networks, system files, and data are all restricted via the use of logical access control (LAC).

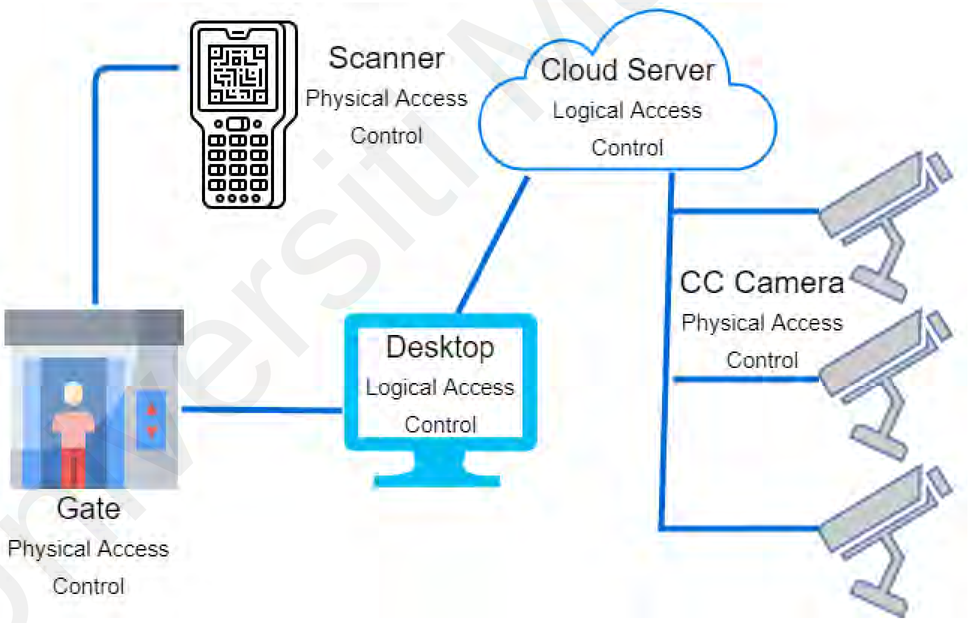


Figure 2.4: Physical and logical access control

In order to reduce the security risk associated with illegal access to physical and logical systems, access control must be implemented. A key component of security compliance systems, access control ensures that the appropriate security technology and access control rules are in place to safeguard sensitive information, such as client data, from unauthorised access. Access to networks, computer systems, applications, files, and sensitive data, such as personally identifiable information (PII) and intellectual property, is often restricted by infrastructure and processes in most companies.

It is possible to implement these security measures by identifying a person or entity, confirming that the person or application is whoever or what it claims to be, and allowing the access level and set of activities associated with a username or Internet Protocol (IP) address. Directory services and protocols, such as the Lightweight Directory Access Protocol (LDAP) and the Security Assertion Markup Language (SAML) or the XACML, provide access controls for authenticating and authorizing users and entities, as well as enabling them to connect to computer resources, such as distributed applications and web servers, to which they have been granted access.

Access control is a procedure that is incorporated into the information technology environment of a business. Identification and access management systems are two examples of what may be included. These systems include access control software, a user database, and administrative tools for enforcing access control rules, auditing access control policies, and auditing access control policies.

As soon as a user is introduced to an access management system, system administrators may employ an automated provisioning system to assign rights based on access control frameworks, job responsibilities, and processes to that user.

In accordance with the principle of least privilege, employees should only have access to resources that are necessary for them to execute their principal job duties.

2.8.1 Existing Access Control Architecture in IoT

Many research papers have been published in recent years that address these security and latency problems in IoMT in order to satisfy future prospective IoMT security needs.

Unauthorized access is the most serious security issue in IoMT (ITSecurityGuru, 2016). Access control is one of the most important security methods that should be used in healthcare to ensure that only authorized people have access to patient data (Alnefaie et al., 2019). Authentication, authorization, and accountability are the three tasks covered by a comprehensive access control system (Suhendra, 2011). The proposed research's primary focus is on the authorization. The authorization process consists of the following steps (Ouaddah et al., 2017): a) Create a security policy (a collection of rules), b) Choose an access control model to encapsulate the policy, and c) Put the model into action. Figure 2.5 shows the different steps of the authorization system.

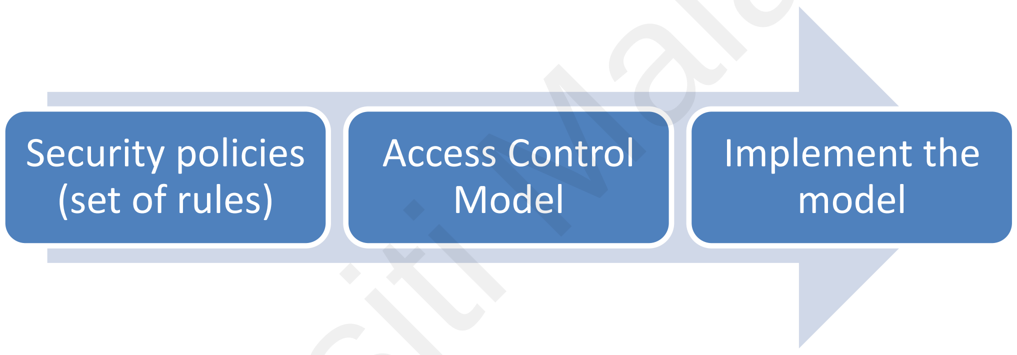


Figure 2.5: Authorization system

Most of the time, ACA is managed in either a completely centralized or a distributed fashion (Hussein et al., 2017). There have been many research papers released on cloud-based centralized ACA. Smithamol et al. (2017) developed and built a new cloud-based security architecture G-CP-ABE based on the attribute-based model, which was subsequently validated. Ungurean et al. (2014) presented an IoT architecture for things from an industrial setting based on cloud computing without developing any access control mechanism, just by providing the architectural model as a starting point. Bitcoin used the idea of blockchain as a new solution for cryptocurrencies in a decentralized way (Nakamoto, 2009), which was the first in the cryptocurrency industry. The signature of a peer ensures the integrity of the information, and any peer may verify any transaction by using the public key of the peer who conducted the transaction, ensuring that the information is accurate. As a consequence, blockchain includes every transaction that has occurred throughout the course of time, and as a result, it is a heavyweight solution. Several other researchers are also investigating access control solutions that make use of blockchain technology (Le-Dang & Le-Ngoc, 2019; Putra et al., 2019; Sabrina, 2019; Sivaselvan et al., 2020; Xu et al., 2019).

Pal et al. (2019) employ a hybrid method for authorization design, using attributes, responsibilities, and capabilities to provide higher performance than an individual model while incurring just a little increase in time overhead. In Aftab et al. (2016), the authors have presented a novel model that combines the advantages of both ABAC and RBAC in order to reap the benefits of both models while also addressing their shortcomings.

Cisco has developed a new fog technology to help decrease latency and throughput in its networks. Access control for healthcare is being dispersed via a fog-based architecture, which will move the decision-making and policy-information processes to the edges of the network, close to the end nodes. This technique improves data availability while also decreasing latency. Alnefaie et al. (2019) had developed a novel attribute-based access control architecture based on fog computing without the need of any physical devices. The researchers shift the design of access control from a centralized to a distributed one.

Specifically, this study lowers data fetching time by designing and implementing a new architecture that is primarily based on ABAC as a worldwide standard AC model (Hu et al., 2014) and combining it with other RBAC and CapBAC models to achieve improved performance.

2.8.2 eXtensible Access Control Markup Language

It is an OASIS standard that specifies a general-purpose access control and authorization system (often known as an XACML system) (Oasis, 2010). It is made up of two parts: a policy language based on XML and a processing system that understands how to interpret the policy in the context of the application. Using policy language, administrators may build policies, with each policy enumerating the criteria for accessing a resource inside a protected environment.

The data-flow diagram of XACML is shown in Figure 2.6, which is presented below, depicts the essential components of the XACML domain. This mode is operated using the following rules/steps:

- The primary responsibility of the PAP is to create policies and policy sets and make them available to the PDP. These policies, or policy sets, represent the total policy of an organization.
- The requester sends an access request to PEP.
- The PEP receives the request and delivers it to the context handler in its native request format, with the subject, resource, action, environment, and other category attribute optionally added.
- An XACML request context is created by the context handler. Adds attributes as needed and sends them to the PDP.
- The PDP asks the context handler for any extra subject, resource, action, environment, and other categories (not displayed) attributes.
- The context handler asks for the attributes to the PIP.
- The desired attributes are obtained by the PIP.
- The PIP sends the required attributes to the context handler.
- The context handler may optionally include the resource in the context.

- The PDP receives the required attributes and (optionally) the resource from the context handler. The policy is assessed by the PDP.

- The response context (containing the permission decision) is returned to the context handler by the PDP.

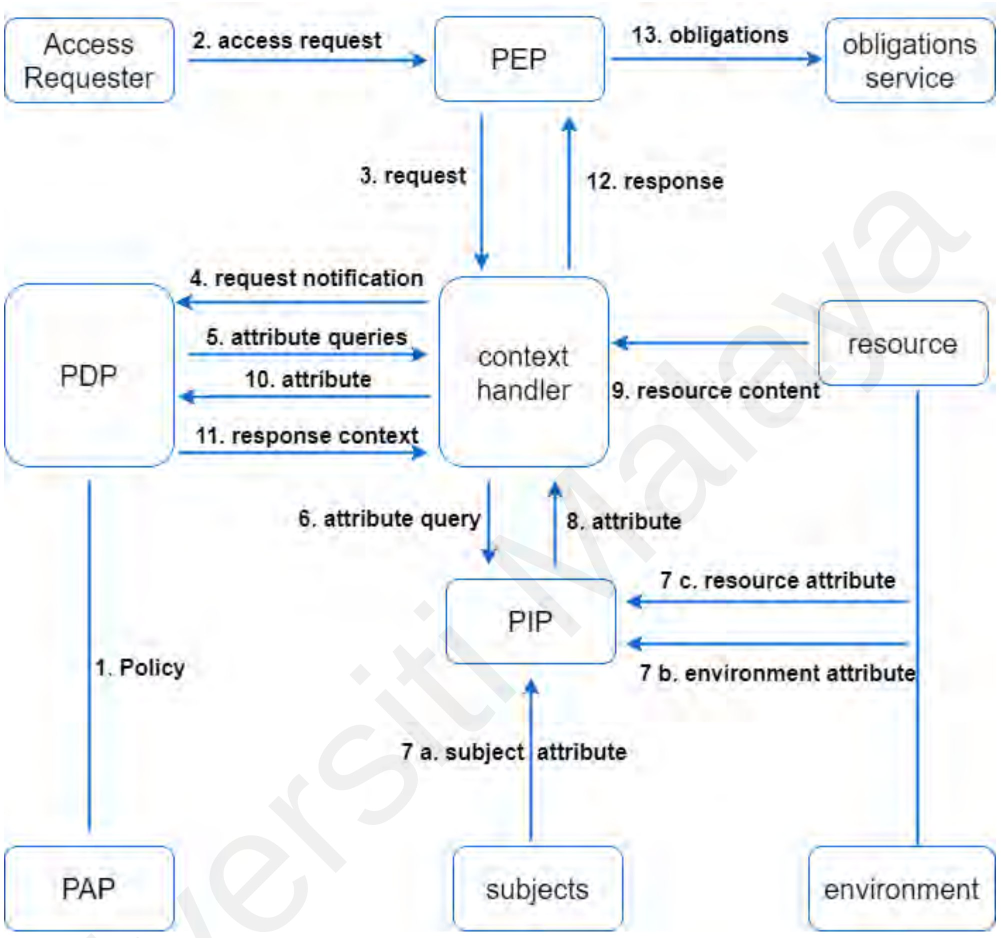


Figure 2.6: XACML data-flow diagram. Erik, R. (2013)

The response context is translated to the PEP's native response format by the context handler. The response to the PEP is returned by the context handler.

- The PEP performs its responsibilities.

- (Not displayed) If access is allowed, the PEP allows access to the resource; otherwise, access is denied.

XACML is designed to work in a wide range of application settings. The XACML context isolates the core language from the application environment, as illustrated in Figure 2.7, where the shaded region indicates the scope of the XACML standard. The XACML context is described by an XML schema that specifies a standard form for the PDP's inputs and outputs. Attributes referenced by an XACML policy instance may be expressed as XPath expressions over the context's <Content> components, or as attribute designators that provide the attribute's category, identifier, datatype, and (optionally) issuer. Implementations must transform attribute representations in the application environment (e.g., SAML, J2SE, CORBA, and so on) to attribute representations in the XACML context. The XACML standard does not explain how this is accomplished.

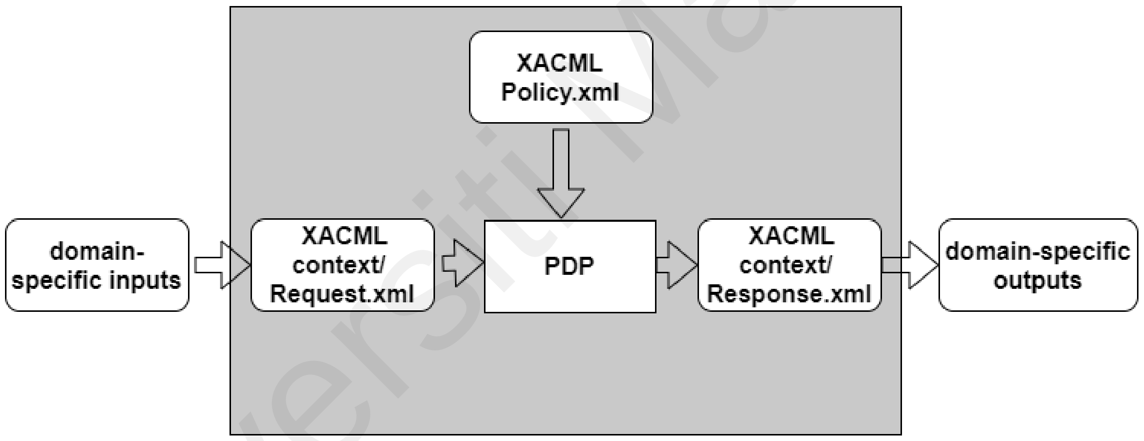


Figure 2.7: XACML context. Erik, R. (2013)

It is notable that PDP is not needed to work directly with a policy's XACML representation. It may be able to act on a different representation directly. The subject, resource, action, and environment are common types of attributes in the context, although users may create their own as required.

2.8.3 Summary of Reviewed ACA

Table 2.1: Summary of existing ACA in IoT

Title	AC Type	Features	Processing Time	Limitation	Authors & year
Policy-Based Access Control for Constrained Healthcare Resources	Combined ABAC, CBAC, RBAC	Partially Distributed Architecture	706.09ms	More Time Consuming	Pal et al., (2019)
A Lightweight Attribute-Based Access Control System for IoT	Attribute-Based Access Control	Cloud-Based Architecture	651ms	Only one model	Monir (2017)
Context-aware pseudonymization and authorization model for IoT-based smart hospitals	Context-aware ACM.	Context-aware pseudonym for reducing unauthorised access & also protect the location.	5ms to 14s	More Time Consuming and Storage overhead.	Zemmoudj et al., (2019)
Scalable Blockchain-based Architecture for Massive IoT Reconfiguration.	Policy-Based ACM	Blockchain-based Access Control Architecture	8.5s	More Time Consuming	Quang Le-Dang and Tho Le-Ngoc (2019)
Towards a Distributed Access Control Model for IoT in Healthcare	ABAC	A decentralized ABAC architecture by using fog technology	Not implemented	It was not implemented.	Alnefaie et al., (2019)

Table 2.1, continued

Title	**AC Type**	**Features**	**Processing Time**	**Limitation**	**Authors & year**
Hybrid Solution for Privacy-Preserving Access Control for Healthcare Data	ABAC	Merge symmetric encryption and CP-ABE scheme to minimize the overall encryption time	120ms to 830ms	Attribute-Based Encryption	Bhaskaran and Sridhar, (2017)
An efficient access control scheme with outsourcing capability and attribute update for fog computing	CP-ABE scheme	CP-ABE scheme	740ms (Encrypt, Decrypt & Key)	Only ABAC	Zhang et al., (2018)

2.9 Challenges in Access Control Architecture

Issues related to privacy, authentication, access control, system setup, information storage and management, and other security concerns are the primary difficulties that every IoT ecosystem must deal with (Ouaddah et al., 2017). Access authorization is one of the most significant security issues that IoTs confront, and it is essential in resource and information protection (Abu Bakar & Jamal, 2020).

When it comes to size, structure, and the number of users and resources (Ouaddah et al., 2017), the authorization method should be scalable. In the case of inflexibility, the access control system must be adaptive to various situations. Furthermore, it should be able to accommodate both long-term and planned models, as well as spontaneous and short-term causal relationships (Ouaddah et al., 2017).

When it comes to IoT systems, traditional access control approaches such as ACL, RBAC and ABAC do not provide scalable, manageable, and efficient mechanisms that are suitable for the demands of the Internet of Things. The very high number of nodes, heterogeneity, and dynamicity of the Internet of Things requires the development of fine-grained, lightweight methods for IoT devices (Chen et al., 2018).

Access control architecture (ACA) is now one of the most challenging obstacles to the widespread adoption of IoT (Uriarte Itzazelaia et al., 2018), and their availability is one of the most significant obstacles. More importantly, because of the high level of dynamicity and the aim of applications based on services in sensors, policy-based security must be implemented locally in constrained device sensors, where resources are limited (Uriarte Itzazelaia et al., 2018).

2.10 Summary

This chapter describes the access control architecture with main components; IoT, cloud computing, fog Computing, XACML, and access control models. Also, an overview of the existing access control architectures and their relative features are presented. Finally, the chapter concludes with the description of the challenges of an access control architecture.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Introduction

This section discussed the applied research methodology. This research methodology is accomplished by three phases are shown in Figure 3.1. In phase one, the research problem is analyzed, and the corresponding literature is reviewed and discussed. This phase includes problem identification, research objectives, review of selected literature such as an overview of IoT, ACA, ACM, cloud computing, fog computing, challenges of access control architecture, and classification of ACM, pros, cons, and review of selected ACM in ACA and analyzed the comparison between the existing ACA. In phase two, designed and developed the T-TAFAC, which is the solution for the research problem. This phase is structured in five different steps such as setting up the environment, write the program for NodeMCU to communicate with the server, database design based on XACML, write the program for fog node to communicate with IoT and cloud, and maintenance. Finally, phase three provides the results and analyzed the developed architecture, where its performance is evaluated against the existing access control architecture.

3.2 Problem Identification

In this phase, the features and challenges of the access control are studied and discussed. Access control architecture consists of IoT, cloud, fog, and access control models. ACA comes with some challenges. In the healthcare industry, latency reduction is one of the significant challenges in ACA. In contrast, for a large organization, access policy management is also a challenge. To reduce this problem, the researcher has proposed many access control architectures that have been reviewed in the literature review section.

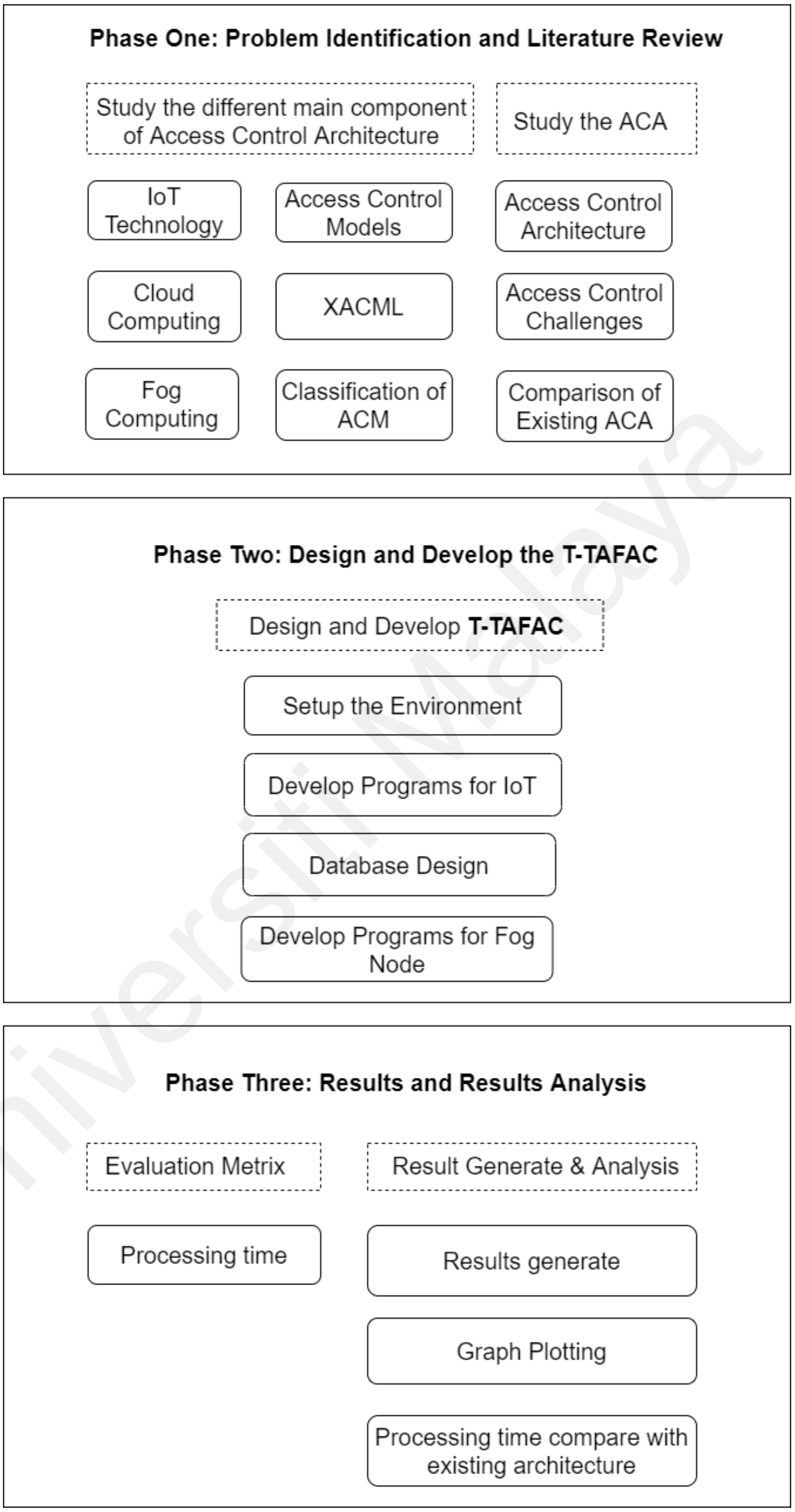

Phase One: Problem Identification and Literature Review
Study the different main component of Access Control Architecture
Study the ACA
IoT Technology
Access Control Models
Access Control Architecture
Cloud Computing
XACML
Access Control Challenges
Fog Computing
Classification of ACM
Comparison of Existing ACA
Phase Two: Design and Develop the T-TAFAC
Design and Develop T-TAFAC
Setup the Environment
Develop Programs for IoT
Database Design
Develop Programs for Fog Node
Phase Three: Results and Results Analysis
Evaluation Metrix
Result Generate & Analysis
Processing time
Results generate
Graph Plotting
Processing time compare with existing architecture

Figure 3.1: Research methodology

ACM is one of the main components of ACA. There are many ACMs and model has its own pros and cons. The researcher proposed different ACA by using different access control models with some drawbacks. Some researchers proposed cloud-based, some researchers proposed blockchain-based access control architecture, and some researchers proposed the access control process inside the constraint IoT devices, all are taking more time at the processing and communication. Already proposed architecture also have their drawbacks such as the selection of access control model, selection of cloud-based/fog-based, selection blockchain, etc. The following, shown and described the existing two access control architectures which are the benchmark architectures of this research.

In Figure 3.2**,** this ACA was designed cloud-based by Pal et al (2019). Processing within devices and the cloud was used for the authorization procedure. As a result, the processing time for authorizations was increased.

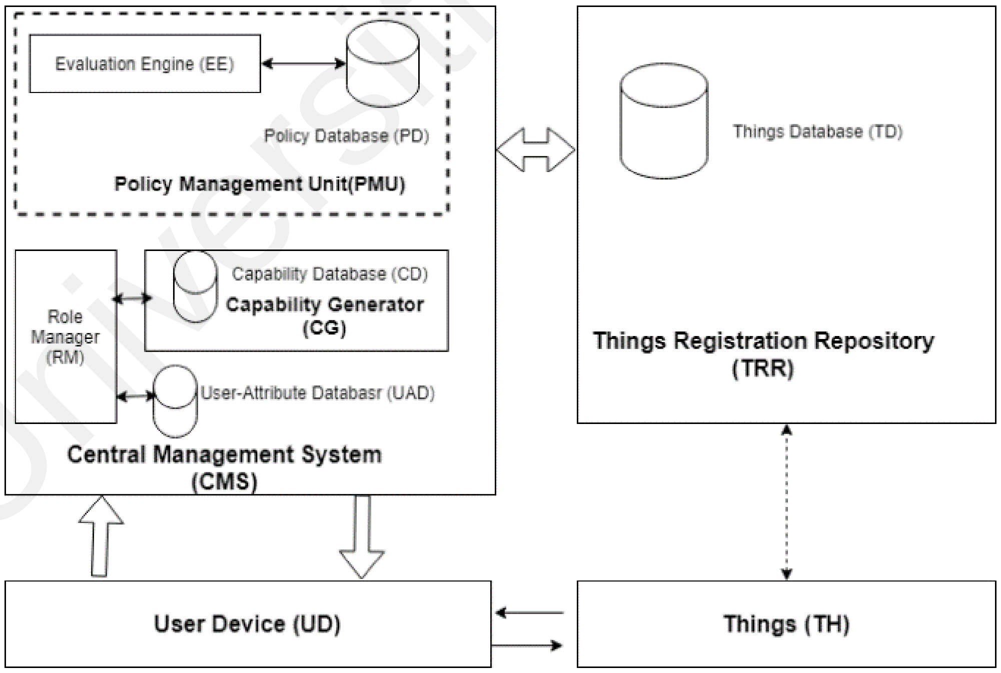

Evaluation Engine (EE)
Policy Database (PD)
Policy Management Unit(PMU)
Role Manager (RM)
Capability Database (CD)
Capability Generator (CG)
User-Attribute Databasr (UAD)
Central Management System (CMS)
Things Database (TD)
Things Registration Repository (TRR)
User Device (UD)
Things (TH)

Figure 3.2: Pal et al., (2019) partial distributed ACA

In Figure 3.3**,** Monir (2017) ACA was also cloud-based. For authorization, the request goes to the cloud every time. Same as Pal et al. (2019) ACA, it also takes more time.

Therefore, a new solution of access control architecture has been proposed where the ACM is the combination of the three most popular ACM like as attribute-based, role-based and capability-based access control models. Designed architecture is introduced fog computing in the middleware to reduce the latency.

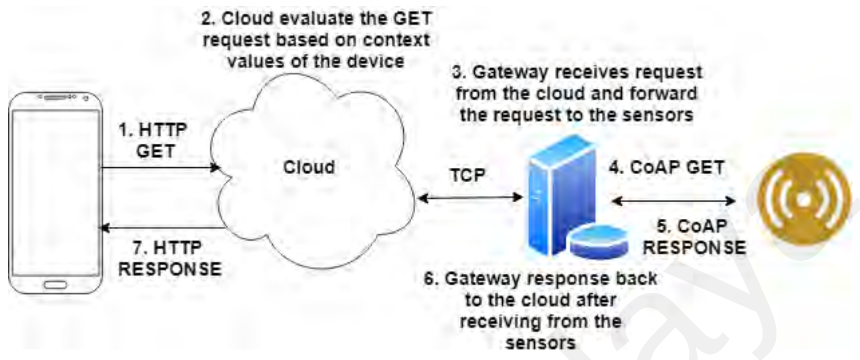

2. Cloud evaluate the GET request based on context values of the device
3. Gateway receives request from the cloud and forward the request to the sensors
1. HTTP GET
Cloud
TCP
4. CoAP GET
5. CoAP RESPONSE
7. HTTP RESPONSE
6. Gateway response back to the cloud after receiving from the sensors

Figure 3.3: Monir (2017) cloud-based ACA

3.3 Design and Development of the T-TAFAC

This phase involves designing and developing the three-tiered architecture. All three tiers' descriptions are discussed in the next following subsections.

3.3.1 Tier 1: Device / Edge Node

The first tier is the perception/device/sensor tier. All kinds of IoT devices/sensors and user devices are included in this tier. Like mobile, desktop, tab, smartwatch, smart thermometers, wearable biosensors, connected inhalers, ventilators, oximeters, defibrillators, pacemakers, neurostimulators, respiratory care devices, and so on. These IoT devices are connected to the Fog tier, where the primary data are stored. That name is Central Management System (CMS).

Figure 3.4 depicts the T-TAFAC. It is composed of: Role Manager (RM), Policy Enforcement Point (PEP), Policy Decision Point (PDP), Policy Information Point (PIP), Policy Administration Point (PAP), Things Registration Repository (TRR), Capability

Engine (CE)/ Capability Generator (CG), User-Attribute Database (UAD), Policy Database (PD), Capability Database (CD), Hospital Management System Database (HMS).

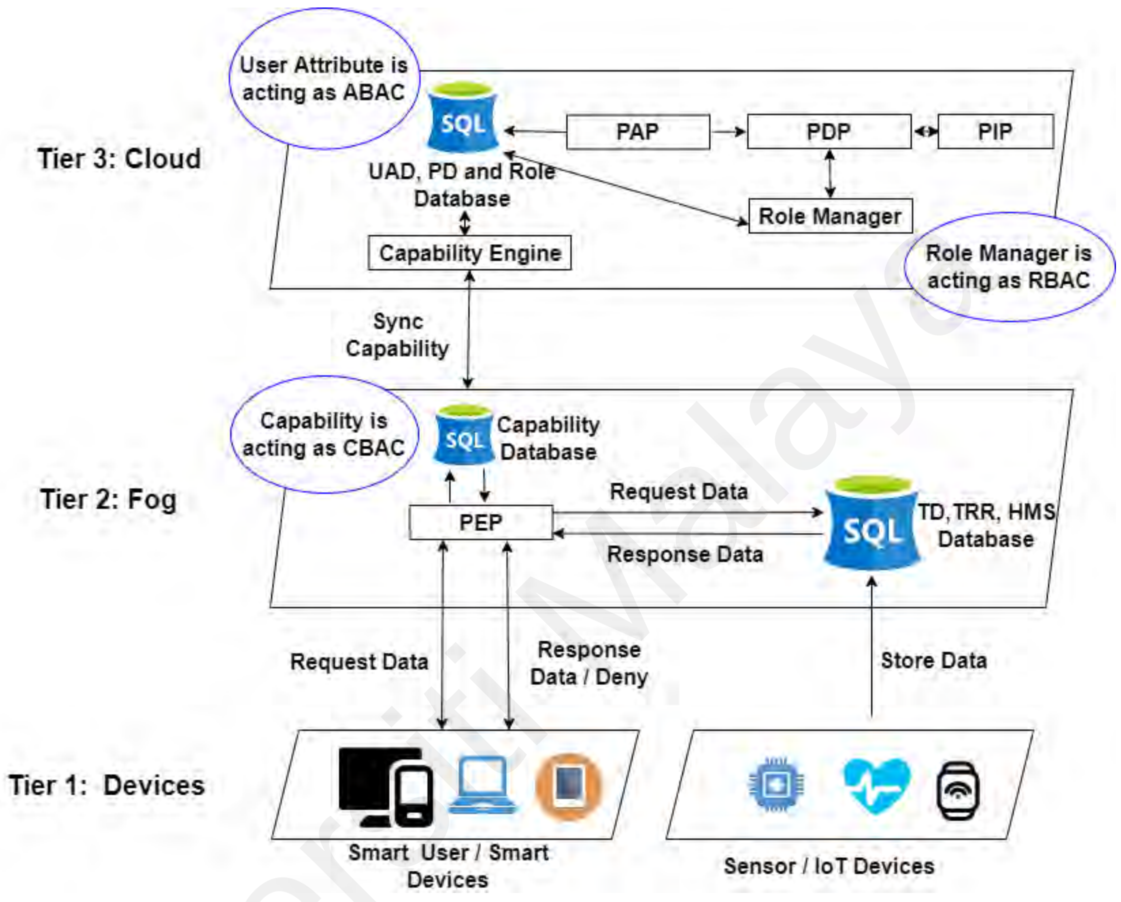


Figure 3.4: Three-tiered architecture for access control

When a patient comes to the hospital for the first time, a patient needs to register to HMS. For the first time, things are assigned to a patient. Some information like things secret key, patient id is inserted into the database, and insert that information inside the thing. Things basic information are store in one data table that is called Things Registration Repository (TRR). TRR creates the link between things and patients. So, things communicate with the central database (HMS Database) by using the secret key. Data is encrypted by that secret key, and in the central system, those data decrypt using that same secret key. Communication becomes an extra layer of security. And all communication protocols are HTTPS for basic data protection. It prevents Man-in-the-middle (MITM) attacks. One of the most popular attacks is Man-in-the-middle (MITM). The MITM attack involves the introduction of an anonymous receiver and transmitter into the original transmitter and receiver system, as shown in Figure 3.5**.** The anonymous attacker attempts to alter or disrupt the transmission or reception of data between a transmitting and receiving device with the aim of obtaining illegal information.

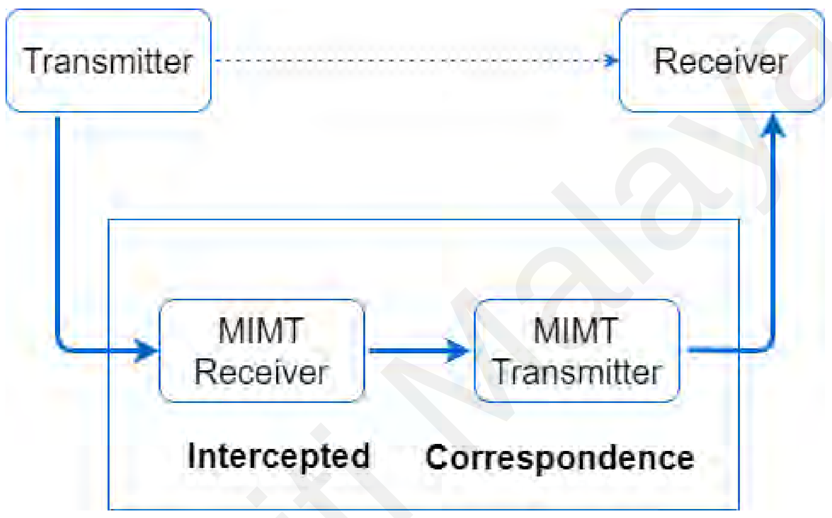


Figure 3.5: MIMT attack

To store patients' data in this testbed, one table is designed. This table creates links between patients and things. On the other side is user devices. Medical personnel, patients, and guardian need to access patient's data. Nowadays mobile, tab, and laptop is a handy device. They use these devices for this purpose. Like the thing's communication, this communication also secures by using HTTPS protocol and encrypted the data. Every user has a secret key. Those keys are used to encrypt all data. User password used as a secret key. This tier sends a request to the fog tier to access patient's information. The fog tier takes the necessary steps and sends back the response. It can be positive or negative. If the user has the right permission, then its response is a positive otherwise negative response.

3.3.2 Tier 2: Fog

The middle tier is called the fog tier. Central management system (CMS) is the main part of this tier. Two primary databases are in this tier: one is the capability database where all user's capabilities are stored in this table. Figure 3.6 shows the structure of the capability table.

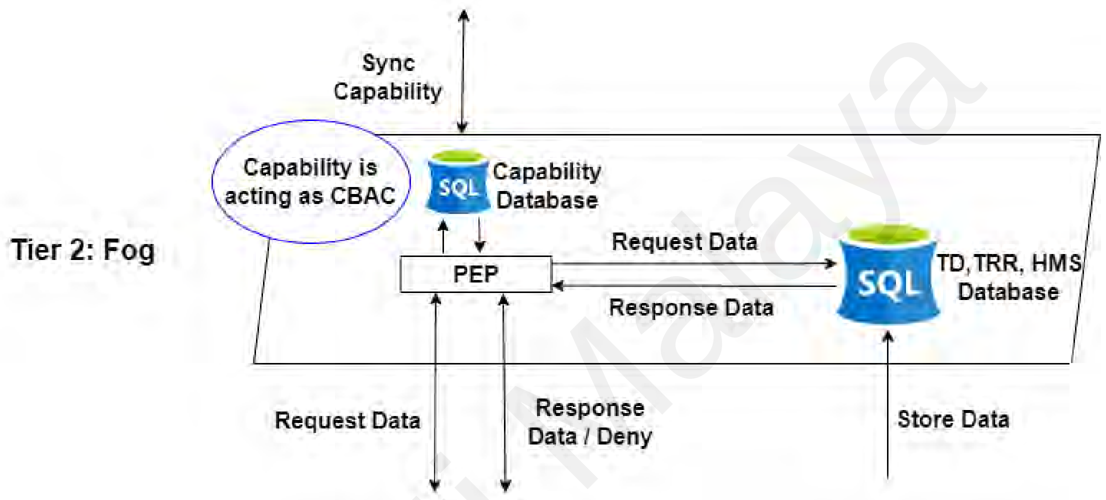


Figure 3.6: The fog tier of T-TAFAC

3.3.2.1 Capability data table structure

$$Cap = \{Cap_{id}, U_{user\_id}, Iss_{id}, Start_{time}, Exp_{time}\}$$

Table 3.1 describe the definition of all notations of capability data table.

Table 3.1: Notation and definition of capability data table structure

Notation	**Definition**
Cap	Capability.
Cap_{id}	Capability Identity. This capability id is unique. Every capability has this unique number.
U_{user_id}	It stores the user identity. It links with the user information data table. It indicates which user owns this capability.

Table 3.1, continued

Iss_{id}	Issuer identity. Who issues this capability? So that anyone can easily understand who created this capability.
$Start_{time}$	Capability start time. This field indicates the starting time of the capability.
Exp_{time}	This is the expiration time. It gives the boundary of the capability.

This is the crucial table in this authorization system. It is synchronized with the capability engine at a specific period. Capabilities are automatically generated before starting the duty of the medical personnel. It takes the responsibility that a user has permission or not to access information. It communicates with the capability engine, HMS, and user. All communications are secured by encrypting data and HTTPS protocol. The other database is Hospital Management System. This is the central database for the hospital. In the actual situation, it can be an ERP system. Enterprise resource planning (ERP) is the integrated management of main business processes, often in real time and mediated by software and technology. Payroll, HRM, Accounts, CRM, Pathology, Inventory, and so on are included in this database. But in this testbed, only few tables are created that are required for this research. User information, Things/IoT device information, Thing's data tables are in this database.

3.3.2.2 User-attribute database (UAD)

It stores user information.

$U = \{U_{user\_id}, U_{user\_name}, U_{user\_password}, U_{user\_date\_of\_birth}, U_{user\_contact\_number}, U_{user\_address}, U_{user\_role\_membership}, U_{user\_created\_date}, U_{user\_updated\_date}\}$

Table 3.2 describe the definition of all notations of user-attribute database.

Table 3.2: Notation and definition of user-attribute database

Notation	**Definition**
U	User.

U_{user_id}	This is the identity key field of the data table. Every user has unique user id.
U_{user_name}	This is the username of the User. Like username Mushfiqur Rahim.
$U_{user\_password}$	This field keeps the value of the secret key. This key is used for encryption.
$U_{user\_date\_of\_birth}$, $U_{user\_contact\_number}$, $U_{user\_address}$	These are the user personal information.
$U_{user\_role\_membership}$	This is a very important field. It holds the role of the user. It reduces the number of policies.
$U_{user\_created\_date}$, $U_{user\_updated\_date}$	These keep the record's creation date and updated date to easily manipulate the user information.

3.3.2.3 Thing's registration repository (TRR)

It holds the things' information.

TRR = {$T_{thing\_user\_id}$, T_{thing_name}, $T_{thing\_secret\_key}$, $T_{thing\_location}$, $T_{thing\_function}$, T_{thing_status}, $T_{thing\_description}$, $T_{thing\_created\_date}$, $T_{thing\_updated\_date}$}

Table 3.3 describe the definition of all notations of Thing's registration repository.

Table 3.3: Notation and definition of TRR

Notation	Definition
TRR	Things Registration Repository. It stores the things information.
$T_{thing\_user\_id}$	This is the user id field of the data table. It creates links to the user. By using this, we can recognize which users are using this thing.
T_{thing_name}	This is the name field of the Things. Like things name Pacemakers.
$T_{thing\_secret\_key}$	This field keeps the things secret key. This key is used for encryption.
$T_{thing\_location}$	It stores the location of the things.
$T_{thing\_function}$	Things functionally are store in this field.

Table 3.3, continued

Notation	Definition

T_{thing_status}	We can easily understand is the thing is active or not by storing the status in this record.
$T_{thing\_description}$	Thing's descriptions are store in this field.
$T_{thing\_created\_date}$, $T_{thing\_updated\_date}$	These all are the same as things registration table.

3.3.2.4 Things datastore (TD)

It holds the things data of a patients.

$$TD = \{TD_{thing\_id}, TD_{thing\_data}, TD_{thing\_data\_created\_date}, TD_{thing\_data\_updated\_date}\}$$

Table 3.4 describe the definition of all notations of Things datastore.

Table 3.4: Notation and definition of Things datastore

Notation	**Definition**
TD	This is the Things Database table. Patient's medical stores are in this table.
TD_{thing_id}	This is the link field of the data table. It creates a link with the Things Registration Repository table.
$TD_{thing\_data\_created\_date}$, $TD_{thing\_data\_updated\_date}$	These all are the same as things registration table.

This tier communicates with the cloud tier to synchronize the capability or first time to create capability for a user.

3.3.3 Tier 3: Cloud

In the designed system, the cloud tier consists of the capability engine in Figure 3.7. In the following, describe the capability engine:

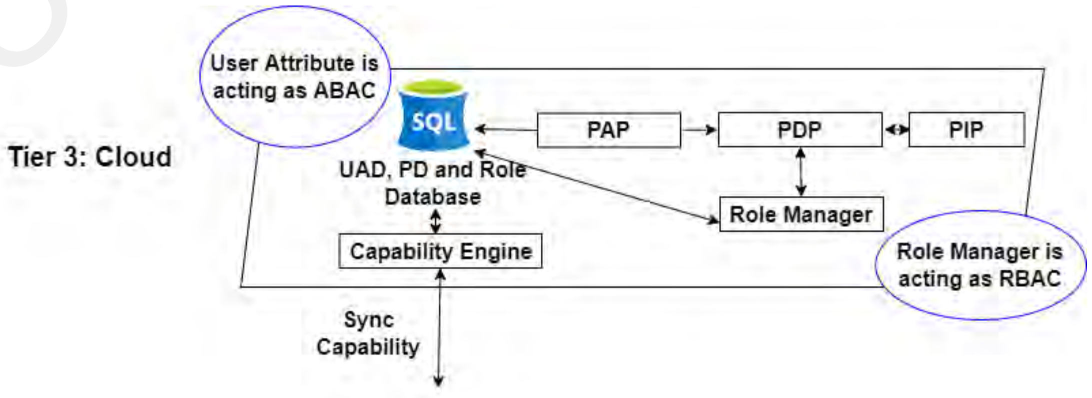

User Attribute is acting as ABAC
Tier 3: Cloud
SQL
UAD, PD and Role Database
PAP
PDP
PIP
Role Manager
Capability Engine
Role Manager is acting as RBAC
Sync Capability

Figure 3.7:The cloud tier of the designed T-TAFAC

Capability Engine: The CE is responsible for generating capabilities for positive XACML responses received from the PEP. Negative reactions will be reported as an authorization failure to the requesting user. It may require communication with the TRR, to recover any TH attributes necessary (e.g., in capability parameterization).

XACML is used to drive the design of this capability generator (eXtensible Access Control Markup Language). This design employs attributes for role administration, capabilities for access right implementation, and attributes for fine-grained policy choices based on capabilities are all included in this architecture. Access rights are embodied in capabilities. Users may request capabilities, which are then made available to them depending on their characteristics and the roles that those attributes have granted them participation in. Once a capacity has been acquired, the user attributes do not need to be verified again as long as the capability is still in effect. It is necessary to go to the cloud for the first time. And the next time, get access directly from the fog node.

As shown in Figure 3.8, the main components of the XACML Standard are as follows:

Policy Administration Point (PAP): is responsible for the creation and management of all policies. Users' requests are intercepted by the Policy Enforcement Point (PEP), which is responsible for enforcing XACML decisions received from the Policy Decision Point (PDP). The Policy Decision Point (PDP) is accountable for assessing users' requests in accordance with current policies and returning XACML choices to the Policy Execution Point (PEP). Finally, the Policy Information Point (PIP) enables the collection of additional characteristics about a person.

Policy Enforcement Point (PEP) is an engine that converts queries into XACML requests that are syntactically valid. By integrating the previously recorded user attributes (from the UAD), service request details, and contextual data, such as timestamps and other information about the targeted resources, it is possible to obtain the desired result.

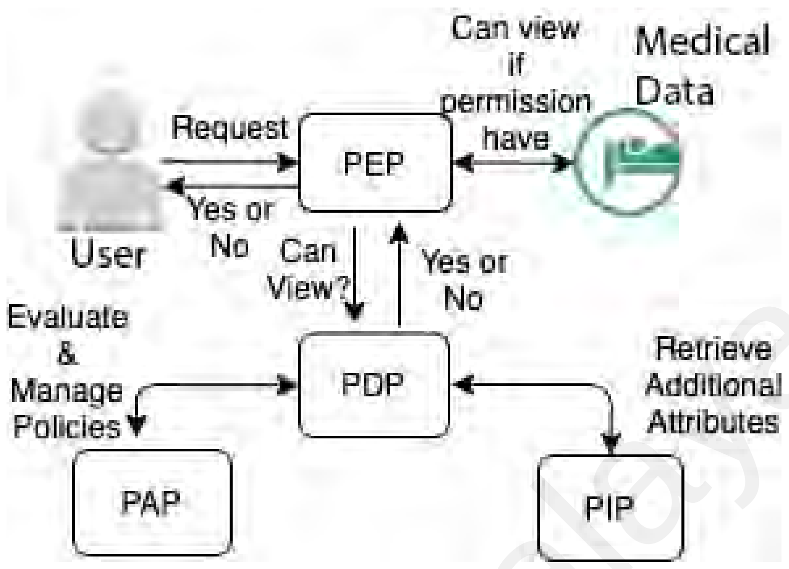

Can view if permission have
Medical Data
Request
PEP
Yes or No
User
Can View?
Yes or No
Evaluate & Manage Policies
PDP
Retrieve Additional Attributes
PAP
PIP

Figure 3.8: XACML main components. User request for medical data. XACML schema check the authenticity. If user has permission to view, then user can view otherwise cannot view the medical data.

Policy Decision Point (PDP): This engine receives the XACML request from the Policy Execution Point (PEP) and compares it against the access control rules in effect. A consultation with the PD is conducted by the PDP in order to establish which role(s) has access to the requested resource. The attribute policy that controls membership in a role is stored with the role and may be accessed via the role. This is also the moment at which the policy is retrieved. Following that, the assessment process compares the policy to the characteristics specified in the XACML request.

Policy Information Point (PIP): Policies are connected to underlying sources of attributes via the Policy Information Point (PIP), which links the PEP (and potentially the PDP) (i.e., the attribute values). In the PEP engine, which interacts with the UAD and the TRR, this capability may be provided automatically without the need for additional code.

Policy Administration Point (PAP): The Policy Administration Point (PAP) is responsible for the creation, management, and editing of the authorization policy or policy sets. It may be an external application that is intended to make the administration of policies more convenient and secure.

Role Management is a new component in our suggested design, which we believe would be beneficial.

Role Management: The RM is in charge of managing roles and responsibilities. This module is in charge of mapping role membership to characteristics depending on the values of the attributes. The jobs are naturally arranged in a hierarchical fashion. This is especially essential when drafting a policy that should be passed down to higher-level positions (e.g., a doctor can access everything a nurse is allowed to access).

3.3.3.1 Model of the XACML policy language

The policy, rule, and policy set are the three major components of the XACML policy language model is shown in Figure 3.9.

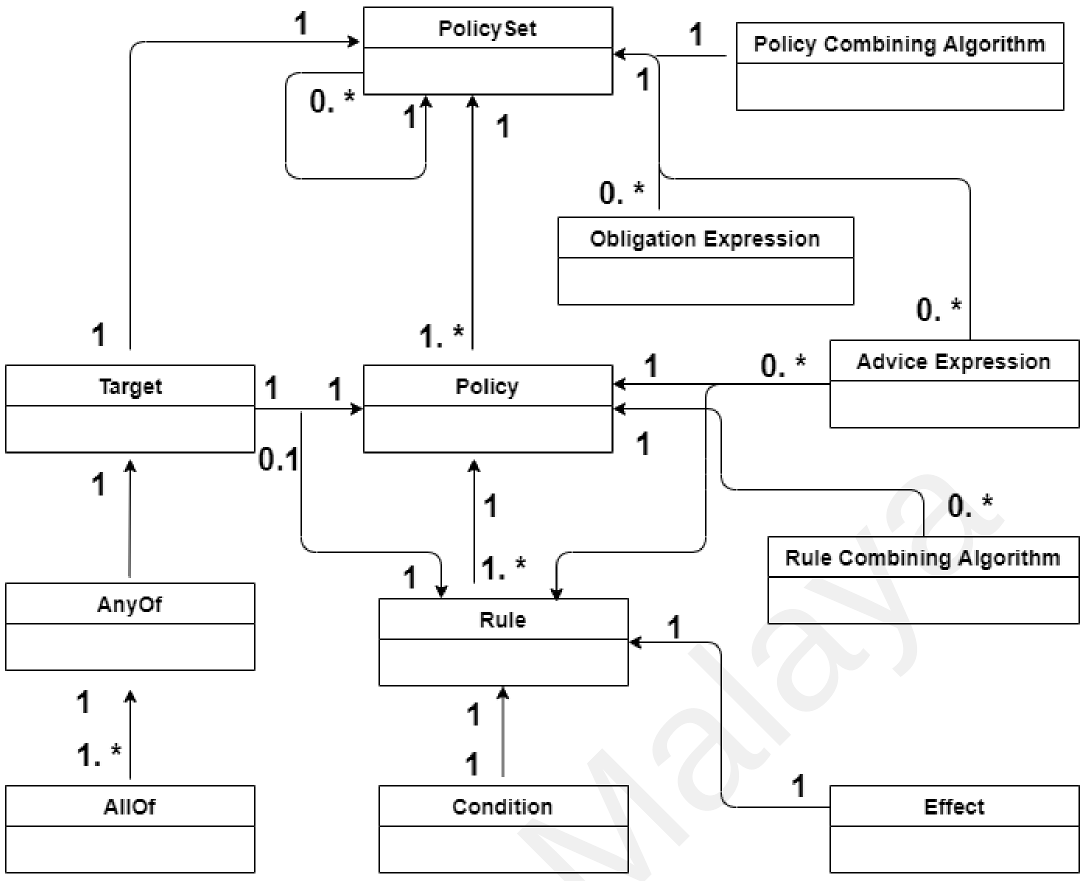


Figure 3.9: Policy language model. Erik, R. (2013)

In the following, describe the data structure and the relationship of those components with their sub-components.

(a) The policy data table structure:

P = {P_{id}, $P_{version}$, $P_{description}$, $P_{defaults}$, P_{target_id}, $P_{rule\_combining\_algid}$, P_{issuer_id}, $P_{variable\_definition}$, $P_{condition}$, $P_{obligation\_expression}$, $P_{advice\_expression}$, $P_{created\_datetime}$, $P_{updated\_datetime}$}

Table 3.5 describe the definition of all notations of the policy data table.

Table 3.5: Notation and definition of the policy data table

Notation	Definition
P	Policy. A collection of rules, a unique identity for the rule-combining method, and (optionally) a set of obligations or advice. It may be a part of a policy set. An XACML policy consists of header information (version, schema location, defaults values), an additional text description of the policy, a target, and an optional set of obligation expressions.

Table 3.5, continued

P_{id}	This is a unique id for every policy. Identify the policy and can edit it easily by this unique id.
$P_{version}$	Every policy has a version name if the policy updates, the policy version name is changed.
$P_{description}$	Short description of a policy that helps readers to understand the policy. It is optional.
$P_{defaults}$	In an organization, there are many policies. But in a specific period, one policy is active. This field does this task.
P_{target_id}	Policy can contain one target. This is also optional. If a policy has a target, then it creates a relationship with that target by taking the target id.
$P_{rule\_combining\_algid}$	In XACML, one policy consists of one or more than one rule. There have some standard rules combining algorithms. By selecting one rule combination algorithm, the policy can decide which rule will be implemented. This policy table links with the rule-combining-algorithm table by this field.
P_{issuer_id}	This field is optional. To keep the record of the issuer, use this field.
$P_{variable\_definition}$	The FunctionId property specifies a function that is used to define policy variables. It takes arguments so that it can take the decision.
$P_{condition}$	This is also optional. If one policy depends on another policy or rules, it would be used. Always the condition must be true to implement the policy.
$P_{obligation\_expression}$	The policy's issuer may include obligation expressions. This is optional. If a policy set requires some obligation, then it will be added. Otherwise, it will be empty. If obligation is added, then the policy set evaluates that obligation and returns it within the response.
$P_{advice\_expression}$	This is also optional. The policy's issuer may include Advice expressions. If a policy set requires some advice, then it will be added. Otherwise, it will be null. If advice express is added, then the policy set evaluates that advice and returns it within the response as advice.
$P_{created\_datetime}$, $P_{updated\_datetime}$	Policy created time, and if the policy is edited, it takes those records.

*(b) **The policy set data table structure:***

$PS = \{PS_{id}, PS_{pid}, PS_{description}, PS_{defaults}, PS_{target\_id}, PS_{policy\_combining\_algid}, PS_{issuer\_id}, PS_{obligation\_expression}, PS_{advice\_expression}, PS_{created\_datetime}, PS_{updated\_datetime}\}$

Table 3.6 describe the definition of all notations of the policy set data table structure.

Table 3.6: Notation and definition of the policy set data table structure

Notation	**Definition**
PS_{id}	This is a unique number for every policy set.
PS_{pid}	One policy set consists of more than one policy. In this field, we store the policy id.
$PS_{defaults}$, PS_{target_id}, $PS_{policy\_combining\_algid}$, PS_{issuer_id}, $PS_{obligation\_expression}$, $PS_{advice\_expression}$, $PS_{created\_datetime}$, $PS_{updated\_datetime}$	All these fields are the same as the policy data table.

*(c) **The rules data table structure:***

$R = \{R_{id}, R_{description}, R_{effect}, R_{target\_id}, R_{variable\_definition}, R_{condition}, R_{created\_datetime}, R_{updated\_datetime}\}$

Table 3.7 describe the definition of all notations of the rules data table structure.

Table 3.7: Notation and definition of the rules data table structure

Notation	**Definition**
R	Rule. Rule is a main component of a policy. It may be a goal, an outcome, a condition, and (optionally) a set of obligations or advice.
R_{id}	This is a unique id for every rule. Identify the rule and can edit it easily by this unique id.
$R_{description}$	It stores the description of the rule. It is optional. But it helps to understand the rule clearly.
R_{effect}	This field is used to keep track of what will happen if the rules are true. The impact of the Rules will be either "Permit" or "Deny." It will evaluate to "Permit" if the rules evaluation result is fulfilled. It means that the desired access should be granted as long as this one rule is followed. Otherwise, the Deny is saved. When a rule evaluates to "False," the value "NotApplicable" is returned. If an error occurs during the evaluation of the rule, the outcome is "Indeterminate." The policy's rule-combining algorithm defines how different rule values are merged into a one policy value.
R_{target_id}	It keeps the target id of this rule. It creates a relationship with the target table.
$R_{variable\_definition}$, $R_{condition}$, $R_{created\_datetime}$, $R_{updated\_datetime}$	These all are the same as policy table data.

*(d) **The target and target relational data table's structure:***

$T = \{T_{id}, T_{created\_datetime}, T_{updated\_datetime}\}$

$TAO = \{TAO_{id}, TAO_{target\_id}, TAO_{ao\_id}, TAO_{created\_datetime}, TAO_{updated\_datetime}\}$

$AOAO = \{AOAO_{id}, AOAO_{ao\_id}, AOAO_{allo\_id}, AOAO_{created\_datetime}, AOAO_{updated\_datetime}\}$

$AO = \{AO_{id}, AO_{ao\_id}, AO_{created\_datetime}, AO_{updated\_datetime}\}$

$AOM = \{AOM_{id}, AOM_{ao\_id}, AOM_{match\_id}, AOM_{created\_datetime}, AOM_{updated\_datetime}\}$

$M = \{M_{matchId}, M_{av\_data\_type}, M_{av\_data}, M_{ad\_mustbepresent}, M_{ad\_category}, M_{ad\_data\_type}, M_{created\_datetime}, M_{updated\_datetime}\}$

$E = \{E_{id}, E_{environment\_name}, E_{environment\_value}, E_{environment\_data\_type}, E_{created\_datetime}, E_{updated\_datetime}\}$

$A = \{A_{id}, A_{action\_name}, A_{action\_value}, A_{action\_data\_type}, A_{created\_datetime}, A_{updated\_datetime}\}$

$R = \{R_{id}, R_{resource\_name}, R_{resource\_value}, R_{resource\_data\_type}, R_{created\_datetime}, R_{updated\_datetime}\}$

$S = \{S_{id}, S_{subject\_name}, S_{subject\_value}, S_{subject\_data\_type}, S_{created\_datetime}, S_{updated\_datetime}\}$

Table 3.8 describe the definition of all notations of the target and target relational data table's structure.

Table 3.8: Notation and definition of the target and target relational data table's structure

Notation	**Definition**
T	Target
TAO	TargetAnyOf
AOAO	AnyOfAllOf
AO	AnyOf
AOM	AllOfMatch
M	Match
E	Environment
A	Action
R	Resource
S	Subject Attributes

(e) ***The subject attributes data table structure:***

Table 3.9 describe the definition of all notations of the subject attributes data table structure.

Table 3.9: Notation and definition of the subject attributes data table structure

Notation	**Definition**
S_{id}	This is the identity key field of the data table.
$S_{subject\_name}$	This is the attribute name of the subject. It can be named, age, date of birth, etc.
$S_{subject\_value}$	This field keeps the value of the attribute. Like client name Mushfiqur Rahim, age is 23 years, date of birth is 25th August 1998.
$S_{subject\_data\_type}$	It keeps the data type of the attribute. Like as string, URL, email, domain name, etc.
$S_{created\_datetime}$, $S_{updated\_datetime}$	Like the other data table, these keep the record's creation date and updated date to easily manipulate the subject attributes.

(f) ***The resource attributes data table structure:***

Table 3.10 describe the definition of all notations of the resource attributes data table structure.

Table 3.10: Notation and definition of the resource attributes data table structure

Notation	**Definition**
R_{id}	Keep the unique id for every individual resource.
$R_{resource\_name}$	Name of the resource. Like as personal information, medical records, specific medical record.
$R_{resource\_value}$	It keeps the value of the resource. Like blood sugar, blood pressure, medical information.
$R_{resource\_data\_type}$, $R_{created\_datetime}$, $R_{updated\_datetime}$	Same as the subject attribute data tables.

(g) ***The action attributes data table structure:***

Table 3.11 describe the definition of all notations of the action attributes data table structure.

Table 3.11: Notation and definition of the action attributes data table structure

Notation	**Definition**
A_{id}	This field keeps the identity number of any action. It helps to manipulate the action data any time quickly.
A_{action_name}	Name of the action.
A_{action_value}	It keeps the value of the action.
$A_{action\_data\_type}$, $A_{created\_datetime}$, $A_{updated\_datetime}$	Same as the subject attribute data tables.

(h) ***The environment attributes data table structure:***

Table 3.12 describe the definition of all notations of the environment attributes data table structure.

Table 3.12: Notation and definition of the environment attributes data table structure

Notation	**Definition**
E_{id}	This field keeps the identity number of any environment. It helps to manipulate the environment data any time quickly.
$E_{environemt\_name}$	Name of the environment.

Table 3.12, continued

$E_{environment\_value}$	It keeps the value of the environment.
$E_{environment\_data\_type}$, $A_{created\_datetime}$, $A_{updated\_datetime}$	Same as the subject attribute data tables.

(i) The target data table structure:

Table 3.13 describe the definition of all notations of the target data table structure.

Table 3.13: Notation and definition of the target data table structure

Notation	**Definition**
T_{id}	T_{id} stores the unique identity of a target.
$T_{created\_datetime}$, $T_{updated\_datetime}$	This field store the created, and updated date.

Target data table is linked with some other tables TAO, AOAO, and AO. Target is linked with the subject, action, resource, and environment by Match table

(j) The match data table structure:

Table 3.14 describe the definition of all notations of the match data table structure.

Table 3.14: Notation and definition of the match data table structure

Notation	**Definition**
$M_{matchautoID}$	It stores the unique identity of a match data table.
$M_{matchId}$	Match Id field contains the function which will be used in performing the match evaluation.
$M_{av\_data\_type}$, M_{av_data}	These stores the matching attribute value data type and value. Like as data type strings and data will be the name of a patient.
$M_{ad\_mustbepresent}$, $M_{ad\_category}$, $M_{ad\_data\_type}$	Attribute designator will be link with the subject/action/resource/actions. It keeps those data.
$M_{created\_datetime}$, $M_{updated\_datetime}$	Same as the subject attribute data tables.

(k) The rule-combining algorithm data table structure:

rule-combining algorithm = {RCA_{id}, Alg_{desc}}

Table 3.15 describe the definition of all notations of the rule-combining algorithm data table structure.

Table 3.15: Notation and definition of the rule-combining algorithm data table structure

Notation	**Definition**
rule-combining algorithm	This algorithm describes how to determine permission based on the individual results of a set of rules.
RCA_{id}	This is the primary/unique key of this table.
Alg_{desc}	This field keeps the main algorithm.

Description of some standard combining algorithms are given in the following:

Deny-overrides (ordered and unordered):

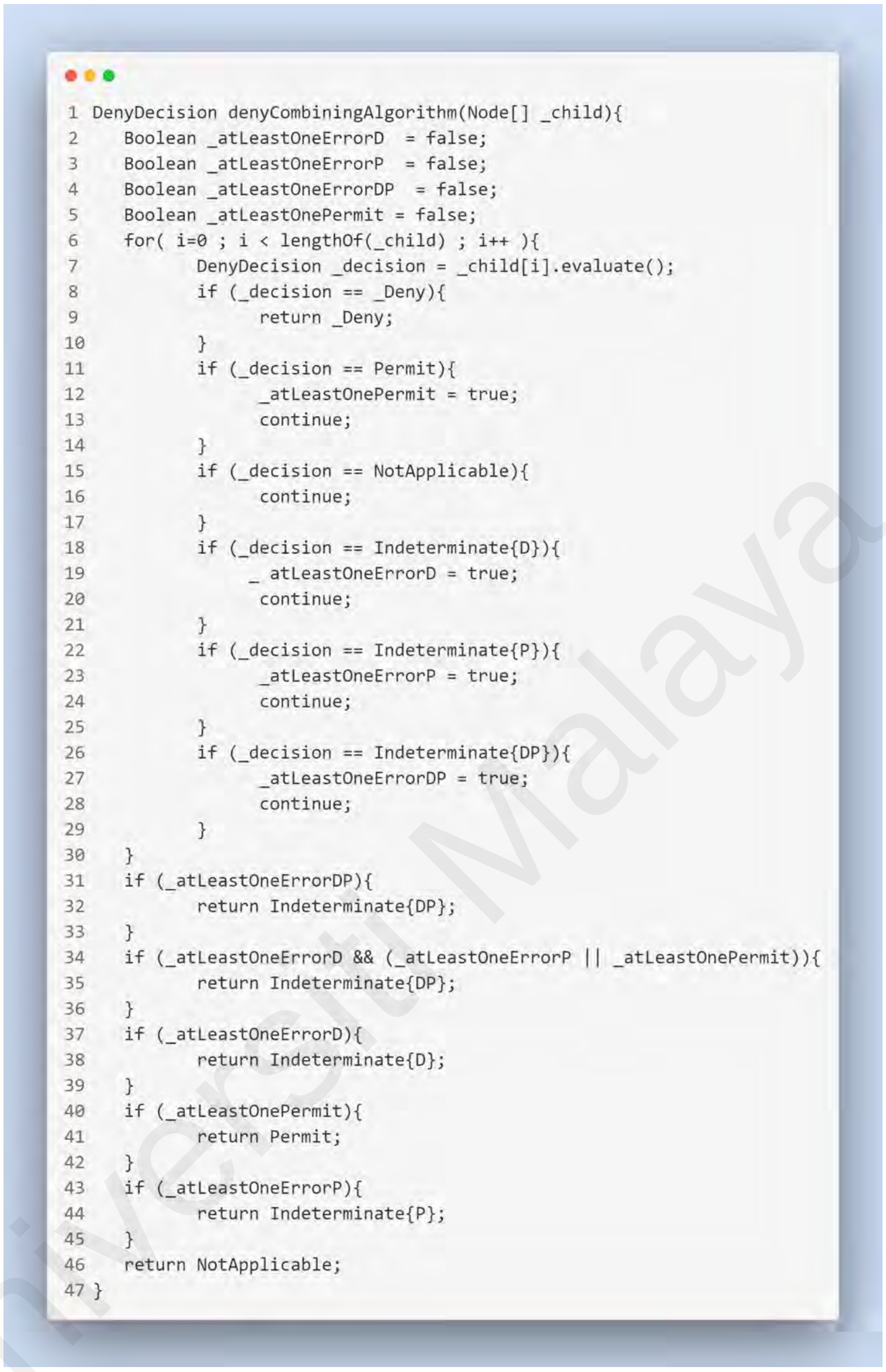
The deny overrides combining method is for situations when a denial decision should take precedence over a permit decision. Table 3.16 describe the algorithm's behavior in the following paragraphs.

Table 3.16: Deny-overrides algorithms' behavior

The outcome is "Deny" if any choice is "Deny."
Otherwise, the outcome is "IndeterminateDP" if any choice is "IndeterminateDP."
Otherwise, the outcome is "IndeterminateDP" if any decision is "IndeterminateD" and another decision is "IndeterminateP" or Permit.
Otherwise, the outcome is "IndeterminateD" if any choice is "IndeterminateD."
If any choice is "Permit," the outcome is also "Permit."
Otherwise, the outcome is "IndeterminateP" if any choice is "IndeterminateP."
Otherwise, "Not Applicable" is returned.

This combining technique is utilized in most authorization systems. This algorithm is used in our testbed.

The normative specification of this combining method is represented by the pseudo-code at Figure 3.10. The method is given here in the form of an array including the policy or policy set's offspring (policies, policy sets, or rules). Because the children may be handled in any sequence, the set of responsibilities or recommendations provided by this method is not predictable.



```
DenyDecision denyCombiningAlgorithm(Node[] _child){
    Boolean _atLeastOneErrorD  = false;
    Boolean _atLeastOneErrorP  = false;
    Boolean _atLeastOneErrorDP  = false;
    Boolean _atLeastOnePermit = false;
    for( i=0 ; i < lengthOf(_child) ; i++ ){
            DenyDecision _decision = _child[i].evaluate();
            if (_decision == _Deny){
                    return _Deny;
            }
            if (_decision == Permit){
                    _atLeastOnePermit = true;
                    continue;
            }
            if (_decision == NotApplicable){
                    continue;
            }
            if (_decision == Indeterminate{D}){
                   _ atLeastOneErrorD = true;
                    continue;
            }
            if (_decision == Indeterminate{P}){
                    _atLeastOneErrorP = true;
                    continue;
            }
            if (_decision == Indeterminate{DP}){
                    _atLeastOneErrorDP = true;
                    continue;
            }
    }
    if (_atLeastOneErrorDP){
            return Indeterminate{DP};
    }
    if (_atLeastOneErrorD && (_atLeastOneErrorP || _atLeastOnePermit)){
            return Indeterminate{DP};
    }
    if (_atLeastOneErrorD){
            return Indeterminate{D};
    }
    if (_atLeastOnePermit){
            return Permit;
    }
    if (_atLeastOneErrorP){
            return Indeterminate{P};
    }
    return NotApplicable;
}
```

Figure 3.10: Pseudo-code of deny-overrides algorithms

Permit-overrides (ordered and unordered):

The permit overrides combining method is for situations when a permit decision should take precedence over a denial decision. Table 3.17 describe the behaviour of this algorithm.

Table 3.17: Permit-overrides algorithms' behavior

The outcome is "Permit" if any choice is "Permit."
Otherwise, the outcome is "IndeterminateDP" if any choice is "IndeterminateDP."
Otherwise, the outcome is "IndeterminateDP" if any choice is "IndeterminateP" and another decision is "IndeterminateD" or "Deny."
Otherwise, the outcome is "IndeterminateP" if any choice is "IndeterminateP"
If any choice is "Deny," the outcome is also "Deny."
Otherwise, the outcome is "IndeterminateD" if any choice is "IndeterminateD"
Otherwise, "Not Applicable" is returned.

First-applicable:

A non-normative instructive explanation of a policy's "First-applicable" rule-combining method follows. Each rule in the policy MUST be assessed in the order in which it is stated. If the target matches and the condition passes to "True" for a specific rule, the policy evaluation SHALL come to a stop, and the rule's associated effect will be the outcome of the policy evaluation (i.e., "Permit" or "Deny"). If the target passes to "False" or the condition evaluates to "False" for a specific rule in the evaluation, the following rule in the order WILL be examined. If there are no more rules in demand, the policy will be considered "Not Applicable". If an error occurs when evaluating a rule's goal or condition, the evaluation will come to a stop, and the policy will be evaluated as "Indeterminate" with the appropriate error status.

Only-one-applicable:

The following explanation is a non-normative and instructive explanation of the "Only-one-applicable" condition A policy set's policy combining algorithm.

If no policy is deemed applicable by virtue of its goal throughout the whole policy collection, the outcome of the policy-combination algorithm WILL be "Not Applicable". The outcome of the policy-combination algorithm WILL be "Indeterminate" if more than one policy is deemed relevant by virtue of its goal.

If only one policy is deemed relevant by its target's assessment, then the outcome of the policy-combining procedure MUST equal the result of the policy evaluation.

If an error occurs when evaluating a policy's target, or if a policy's reference is deemed incorrect, or if the policy evaluation returns “Indeterminate”, the policy set WILL passes to “Indeterminate." ", with the appropriate status for the mistake.

In this system, the deny-overrides algorithm has been used.

3.4 Simulation of Designed T-TAFAC

In this section, describe the simulation of the designed ACA. T-TAFAC is developed in a prototype testbed (Figure 3.11). In the following, describe the different tiers of the designed T-TAFAC.

3.4.1 Fog Node

In the designed T-TAFAC, the middle and primary tier is the fog tier. Most of the functionality is controlled in this tier. This tier is known as Central Management System (CMS) that used Lenovo Y700 Laptop, which necessary configurations are given in given in the following:

Brand and Model: Lenovo IdeaPad Y700.

Operating System: Windows 10 Home.

Display: 17.30-inch display with a resolution of 1920x1080 pixels.

Processor: Core i7 processor

Memory: 16GB of RAM.

Storage: 1TB of HDD storage and 128GB of SSD storage.

Graphics: AMD Radeon R16M M1 30.

Connectivity: Wi-Fi 802.11 ac, Ethernet 10/100/1000 Base T, and RJ45 (LAN) ports.

All the programs for CMS are developed in C#. C# is structured, imperative, object-oriented, event-driven, task-driven, functional, generic, reflective, concurrent programming language design and developed by Microsoft. Developed three different applications for this T-TAFAC. One is web application for doctors, patients, and healthcare personnel. The second one is the console application for service used for the patient's data insert. The frontend application gets the data from the backend application by Web API. Used SQLite databases: open source, small, fast, self-contained, high-reliability, and full-featured (SQLite, 2000) for this T-TAFAC. Two databases are designed in this testbed. Authorization database for the authorization and HMS database for the Healthcare information. For testbed, all databases are in a single server. But in the real-time authorization database by using XACML will be in the cloud.

3.4.2 Device / Edge Node

To implement the IoT platform, used a NodeMCU V3 (NodeMCU, 2013) as a Things for this T-TAFAC, which is open-source, interactive, programmable, low cost, simple, smart Wi-Fi (ESP8266-12E microcontroller) enabled. The ESP8266 is a cheap Wi-Fi microchip with a full TCP/IP stack and microcontroller capability, Tensilica Xtensa 32-bit LX106 RISC microprocessor. This microprocessor supports RTOS and operates at 80MHz to 160 MHz adjustable clock frequency. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. All things/sensors are connected by NodeMCU. Developed the program for IoT by Lua Language. Lua is a powerful, efficient, lightweight, embeddable scripting language (Lua, 2021). All communications are performing over HTTPS. SHA256 is used for the hash and 128-bit key size for AES encryption.

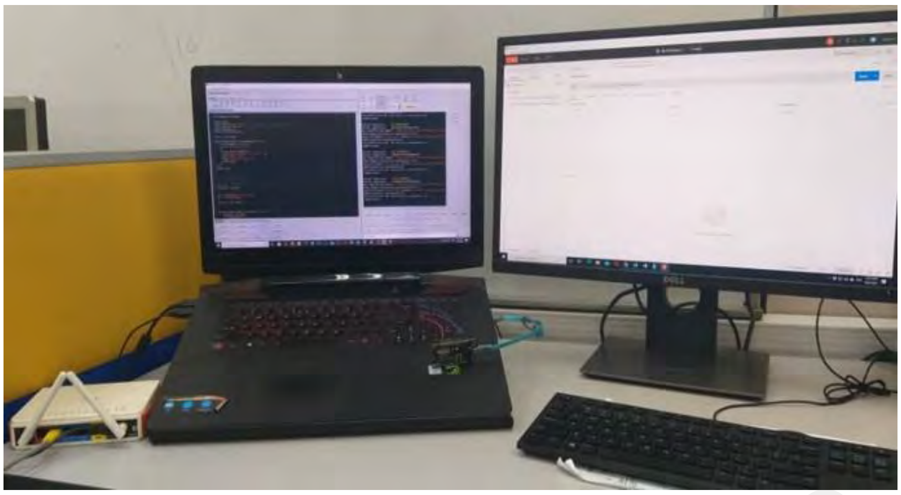


Figure 3.11: Simulation set-up environment

Dell OptiPlex 3050 Desktop (Figure 1.1) used as a User Device which configuration is Intel Core i7–6700 CPU@3.40GHz and 3.41GHz processor, and operating system Windows 10 Home with 8GB RAM. It is connected to the router by LAN.

Google Chrome is the most famous cross-platform web browser developed by Google. *Chrome* DevTools is a set of web *developer tools* built directly into the Google *Chrome* browser. In this research, these tools are used for processing and communication time measurement.

The User Device (UD) is communicating with CMS by the web application and web API. And Things/NodeMCU is communicating with CMS by web service.

3.5 Chapter Summary

This chapter has presented the research methodology used in this research which consists of three different phases. In phase one, the background and challenges of an access control architecture are studied and discussed. In medical, due to the human life involvement, data authorization with processing time is very crucial. Many researchers have proposed different types of an access control architecture for IoMT. The different components of an access control architecture are studied, and the literature review on existing access control architecture is assessed thoroughly. Next, the advantages and drawbacks of existing access control architecture are evaluated.

In phase two, the design and development of the designed architecture is simulated, which is developed using C#, Laptop, Desktop and NodeMCU as development tools.

Lastly, in phase three, generated and analyzed the results. Experiments are carried out by comparing our designed T-TAFAC with benchmark architecture.

CHAPTER 4: RESULTS AND PERFORMANCE ANALYSIS

4.1 Introduction

This chapter discusses and evaluates the results and performance analysis of the designed T-TAFAC. First, is the processing time performance of this architecture that reduces the processing time of the user to access the medical records; second, is the performance in the management of the policy and third is the comparison with existing benchmark papers.

4.2 Scenario: Data Access from Desktop or Mobile

There may be two scenarios for data access based on the device used from which the request is sent. The first scenario is shown in Figure 4.1. In this case, the user can be a doctor/patient/guardian who wants to access patients' data from the desktop inside the hospitals. That means they are inside the hospital.

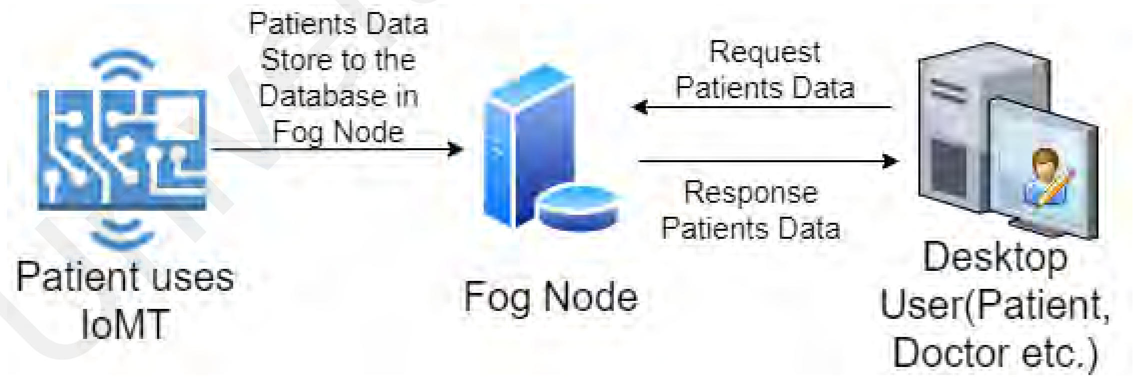

Patients Data Store to the Database in Fog Node
Patient uses IoMT
Fog Node
Request Patients Data
Response Patients Data
Desktop User(Patient, Doctor etc.)

Figure 4.1: Scenario one: Patient data insert to the database in fog node and the Desktop user request patient date from the fog node.

The second scenario is shown in Figure 4.2. Here, the user can be a doctor/patient/guardian who wants to access patients' data from the mobile inside the hospital. That means they are inside the hospital. They are using the local network.

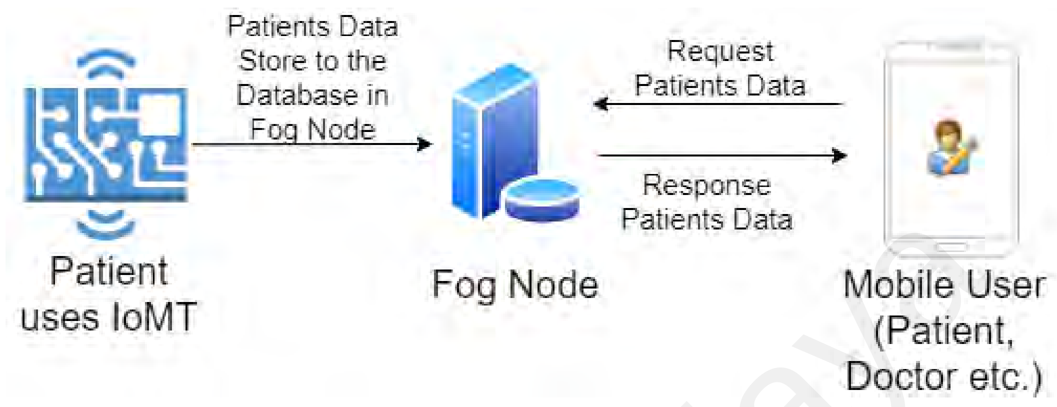

Patients Data Store to the Database in Fog Node
Request Patients Data
Response Patients Data
Patient uses IoMT
Fog Node
Mobile User (Patient, Doctor etc.)

Figure 4.2: Scenario two: Patient data insert to the database in fog node and the Mobile user request patient date from the fog node.

4.3 Performance Discussion of T-TAFAC

In T-TAFAC, there are four performance metrics. In the following, describe those metrics:

4.3.1 Data Request and Response Time

The processing time has been measured from when a user wants (i.e., user send request) to access patient data to get it from the CMS. The test has been repeated 100 times to ensure the accuracy of data access.

For the previous scenarios, two statuses are considered: capable users and another is not capable users. One has authorized permission to access the patient data. Other users do not have the authority to access that information. Repeated the task 100 times for both authorized and unauthorized users.

In Figure 4.3, If the desktop user has the capability, it takes 457ms. On the other side, incapable users get the response within 402ms. Incapable users' responses take less time because getting patient information is not required in that scenario. In that time, the system does not need to process the patient's information. Authorized and unauthorized user send the user id, password, the patients id and the resource id in the request as attributes. In the testbed, patient's temperature information only requested.

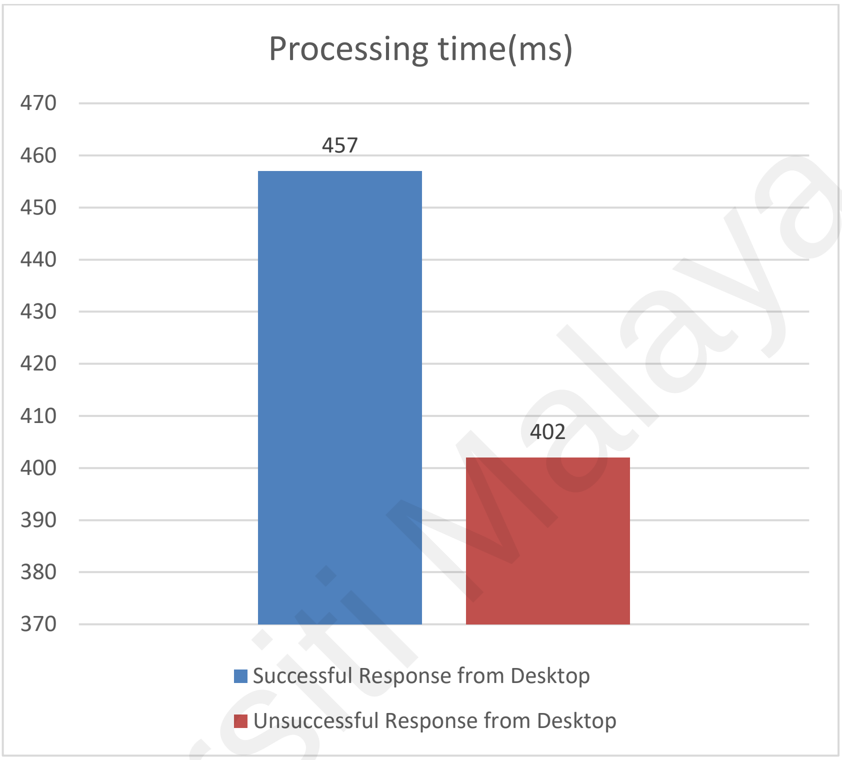


Figure 4.3: Scenario one: response time for desktop users

In Figure 4.4, In the second scenario, mobile users same as the previous one; there are two statues. One is for authorized users, and another one is for unauthorized users. The authorized mobile user receives the response with the patient's information within 667ms. But unauthorized mobile user receives the response without the patient's information within 641ms.

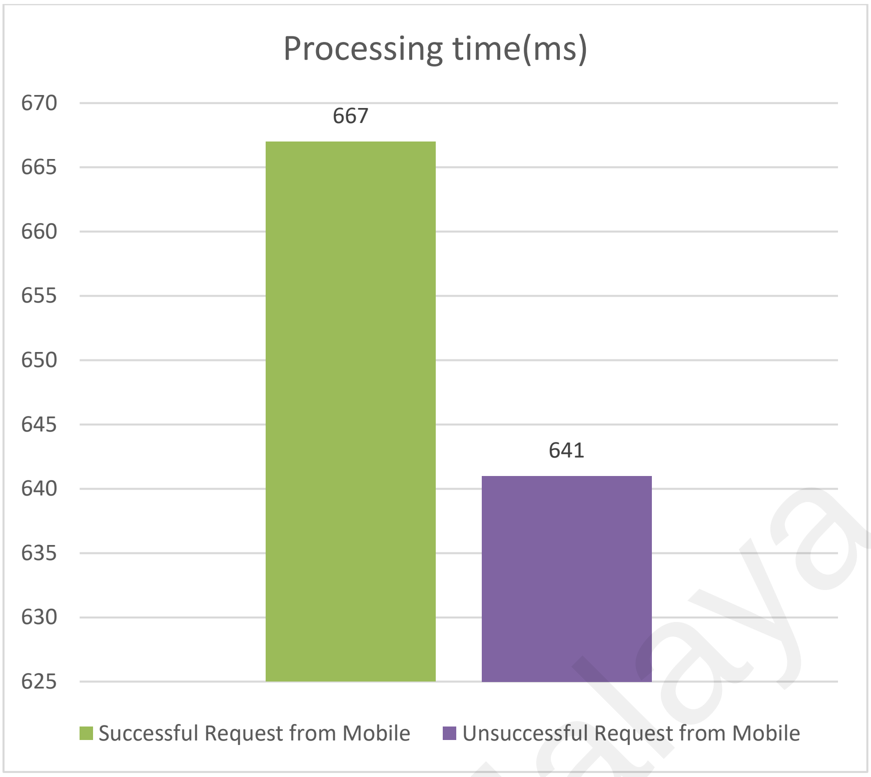


Figure 4.4: Scenario two: response time for mobile users

Data has been taken 100 times for each scenario. For the testbed, 20000 sample data including active users, things, things data and capabilities have been inserted in each data table. After getting the records of all processing time, the average processing time is 541ms.

4.3.2 Data Insertion Time

In Figure 4.5, sensors gather data from the patient's and send those data to the CMS. It takes 100ms. T-TAFAC gathers the information every 100ms. IoT Sensors gather the data and encrypted inside the IoT devices and transfer those data to CMS. In the testbed, patients' temperature data gather from the patients. Hospital Management System is inside the CMS. The HMS decrypts those data and inserts those into the database.

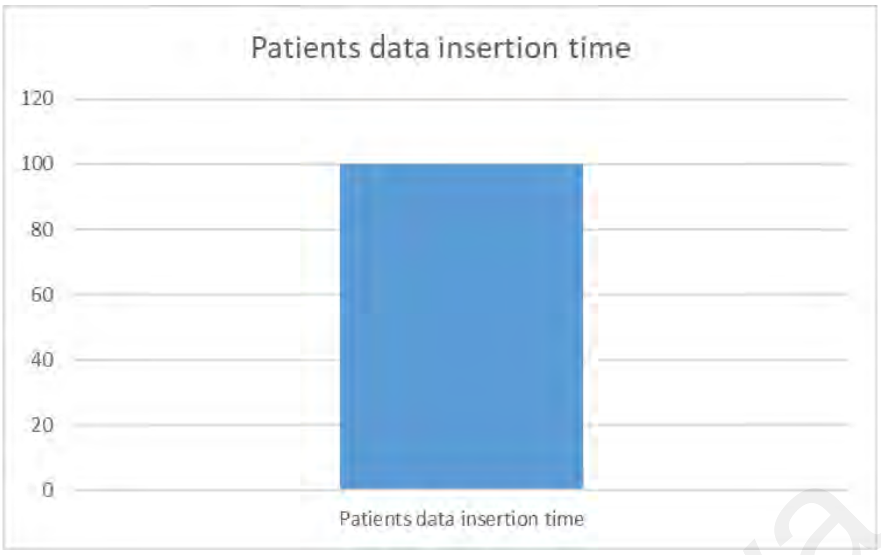


Figure 4.5: Patient's data insertion time.

4.3.3 Capability Generation Time

In this step, the task is to generate the capability generator as shown in Figure 4.6. In T-TAFAC, the capability is generating by using a query in the cloud tier. This research considers only the query time, but the synchronization time between cloud and fog tier is out of the scope of this research. To generate capability takes 5ms. This 5ms is only for a specific user all capability.

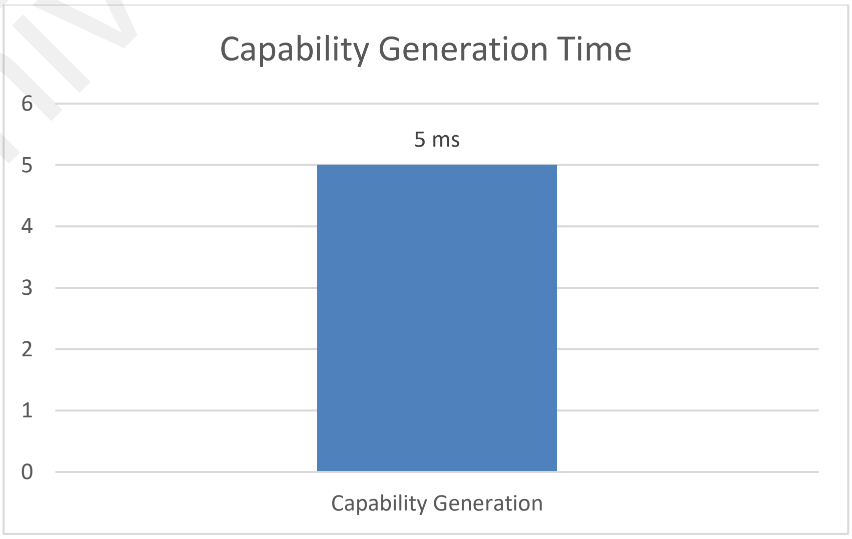


Figure 4.6: Average capability generation time. Each user's capability is generated by 5ms.

4.3.4 Total Processing Time

Figure 4.7 shows the total processing time of the testbed. The average processing time taken is 646ms by using the following calculation.

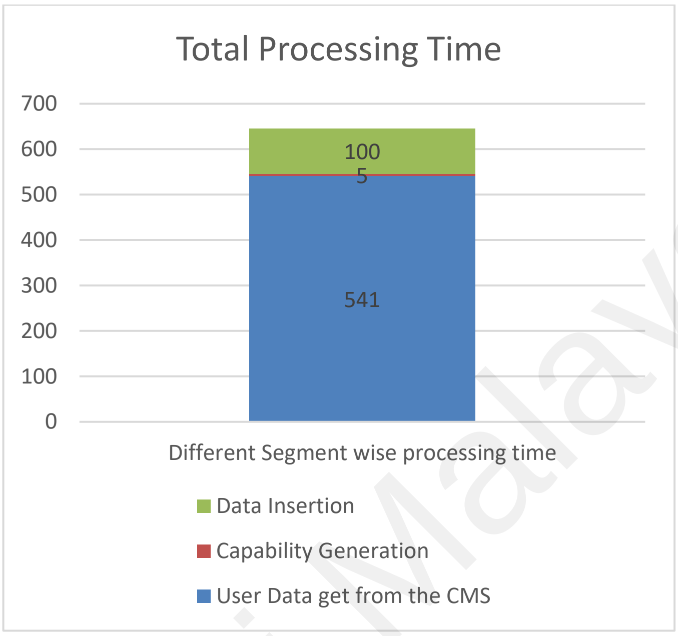


Figure 4.7: Total data processing time

$TDP_{time} = RR_{time} + DI_{time} + CG_{time}$

where

TDP_{time} is Total data processing time.

RR_{time} is Request & Response time

DI_{time} is Data Insertion time.

CG_{time} is Capability Generation time

In the present study, the total data processing time is calculated as:

$TDP_{time} = 554\text{ms} + 100\text{ms} + 5\text{ms} = 646\text{ms}$

In Figure 4.8 shows the performance comparison of T-TAFAC with existing two ACA. Pals et al. (2019), ACA took 706ms, but T-TAFAC takes 646ms. Because that ACA did the authenticate process inside the IoT devices, on the other hand T-TAFAC is done authentication in the fog tier. That's why this architecture take less time than Pal et al., (2019) ACA.

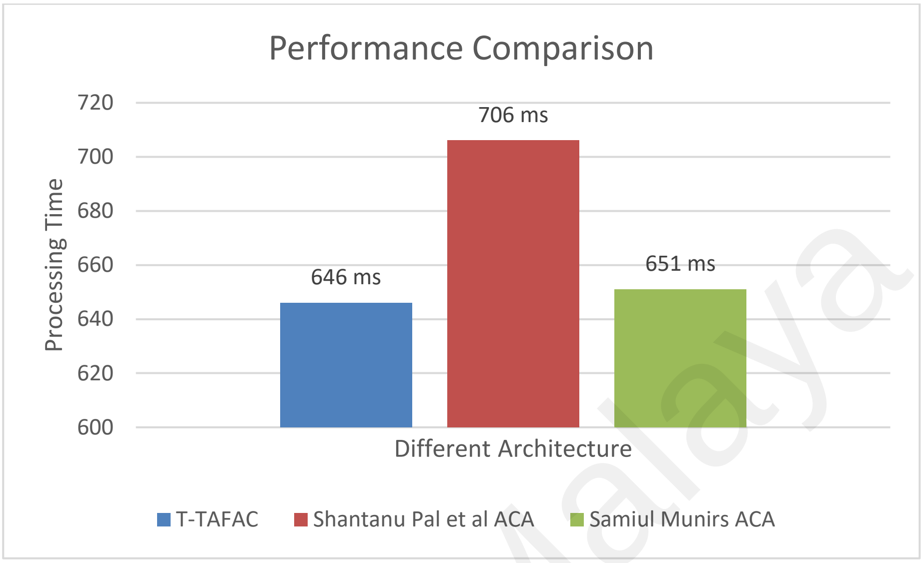


Figure 4.8: Performance comparison of T-TAFAC with Pal et al., (2019) ACA and Monir (2017) ACA

This system also gives good results rather than Monir (2017) ACA because that system did the authorization process at the cloud tier. So, every time to communicate with the cloud tier take more time rather than the fog tier.

4.4 Performance Discussion on Access Control Model

The T-TAFAC considered the three most common uses of ACM. By using a combination of these models, T-TAFAC reduces the complex management deficiency of significant policy in ABAC. T-TAFAC treat role attribute as a Role Membership. It reduces the number of policies, and it works like as RBAC. By implementing ABAC, it becomes fine-grained. T-TAFAC generate capability and store those capabilities in the fog tier. So that T-TAFAC gets the pros of CBAC that is time efficiency.

4.5 Chapter Summary

In this research, designed three-tiered architecture but compare with two-tiered architecture. Because this research main objective is to reduce the authorization processing time in the healthcare data process and the second objective is to reduce the number of policies for better management. Alnefaie et al., (2019) had proposed three-tiered architecture for healthcare without implementation and use only one attribute-based access control model. For that reason, The T-TAFAC compare with Pal et al., (2019) ACA and Monir (2017) ACA both are for the healthcare industry.

CHAPTER 5: CONCLUSIONS

In this concluding chapter, the achievements of the research, findings and contributions of the research are delineated along with recommendations for future works.

5.1 Achievements

The three-tiered architecture for access control is developed for better performance in the authorization system to consider the processing time. In this system, introduce fog computing in the middleware, which stores the capability database. That reduces the total processing time in the IoMT networks.

In chapter one, initially is discussed the research background. The research background describes the IoT, cloud computing, fog computing, access control architecture (ACA), and access control model (ACM).

In chapter two, reviewed the existing access control architecture; later describe the main components of the access control architecture. Summarize the reviewed ACA and finally explain the challenges in ACA.

In chapter three, the fog-based access control architecture is designed. In this architecture, firstly fog tier stores the capability generated by the cloud tier. Secondly, the capability is developed by considering the XACML, which is an Attribute-based access control model. The primary data communication is between the user with the fog tier. The IoMT devices store data in the fog tier.

In chapter four, the designed T-TAFAC is developed in a testbed using C# technology for programming and NodeMCU for IoT devices (NodeMCU, 2013). Get the result from the testbed, and the result then compares with Pal et al., (2019) ACA and Monir (2017)

ACA. Based on the comparison, T-TAFAC gives better performance in considering the processing time.

5.2 Findings and Contributions

The main contribution of this research is to develop a T-TAFAC that should improve authorization processing time and reduce the number of policies by using fog computing and combining ABAC, RBAC, and CBAC. For this purpose, the research contributions are summarized as follows:

- Develop a three-tiered architecture for access control.
- Emulating the T-TAFAC in the C# programming language.
- Evaluate and analyse the T-TAFAC, compare its performance with existing ACA, such as Pal et al. (2019), ACA and Monir (2017) ACA.

5.3 Conclusion

ACA is an essential element in the security perspective to protect the data from unauthorised access. In the same way, a doctor requires to access the medical data of a patient at a minimal time, is also very crucial. So, it is very challenging to design an ACA by considering the processing time. This research designs a T-TAFAC with a testbed, based on three categories of access control models by introducing fog computing. The testbed result shows that the processing and communication time in IoMT networks is minimal compared to other existing architecture (Monir, 2016; Pal et al., 2019).

5.4 Future Work

In this research, the fog-based access control architecture is designed and implemented, which improved the authorization processing time and reduced the number of policies by introducing fog computing and combining the three most common ACM. The T-TAFAC also is used the database based on XACML. T-TAFAC can be improved further by fully implementing the database based on XACML in the cloud. Apart from that, it can be enhanced by the device-to-device communication. Moreover, the authorization process will be done in the fog tier and the data communication will be device to device.

REFERENCES

Abu Bakar, M. T., and Jamal, A. A. (2020). Latency issues in internet of things: A review of literature and solution. *International Journal of Advanced Trends in Computer Science and Engineering*, *9*(1.3): 83–91.

Aftab, M. U., Habib, M. A., Mehmood, N., Aslam, M., and Irfan, M. (2016). *Attributed role based access control model*. Proceedings of Conference on Information Assurance and Cyber Security 2015 (CIACS 2015), December 18, 2015, Rawalpindi, Pakistan, 83–89.

Alnefaie, S., Cherif, A., and Alshehri, S. (2019). *Towards a Distributed Access Control Model for IoT in Healthcare*. 2nd International Conference on Computer Applications and Information Security (ICCAIS), May 1-3, 2019, Riyadh, KSA.

Anawar, M. R., Wang, S., Azam Zia, M., Jadoon, A. K., Akram, U., and Raza, S. (2018). Fog computing: An overview of big IoT data analytics. *Wireless Communications and Mobile Computing*, 2018: 1-22.

ANSI. (2016). *INCITS: INCITS 459-2011[R2016]: Information Technology - Requirements for the Implementation and Interoperability of Role Based Access Control*. https://standards.incits.org/apps/group_public/project/details.php?project_id=1072

Arutyunov, V. V. (2012). Cloud computing: Its history of development, modern state, and future considerations. *Scientific and Technical Information Processing 2012*, *39*(3): 173–178.

Badidi, E., and Moumane, K. (2019). *Enhancing the processing of healthcare data streams using fog computing*. Proceedings of IEEE Symposium on Computers and Communications, June 29 - July 3, 2019, Barcelona, Spain.

Bhaskaran, S. M., and Sridhar, R. (2017). Hybrid solution for privacy-preserving access control for healthcare data. *Advances in Electrical and Computer Engineering*, *17*(2): 31–38.

Bonomi, F., Milito, R., Natarajan, P., and Zhu, J. (2014). Fog Computing: A Platform for Internet of Things and Analytics. *Studies in Computational Intelligence*, *546*: 169–186.

Chauhan, P., and Sood, M. (2021). Big data: Present and future. *Computer*, *IEEE Access, 54*(04): 59–65.

Chen, Y., Xu, R., Blasch, E., and Chen, G. (2018). A federated capability-based access control mechanism for Internet of Things (IoTs). In K. D. Pham & G. Chen (Eds.), *Sensors and Systems for Space Applications XI* (Vol. 10641, p. 29). SPIE.

Dennis, J. B., and Van Horn, E. C. (1966). Programming semantics for multiprogrammed computations. *Communications of the ACM*, *9*(3): 143–155.

Dhivya, A. D., Ramya. K, and Kanimozhi. N. (2017). Fog Computing-An Overview. *International Journal of Trend in Research and Development*, *4*(1): 2394–9333.

Dilawar, N., Rizwan, M., Ahmad, F., and Akram, S. (2019). Blockchain: Securing internet of medical things (IoMT). *International Journal of Advanced Computer Science and Applications*, *10*(1): 82–89.

Dirk, B., Glenn, D., and D. K., S. (2005). *Sixteen. Making the Impossible Easy: Usable PKI - Security and Usability [Book]*. O'Reilly Media, Inc. https://www.oreilly.com/library/view/security-and-usability/0596008279/ch16.html

Ellison, C., Intel, B. Frantz, Electric Communities, B. Lampson, Microsoft, R. Rivest, M. L. for C. S., Thomas, B., and Southwestern Bell, T. Ylonen, S. (1999). *SPKI Certificate Theory, IETF RFC 2693*. The Internet Society (1999). https://www.ietf.org/rfc/rfc2693.txt

Erik, R. (2013). *eXtensible Access Control Markup Language (XACML) Version 3.0*. OASIS Open 2013. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

Fang, L., Dennis, G., and Siebenlist, F. (2005). *XPOLA – An Extensible Capability-based Authorization Infrastructure for Grids*. 4th Annual PKI R&D Workshop, (pp 30-40), April 19-21, 2005. Gaithersburg, MD USA

Ferraiolo, D. F., and Kuhn, D. R. (1992). *Role-Based Access Controls*. 15th National Computer Security Conference, (pp. 554-563.). October 13-16, 1992. Baltimore MD, USA.

Henry, M. L. (1984). *Capability-Based Computer Systems[book]*. Digital Press. https://homes.cs.washington.edu/~levy/capabook/

Hu, V. C., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R., and Scarfone, K. (2014). *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*. NIST Special Publication 800-162, USA.

Hussein, D., Bertin, E., and Frey, V. (2017). *Access control in IoT: From requirements to a candidate vision*. Proceedings of 20th Conference on Innovations in Clouds, Internet and Networks (ICIN 2017) (pp. 328–330). March 7-9, 2017, Paris, France. IEEE Xplore. https://doi.org/10.1109/ICIN.2017.7899435

ITSecurityGuru. (2016). *IT Pros Admit Unauthorised Access, Malware and DoS Attacks are Top Three Cloud Security Concerns Today - IT Security Guru*. https://www.itsecurityguru.org/2016/11/16/pros-admit-unauthorised-access-malware-dos-attacks-top-three-cloud-security-concerns-today/

Jerkins, J. A. (2017). *Motivating a market or regulatory solution to IoT insecurity with the Mirai botnet code*. IEEE 7*th* Annual Computing and Communication Workshop and Conference (CCWC 2017), Jan 9-11, 2017 Las Vegas, USA. https://doi.org/10.1109/CCWC.2017.7868464

Karen, T., Mark, S., Amen, S., and Matthew, T. (2018). *Medtech and the Internet of Medical Things How connected medical devices are transforming health care*. https://www2.deloitte.com/content/dam/Deloitte/global/Documents/Life-Sciences-Health-Care/gx-lshc-medtech-iomt-brochure.pdf

Kayla, M. (2019). *Is The Internet of Medical Things (IoMT) on Par With the IoT Market as a Whole?* -. Hitconsultant.Net. https://hitconsultant.net/2019/01/30/is-iomt-tech-iot-market-as-a-whole/#.XzO1ZJMzaDU

Kraemer, F. A., Braten, A. E., Tamkittikhun, N., and Palma, D. (2017). Fog Computing in healthcare: A review and discussion. *IEEE Access*, 5: 9206–9222.

Le-Dang, Q., and Le-Ngoc, T. (2019). *Scalable blockchain-based architecture for massive IoT reconfiguration*. 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE 2019), 5-8 May 2019, Edmonton, Canada.

Liu, H. Y., Deng, M. L., and Yang, W. D. (2012). *A context-aware fine-grained access control model*. Proceedings of 2012 International Conference on Computer Science and Service System (CSSS 2012) (pp. 1099–1102), August 11-13, 2012, Nanjing, China.

Monir, S. (2017). A lightweight attribute-based access control system for IoT (Doctoral Thesis): University of Saskatchewan, Canada.

Mukherjee, M., Matam, R., Shu, L., Maglaras, L., Ferrag, M. A., Choudhury, N., and Kumar, V. (2017). Security and privacy in fog computing: Challenges. *IEEE Access*, *5*: 19293–19304.

Nakamoto, S. (2009). *Bitcoin: A Peer-to-Peer Electronic Cash System*. https://www.researchgate.net/publication/228640975_Bitcoin_A_Peer-to-Peer_Electronic_Cash_System

NodeMcu (2013) *An open-source firmware based on ESP8266 wifi-soc.* (n.d.). Retrieved April 21, 2021, from https://www.nodemcu.com/index_en.html

Oh, S., and Park, S. (2000). Task-role based access control (T-RBAC): An improved access control model for enterprise environment. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *1873*: 264–273.

Ouaddah, A., Mousannif, H., Abou Elkalam, A., and Ait Ouahman, A. (2017). Access control in the Internet of Things: Big challenges and new opportunities. *Computer Networks*, 112: 237–262.

Pal, S., Hitchens, M., Varadharajan, V., and Rabehaja, T. (2018). Fine-grained access control for smart healthcare systems in the Internet of Things. *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, *4*(13): 154370.

Pal, S., Hitchens, M., Varadharajan, V., and Rabehaja, T. (2019). Policy-based access control for constrained healthcare resources in the context of the Internet of Things. *Journal of Network and Computer Applications*, 139: 57-74.

Prakash, P., Darshaun, K. G., Yaazhlene, P., Ganesh, M. V., and Vasudha, B. (2017). Fog computing: Issues, challenges and future directions. *International Journal of Electrical and Computer Engineering*, *7*(6): 3669–3673.

Putra, D. R., Anggorojati, B., and Hartono, A. P. P. (2019). *Blockchain and smart-contract for scalable access control in Internet of Things*. Proceeding of 2019 International Conference on ICT for Smart Society: Innovation and Transformation Toward Smart Region (ICISS 2019) November 19-20, 2019, USA.

Reinsel, D., Gantz, J., and Rydning, J. (2017). *Data Age 2025: The Evolution of Data to Life-Critical Don't Focus on Big Data; Focus on the Data That's Big Sponsored by Seagate The Evolution of Data to Life-Critical Don't Focus on Big Data; Focus on the Data That's Big*. www.idc.com

Sabrina, F. (2019). *Blockchain and structural relationship based access control for IoT: A smart city use case*. Proceedings of 2019 IEEE 44th Conference on Local Computer Networks, (LCN, 2019) (pp. 137–140), October 14-17, 2019 Osnabrueck, Germany.

Salonikias, S., Mavridis, I., and Gritzalis, D. (2016). Access control issues in utilizing fog computing for transport infrastructure. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9578*, 15–26. https://doi.org/10.1007/978-3-319-33331-1_2

Siddiqui, F., Hagan, M., and Sezer, S. (2018). Embedded policing and policy enforcement approach for future secure IoT technologies. *IET Conference Publications* (pp. 10-10), March 28-29, 2018, London, UK.

Sivaselvan, N., Bhat, V. K., and Rajarajan, M. (2020). *Blockchain-based Scheme for authentication and capability-based access control in IoT environment*. 2020 11th IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON 2020) (pp. 0323–0330), October 28-31, 2020, New York, USA.

SQLite (2000). *What is SQLite*? Retrieved on April 21, 2021, from https://www.sqlite.org/index.html

Suhendra, V. (2011). A survey on access control deployment. *Communications in Computer and Information Science*, *259 CCIS*: 11–20, Springer, Berlin, Heidelberg.

Sultan, N. (2010). Cloud computing for education: A new dawn? *International Journal of Information Management*, *30*(2): 109–116.

Surbiryala, J., and Rong, C. (2019). *Cloud computing: History and overview*. Proceedings of 2019 3rd IEEE International Conference on Cloud and Fog Computing Technologies and Applications, Cloud Summit, (pp. 1–7) August 8-10 2019, Washington, DC, USA.

Tanenbaum, A. S., Mullender, S. J., and Van Renesse, R. (1986). *Using sparse capabilities in a distributed operating system*. In Proceedings of the IEEE 6th International Conference on Distributed Computing Systems (ICDCS 1986) (pp. 558-563), May 19-23, 1986, Cambridge, USA.

Lua (2021) *The Programming Language Lua*. Fourth edition of Programming in Lua. Retrieved on April 21, 2021 from http://www.lua.org/

Uddin, M., Islam, S., and Al-Nemrat, A. (2019). A dynamic access control model using authorising workflow and task-role-based access control. *IEEE Access*, *7*: 166676–166689.

Ungurean, I., Gaitan, N. C., and Gaitan, V. G. (2014). *An IoT architecture for things from industrial environment*. 2014 IEEE 10th International Conference on Communications (COMM), May 29-31, 2014, Bucharest, Romania.

Uriarte Itzazelaia, M., Astorga, J., Jacob, E., Huarte, M., and Romaña, P. (2018). Feasibility assessment of a fine-grained access control model on resource constrained sensors. *Sensors*, 18(2): 575.

Vijayalakshmi, V. A., and Arockiam, L. (2018). *Hybrid security techniques to protect sensitive data in E-healthcare systems*. Proceedings of the International Conference on Smart Systems and Inventive Technology (ICSSIT 2018) (pp. 39–43), Decemnber 13-14, 2018, Tirunelveli, India.

Wang, P., & Jiang, L. (2015). Task-role-based access control model in smart health-care system. In MATEC Web of Conferences (Vol. 22, p. 01011). EDP Sciences. May 29-30, 2015, Xiamen, China.

Xu, R., Nikouei, S. Y., Chen, Y., Blasch, E., and Aved, A. (2019). *BlendMAS: A blockchain-enabled decentralized microservices architecture for smart public safety*. Proceedings of 2019 2nd IEEE International Conference on Blockchain, (Blockchain) (pp. 564–571) . July 14-17 2019, Atlanta, GA, USA

Wikipedia (2022) *Cloud Computing*. Retrieved on January 29, 2022 from https://en.wikipedia.org/wiki/Cloud_computing

Zemmoudj, S., Bermad, N., and Omar, M. (2019). Context-aware pseudonymization and authorization model for IoT-based smart hospitals. *Journal of Ambient Intelligence* *and Humanized Computing*, *10*(11): 4473–4490.

Zerkouk, M., Mhamed, A., and Messabih, B. (2013). A User Profile Based Access Control Model and Architecture. *International Journal of Computer Networks & Communications*, *5*(1): 171–181.

Zhang, P., Chen, Z., Liu, J. K., Liang, K., and Liu, H. (2018). An efficient access control scheme with outsourcing capability and attribute update for fog computing. *Future Generation Computer Systems*, *78*: 753–762.

Zhang, Y., and He, J. (2016). *A proactive access control model based on stochastic game*. Proceedings of 2015 4th International Conference on Computer Science and Network Technology, ICCSNT 2015 (pp. 1008–1011), December 19-20, 2015, Harbin, China.