Secured System Architecture for the Internet of Things Using a Two Factor Authentication Protocol

by

Chuan Xiong

Thesis submitted to the University of Ottawa
In partial fulfillment of the requirements
For the Master of Applied Science in
Electrical and Computer Engineering

uOttawa

School of Electrical Engineering and Computer Science
Faculty of Engineering, University of Ottawa
Ottawa, Canada

Abstract

The security concerns for the Internet of Things (IoT) systems have become a very important topic due to the rapid development of IoT in many different fields including industrial, commercial, personal, etc. Since the protocols used for IoT communication are usually very light-weighted, it is more insecure when compared with internet protocols due to the nature of IoT devices been resource limited. The development of IoT systems has been rapidly growing, however there is still no groundbreaking development for the security aspect of the IoT system. Hence, the existing security mechanisms are either not applicable in IoT systems or can no longer satisfy the security requirements for IoT systems.

This thesis presents a solution that utilizes Two Factor Authentication (TFA) in the system's authentication protocol, which requires both the Server and the Broker in the IoT systems to function as the authentication guards. The solution is designed to be generic so that it can be deployed into virtually any IoT system, and more importantly meets the security requirements and guidelines for IoT systems. Specifically for this thesis, the TFA solution will be deployed on a fleet management system for experimental and testing purposes. This is also a demonstration of how this TFA protocol can be used in different IoT applications. The experimental part of this thesis will include a prototype that includes the three main components the Server, the Broker and the Client, which will be coded using Python, then use Wireshark to act as an eavesdropper on the system. For the protocol verification part, the TFA protocol was modelled as a Coloured Petri Net and uses a Coloured Petri Net simulation tool to prove that the modelled Net indeed possess the properties of Coloured Petri Net.

Acknowledgements

I would like to express my sincerest appreciation to my supervisor, Prof. Tet Yeap and Prof. Iluju Kiringa, for their guidance and support throughout my graduate study. Their patience, understanding and encouragement are precious for completing my research work. It is an honored for me to work with not just one, but two responsible and excellent supervisors. I was able to receive tremendous amount of knowledge and experience during my period of study, which did not only benefit my research journey at Uottawa, but will also help my further career as well. I am very grateful for everything that I have experienced while working with these two tough but also caring professors.

I would also like to thank my parents who unconditionally support and love me, as well as all the friends who supported and cared for me mentally when I was struggling. I appreciate the experience working in our IoT Research Lab, the colleagues in there are a great help towards completing my research. Specifically, I would like to thank Patrick, Sergei, Edward and Mehdi for their help and support, my work would be much more difficult without you guys.

Table of Contents

Abstract ii

Acknowledgements iii

Table of Contents iv

List of Figures vii

Nomenclature ix

1 Introduction **1**

1.1 Motivation 1

1.2 Problem Statement 3

1.3 Proposed Solution 4

1.4 Contributions 4

1.5 Thesis Outline 5

2 Related Work **7**

2.1 Internet of Things 7

2.1.1 What is an Internet of Things 7

2.1.2 General Architecture for the Internet of Things 9

2.1.3 Challenges and Future Development for the Internet of Things . . 14

2.2 IoT Communication Protocols in an Overview 17

2.3 Details on MQTT and Related Features 18

2.3.1 MQTT Protocol Overview 19

2.3.2 Publish and Subscribe 20

2.3.3 Topics 21

2.4 Security and Privacy Requirements for the Internet of Things 22

2.4.1 Security and Privacy Risks for IoT System 23

2.4.2 Security and Privacy Requirement Guideline for IoT System . . . 24

2.4.3 MQTT Security Related 29

2.4.4 Security Assessment for Current IoT Systems Based on Existing communication protocol . . . 32
2.5 MQTT VS HTTP . . . 35
2.6 Two Factor Authentication . . . 37
2.7 Protocol Verification Using Coloured Petri Nets . . . 39
2.7.1 Petri Nets . . . 41
2.7.2 Coloured Petri Nets . . . 42
2.7.3 Expressing Properties of protocols using CPN . . . 43
2.7.3.1 Properties of CPN . . . 43
2.7.4 Verifying protocols using CPN Tool . . . 49
2.8 Summary . . . 49

3 Two Factor Authentication Protocol for General IoT Architecture 50
3.1 TLS Quick Review . . . 51
3.2 Two Factor Authentication Protocol Architecture . . . 53
3.2.1 Two Factor Authentication Protocol without Broker Variation . . . 56
3.3 Two Factor Authentication Protocol with Certificate Authority - Further Enhancing Authorization and Access Controls . . . 57
3.3.1 Solution without CA . . . 58
3.4 Two Factor Authentication Protocol - Step by Step . . . 59
3.4.1 Authentication Procedures for Things . . . 60
3.4.2 Authentication Procedures for Users . . . 64
3.4.3 Out-of-Band Authentication Option - An Enhancement for Building Secured Channel . . . 65
3.4.4 TFA on Other Messaging Protocols . . . 66
3.4.5 Exception Handling . . . 66
3.5 Summary . . . 70

4 System Security Solution for Fleet Management IoT Architecture 71
4.1 IoT Architecture for Fleet Management System . . . 72
4.1.1 Control Plane . . . 72

4.1.2 Data Plane 74
4.1.3 Different Nodes in the Architecture 75
4.2 Two-Factor Authentication Protocol for the Fleet Management Architecture 76
4.3 Security and Privacy Related Issues with Fleet Management Architecture 79
4.4 Summary 79

5 Experiments and Evaluations 82
5.1 Out-of-Band Authentication Implementation 83
5.1.1 Pre-implementation Work 83
5.1.2 Implementation 84
5.2 Simulation Results and Evaluation of Two-Factor-Authentication Implementation 86
5.3 Coloured Petri Nets Simulation 92
5.3.1 Expressing the TFA Protocol Using CPN 92
5.3.2 Simulate the TFA Protocol using the CPN Tools 93
5.4 Summary 96

6 Conclusion and Future Work 98
6.1 Conclusions and Limitation 98
6.2 Future Work 99

References 101

List of Figures

2.1 Different approaches of integrating WSNs with the Internet [16]. 11
2.2 Internet of Things architecture [29]. 12
2.3 Three-layer concept of Iot architecture [18]. 13
2.4 Five-layer concept of Iot architecture [28]. 15
2.5 MQTT publish and subscribe model [36]. 20
2.6 MQTT Topic wildcards [36]. 22
2.7 Security assessment [36] [14]. 35
2.8 The average amount of data transmitted per message for different numbers of messages transmitted over the same connection [35]. 36
2.9 Transmission time for different message sizes [35]. 37
2.10 A List of protocol verification methods[8]. 39
2.11 An illustration of the complementary-place transformation:
(a) A finite-capacity net (N, M_0).
(b) The net (N', M_0') after the transformation.
(c) The reachability graph for the net (N, M_0) shown in (a) [1]. 44
2.12 Example nets of CPN properties [1]. 48

3.1 TLS handshake protocol [25] 52
3.2 Authentication process for existing IoT system 55
3.3 Two-Factor Authentication solution for IoT system 56
3.4 Security assessment with OOBA 57
3.5 Security assessment after incorporating CA 59
3.6 Existing TLS authentication protocol for IoT system 60
3.7 Two-factor Authentication solution for IoT system 61
3.8 Finite state machine for Server/Broker 67
3.9 Finite state machine for Things 69

4.1 Fleet management IoT architecture.[31]. 73
4.2 Relationship between different nodes in Fleet management IoT system [30]. 77
4.3 Two-Factor-Authentication sequence for Fleet management IoT system . 80

5.1 Authentication protocol sequence diagram 87
5.2 Connection refused by the server. 88
5.3 Connection refused by the server due to SSL related error. 88
5.4 Sample implementation source code. 89
5.5 Output result for Server component. 89
5.6 Output result for Client component. 90
5.7 Output result for Broker component. 90
5.8 Wireshark packet capture for non-encrypted MQTT connections. 91
5.9 Wireshark packet capture for non-encrypted MQTT connections. 91
5.10 Wireshark packet capture for encrypted TLS MQTT connections. 92
5.11 CPN - initial state 94
5.12 CPN simulation - step 1 94
5.13 CPN simulation - step 2 95
5.14 CPN - step 3 95
5.15 CPN - step 4 95

Nomenclature

IoT - Internet of Things

TCP - Transmission Control Protocol

TLS - Transport Layer Security

UDP - User Datagram Protocol

DTLS - Datagram Transport Layer Security

MQTT - Message Queue Telemetry Transport

AMQP - Advanced Messaging Queuing Protocol

CoAP - Constrained Application Protocol

XMPP - Extensible Messaging and Presence Protocol

NFC - Near Field Communication

QoS - Quality of Service

GSM - Global System for Mobile Communications

UMTS - Universal Mobile Telecommunications Service

DMZ - Demilitarized Zone

OSI model - Open Systems Interconnection model

T2T - Things to Things

P2P - Peer-To-Peer

DoS - Denial of Service

DDoS - Distributed Denial of Service

TFA - Two factor authentication

WSN - Wireless Sensor Network

RFID - Radio Frequency Identification

CPN - Coloured Petri Nets

CA - Certificate Authority

Chapter 1

Introduction

1.1 Motivation

With the rapid development of Wireless Sensor Network (WSN) technologies which has been deployed in many areas of modern infrastructure, this enables the ability of measuring, inferring and understanding the environment around us to a different level. These abilities range from simple environmental indicators like temperature and air humidity in our household, to delicate ecology monitors and industrial natural resource sensor networks. Radio Frequency Identification (RFID) is another key component of the wireless data communication. As each device or thing is coded by UID or EPC, then stored in a RFID electronic tag, this enables each device to be uniquely recognized. The con-

cept of IoT was first created when RFID is an essential component for each "thing" to be uniquely identified. As the technology progresses, RFID in certain applications can be replaced with Satelite, WiFi Bluetooth, or Near Field Communication (NFC) on the "things" [32]. Regardless of which wireless communication protocol was used in the application, they allow billions of devices and services to be connected and able to communicate with one another [14]. By having all the sensors and actuators blend seamlessly, and having them to communicating–actuating in a networked fashion, this creates the Internet of "Things" [12].

The concept of "Internet of Things" (IoT) is to create a network or multiple networks consist of many wired or wireless identifiable objects and devices. Within such a network, these things and devices are able to communicate, exchange data, and using any services provided within the network. The IoT facilitates the development of numerous applications, some of the IoT applications that have been widely deployed can be categorized into the following domains: personal and social, enterprises and industries, service and utility monitoring, plus mobility and transportation [14]. Regardless of whether the application is designed for regular users or to the enterprise sectors, it is extremely important to maintain the secrecy of the data from unauthorized people, as well as maintain a high degree of reliability of the service system. There had been countless breaches in security and privacy across so many different industries in the past years ever since the internet had become an indispensable part of the world. As the result, security and privacy is a major concern for developing an application. A more detailed guideline on security and privacy requirements will be discussed in Chapter 2.

Specifically for IoT systems, they are vulnerable to different types of attacks. The type of attacks can be either passive or active. Passive attacks are usually trying to sniff information from the network, but in general do not have an impact on the IoT system behaviours. However, active attacks are more direct and threatening in terms of the potential damage that can be done on the system. These two kinds of attacks can both originate from outside of the network or from internally. There are reports indicated that internal attacks tend to be more serious since the attacker is likely to know the system's flaws better than the external attackers. Since the communication protocol designed for

the IoT system required to be lightweight, it is more insecure when compared with others. Although there have been numerous techniques that people had developed in past years to secure different kinds of internet-related applications, however many of them are not suitable for IoT related architectures due to the nature of IoT devices been resource-limited. The development of IoT systems has been rapidly growing, however there is still no groundbreaking development for the security aspect of the IoT system. In the past, people have already adopted features such as TLS or DTLS to create a point-to-point secured communication channel, but there is no guarantee that the system is end-to-end secured in terms of communication that happens within the system. The main reason is that techniques such as TLS and DTLS are adopted in a broker based simple system that is designed for MQTT, not for a sophisticated IoT system. In an IoT system, if any point within the system has been compromised, the whole system's security will break down.

1.2 Problem Statement

In the previous section, it has already stated that the existing security techniques are no longer sufficient for current IoT systems. The current security infrastructure can only provide a point-to-point secured communication channel. The goal of this thesis is to design and implement an end-to-end secured system architecture that can be deployed on any IoT systems. In the other words, this architecture should be able to implement on any IoT related applications, which may vary from a small system that can only have a few devices to a larger scale such as a fleet management system for a local transportation company, or even to a greater scale that can include more than 10,000 devices and have them distributed all over the world. Therefore, the goal of this thesis is to design and verify a solution such that it is suitable for any IoT based system. This solution needs to work in a resource-constrained environment, and the system needs to be end-to-end secure.

1.3 Proposed Solution

With the problem been described in the previous section, it is clear that the solution will need to be very generic so that it will be able to be deployed into virtually any IoT system. In other words, it should not have too many requirements or restraints that will limit its applicable fields. Therefore, many advanced techniques that have already been developed in internet-related applications will not be feasible in this situation due to the resource limitation. To overcome the limitations in most of the IoT system architectures, the solution will take advantage of Two Factor Authentication (TFA) method, which has been used in many applications for strengthening authentication security. The biggest convenience given by the TFA solution is that it will not require complicated upgrades towards the existing IoT system, the more detailed solution will be explained in Chapter 3.

1.4 Contributions

The central problem that this thesis is trying to solve has been clearly stated, and the solution that will be used to solve this problem has also be proposed in the previous section as well. The following is a list of items that this thesis has initiated or achieved so that any future research and development can be done beyond this thesis:

- Proposed a generic Two Factor Authentication Protocol that can be applied to any IoT systems that require security and authentication enhancement. This protocol is able to solve the security issues that exist in current IoT systems. It makes use of a secondary authentication "Broker" as the second factor in the authentication process, such that a user or a device will not only have to authenticate into the broker, but also authenticate in the server before he can acquire the access privilege on the system. By adopting this technique, the IoT system will be end-to-end secure.

- Provided an extended solution using both CA and without CA to further address

the user privacy and access control related issues.

- Proposed possible extension for using the Out-of-Band Authentication as the secondary authentication factor in the TFA protocol.
- Proposed an architecture regarding the implementation of the TFA protocol in the Fleet Management IoT system.
- Verified the Two Factor Authentication Protocol using a CPN Tool, which is a protocol verification tool that examines the authenticity and stability of the protocol using the Coloured Petri Nets.
- Implemented a prototype of the TFA protocol. Python was used to code the three core components of the protocol to demonstrate that this protocol indeed works. The communication between the three clients was monitored using a packet capturing tool - Wireshark, to show that no information can be snipped during the communication.
- Provided some of the future research and development areas that this thesis is not able to cover in detail.

1.5 Thesis Outline

This thesis will be organized as follows:

1 Chapter 2 will be the related work, which will include all the important background information that is related to my work, as well as an evaluation of all existing solutions to IoT system security.

2 Chapter 3 will be describing the TFA solution to the problem been stated.

3 Chapter 4 will be describing the TFA solution specifically designed for the fleet management system.

4 Chapter 5 will be explaining how the TFA solution was implemented, as well as how CPN Tools was used to verify and evaluate the TFA protocol.

5 Chapter 6 will be the conclusion that is to summarize the work been done and evaluate the strength and weaknesses of this solution regarding the target goal been set at the very beginning; This chapter will also discuss some future work that can be done.

Chapter 2

Related Work

2.1 Internet of Things

2.1.1 What is an Internet of Things

The Internet of Things (IoT) is a novel paradigm that has been rapidly researched and developed in the modern wireless telecommunication field. There are many pieces of literature out there defining what is IoT, but in terms of how "Internet" and "Things" are been defined, there is not a widely accepted definition. The main reason behind this is that different authors have different opinions on what is considered as an IoT. Some people think that having a single sensor device connects to the internet is the

simplest example for IoT, while others think that IoT needs to have a framework that combines the devices and network together into a service that can be beneficial to a user group. According to [21], "IoT solutions are designed to make our knowledge of the world around us more timely and relevant by making it possible to get data about anything from anywhere at any time". Regardless how the definition of IoT will be refined as the time goes, the idea behind IoT remains the same: "the essence of the IoT is having interconnected devices that generate and exchange data from observations, facts, and other data, then make these data available to users" [21]. "The basic idea of this concept is the pervasive presence around us of a variety of things or objects – such as Radio-Frequency Identification (RFID) tags, sensors, actuators, mobile phones, etc. – which, through unique addressing schemes, are able to interact with each other and cooperate with their neighbors to reach common goals" [4]. To break down the definition for IoT, the concept of 'Things' in the context of IoT may refer to any real or virtual participating actors such as real-world objects, human beings, virtual data, and intelligent software agents. "The purpose of the IoT is to create an environment in which the basic information from any one of the networked autonomous actors can be efficiently shared with others in real-time. The definitions may arise from the word 'Internet' and lead to an 'Internet-oriented' vision, or 'things' and lead to a 'things oriented' vision. Putting the word 'Internet' and 'Things' together semantically means a world-wide network of interconnected objects uniquely addressable, based on standard communication protocols" [13].

From a technical point of view, the backbone of IoT is coming from the rapid development of Wireless Sensor Network (WSN) technologies that has been deployed in many areas of modern living infrastructure. This development enables the ability of measuring, inferring and understanding the environment around us, which ranges from simple environmental indicators like temperature and air humidity in our household, to delicate ecology monitors and industrial natural resource sensor networks. With these sensors and actuators blend seamlessly around us, as well as allowing them to communicate–actuating in a networked fashion, this creates the internet of "Things" [12]. Another fundamental feature is based on data communication tools, primarily RFID, which is an-

other key component to wireless data communication. As each device or thing is coded by UID or EPC, then stored in RFID electronic tag, this enables each device to be uniquely recognized. Although RFID seems to be the technology has been referred the most on IoT related articles, it is important to keep in mind that other technologies are available: Satelite, WiFi Bluetooth, or Near Field Communication (NFC) can be used on "things". Lastly, with the aid of networks and communication technologies (Wire and Wireless technologies: GSM and UMTS, Wi-Fi, Bluetooth, ZigBee), they allow billions of devices and services to be connected and able to communicate with one another[14][5]. These components are considered the building blocks or the backbones for the Internet of Things.

Some fundamental features of IoT systems include but not limited to the following [13]:

- IoT technology is network based, which means physical boundary is almost out of concern. Any object linked with the network can be incorporated into the IoT. Data communication is real-time or almost real-time over the IoT network, which makes IoT different from traditional databases or web systems.
- IoT is wireless and possesses the ability to provide comprehensive data about its surroundings.
- IoT has the ability to monitor the environment, as well as trace and track objects in a real-timed fashion. Such visibility enables the instant response to any exceptional event, distributed information sharing among multiple organizations, multiple users, and resource distribution.

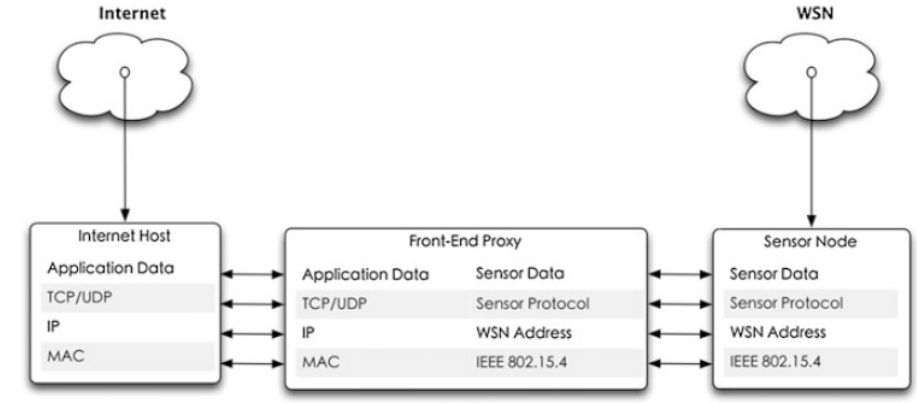
2.1.2 General Architecture for the Internet of Things

The concept of the Internet of Things is to create a network or multiple networks consists of many wired or wireless identifiable objects and devices that communicate, exchange data, and using services in the network. One of the major differences between the IoT and the Internet is that the terminal devices for IoT systems are smart things rather than computers, and WSNs are a network of interconnected smart things. Connecting

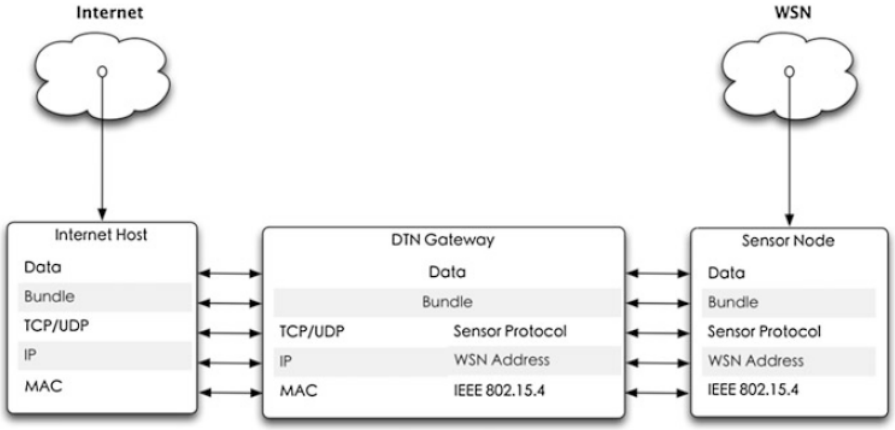
WSNs with the Internet will form an IoT information infrastructure. As shown in Figure 2.1, there are many ways of connecting WSNs to the Internet, which can be categorized into three main groups in terms of their communication architectures. The first approach is the frontend proxy solution, where the connection is performed by middleware proxy and there is no direct connection between WSNs and the Internet. The second approach is a gateway solution where a gateway is located between WSNs and the Internet. The final approach is the TCP/IP overlay solution, which is performed by an overlay network constructed on either WSNs or the Internet [16].

As IoT systems have been deployed into so many different industries, in order to facilitate all the devices in a complicated system into a network-fashioned environment, a well-designed architecture that is not only allowing the communication between WSNs to the internet, but also allows the inter-communications between all devices or things, servers and users. Regardless of which communication approach was used, there is always a sense of middleware within the infrastructure as shown in Figure 2.1 and Figure 2.2. The middleware layer can encapsulate many kinds of services and system functionality with the aid through cloud computing to implement information discovery. The Middleware has the option to implement data processing functionality locally or leave this functionality to the remote cloud server. In the first case, whenever the processed information indicating a critical event, it is then sent from the middleware layer to the application layer. Based on the information received, operators can then give further instructions to the things depending on specific services.

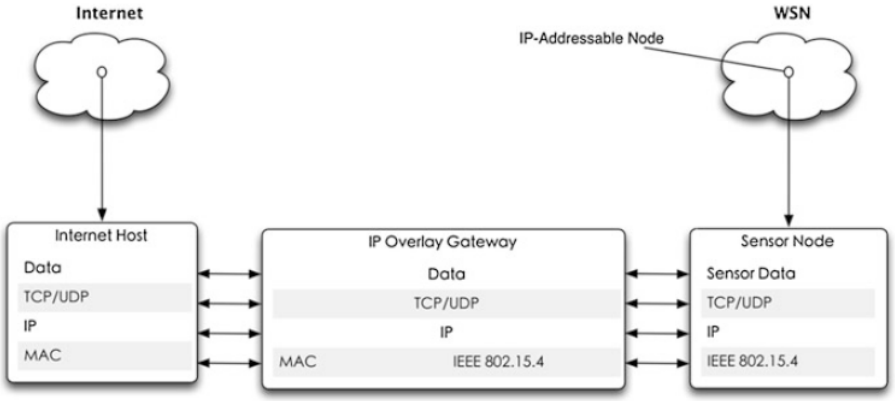
"The vision of IoT can be seen from two perspectives—'Internet' centric and 'Thing' centric. In the object-centric architecture, smart objects take the center stage. The Internet-centric architecture will involve internet services being the main focus while data is contributed by the objects "[12]. Internet-centric architecture has the potential of utilizing cloud computing and ubiquitous sensing with high scalability. "Sensing service providers can join the network and offer their data using a storage cloud; analytic tool developers can provide their software tools; artificial intelligence experts can provide their data mining and machine learning tools useful in converting information to knowledge and finally computer graphics designers can offer a variety of visualization tools. Cloud



Front-end proxy solution



Gateway solution



TCP/IP overlay solution

Figure 2.1: Different approaches of integrating WSNs with the Internet [16].

computing can offer these services as Infrastructures, Platforms or Software where the full potential of human creativity can be tapped using them as services" [12].

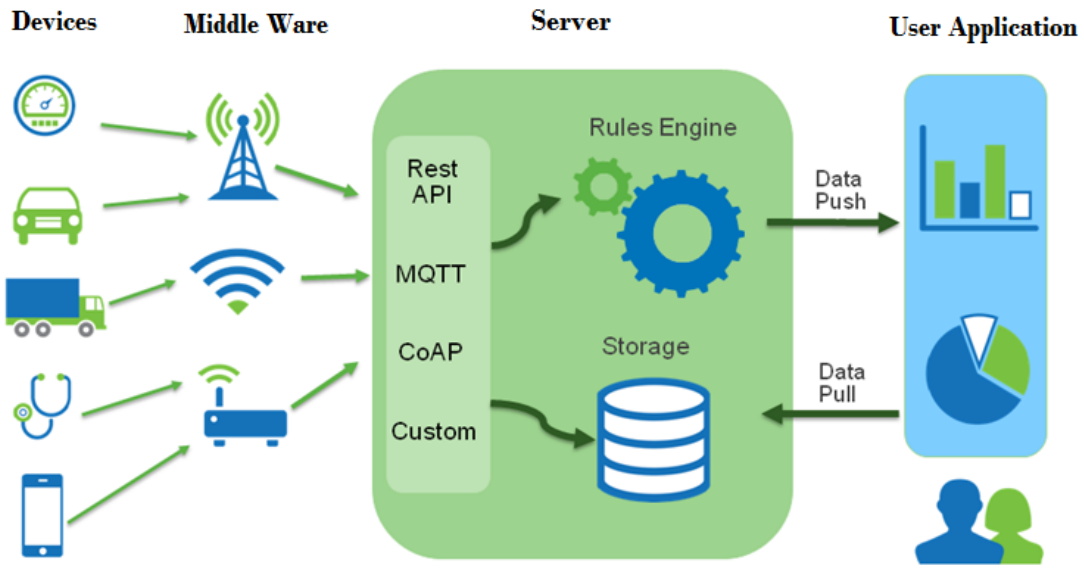


Figure 2.2: Internet of Things architecture [29].

From an architectural point of view, there is no single consensus on the design of IoT architecture that is agreed universally. Different architectures have been proposed by different researchers. In the field of IoT, many researchers inherit the networking layers' concept to define IoT architecture. Each layer is defined by its functionality and the devices that are used in that layer. Each layer of the IoT has inherent security issues associated with it as well. The most basic architecture is a three-layer architectural framework of IoT in regards to the devices and technologies that encompass each layer as shown in Figure 2.3 [28]. The fundamental functionalities of each layer are outlined as following[18][28]:

- The perception layer is the physical layer, which has sensors for sensing and gathering information about the environment. It senses some physical parameters or identifies other smart objects in the environment. The purpose of this layer is to acquire the data from the environment with the help of sensors and actuators.

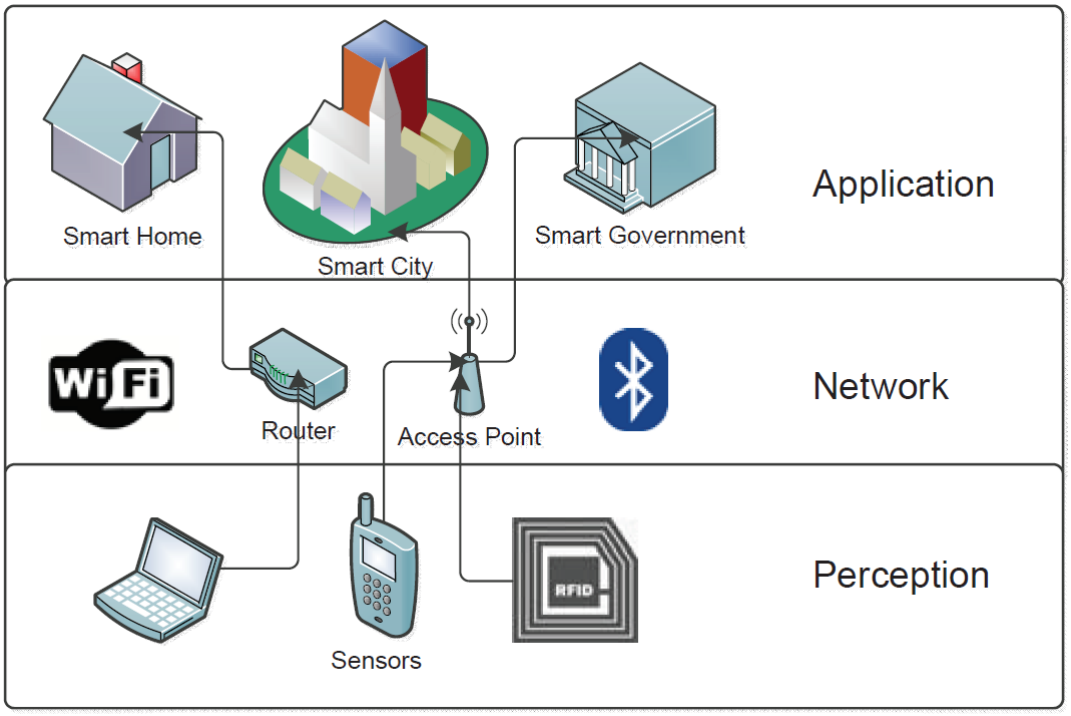


Figure 2.3: Three-layer concept of Iot architecture [18].

This layer detects, collects, and processes information and then transmits it to the network layer.

- The network layer is responsible for connecting to other smart things, network devices, and servers. Its features are also used for transmitting and processing sensor data. The network layer of IoT serves the function of data routing and transmission to different IoT hubs and devices over the Internet. At this layer, cloud computing platforms, Internet gateways, switching, and routing devices, etc. operate by using some of the very recent technologies such as WiFi, LTE, Bluetooth, 3G, Zigbee, etc. The network gateways serve as the mediator between different IoT nodes by aggregating, filtering, and transmitting data to and from different sensors.

- The application layer is responsible for delivering application-specific services to the user. It defines various applications in which the Internet of Things can be deployed, some examples include smart homes, smart cities, and smart health. The application layer guarantees the authenticity, integrity, and confidentiality of

the data. At this layer, the purpose of IoT or the creation of a smart environment is achieved.

The three-layer architecture gives the audience a general idea of the Internet of Things, but it is considered not sufficient for many IoT researchers because research often focuses on finer aspects of the Internet of Things. Hence, there is another five-layered architecture been described, which additionally includes the processing and business layers. The five layers are perception, transport, processing, application, and business layers (see Figure 2.4). The role of the perception and application layers is the same as the three-layered architecture. The functions of the remaining three layers are outlined as following [28] [9]:

- The transport layer transfers the sensor data from the perception layer to the processing layer and vice versa through networks such as wireless, 3G, LAN, Bluetooth, RFID, and NFC.

- The processing layer is also known as the middleware layer. It stores, analyzes, and processes huge amounts of data that come from the transport layer. It can manage and provide a diverse set of services to the lower layers. It employs many technologies such as databases, cloud computing, and big data processing modules.

- The business layer is responsible for the management of the overall system, which includes the applications, services, business and profit models, as well as users' privacy. It builds business models, graphs, flowcharts, etc based on the data received from the Application layer. Based on the analysis of results, this layer will help to determine future actions and business strategies.

2.1.3 Challenges and Future Development for the Internet of Things

The key functionality in the technology level is to enable interaction between the "things", which are considered as identification, sensing, storage, actuating, and other interfacing

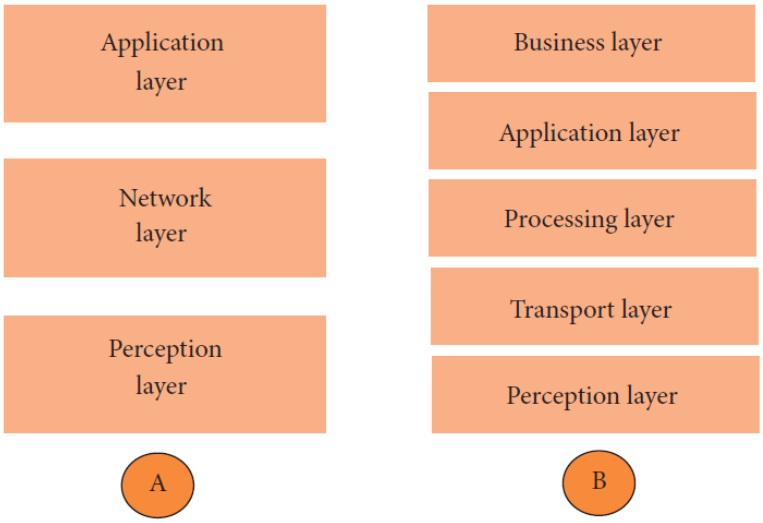


Figure 2.4: Five-layer concept of Iot architecture [28].

activities. The interface between the real and the digital worlds requires the capacity for the digital world to sense the real world and to act accordingly. However, simply equipping objects with microchips and retrieving information at a local level is insufficient. These smart 'network-enabled' objects need to be based on cheap and small wireless devices with sensing, acting, communication, and advanced signal and information processing capabilities [13].

The Internet has enabled developers to create solutions that produce data that can be viewed by anyone anywhere in the world. Incorporate the Internet into a prototypes can be a challenge. It is not as simple as taking a working solution on a local area network or other communication network, then add Internet connectivity on top of it. For example, growing a sensor network from a few sensors monitoring data viewed by a few people to a sensor network incorporating hundreds of sensors with the data viewable by potentially everyone may require redesigning your communication methods, data collection, and data storage. It is no longer a problem of figuring out how to scale the communication among the sensors, data collectors, and data-hosting services or database servers, but a even greater problem of how to deal with the explosion of data. Capturing data from

a dozen sensors is relatively easy and may not require careful planning, but capturing data from hundreds or even thousands of sensors is much more difficult because the data accumulates exponentially. Clearly, storing the data locally on the middleware is out of the question — especially so if considering how the data will be used. For example, what would happen if every device in your home, car, office, and so on, were to produce data? On top of this, the rising interest in wearable sensors and similar devices have the potential to generate more data than any human can manage or even decipher. The greatest concern of innovators in the emerging and developing world of the IoT is the potential for a data explosion as more and more devices generate, communicate, and present data. What we desperately need is a way to explore and exploit this data, as well as one place to start on how the data should be gathered and stored on a smaller scale like a sensor network [21].

Many challenging issues still need to be addressed in both technology and social aspects. The central issues are making full interoperability of interconnected devices possible, providing them with an even higher degree of smartness by enabling their adaptation and autonomous behavior, while guaranteeing trust, privacy, and security. Also, the IoT idea poses several new problems concerning the networking aspects. In fact, the things composing the IoT will be characterized by low resources in terms of both computation and energy capacity. Accordingly, the proposed solutions need to pay special attention to resource efficiency besides the obvious scalability problems. The main strength of the IoT idea is the high impact it will have on the everyday life and behavior of potential users. From a private user's point of view, the most obvious effects of the IoT introduction will be visible in both working and domestic fields. Similarly, from the perspective of business users, the most apparent consequences will be equally visible in fields such as automation and industrial manufacturing, logistics, business/process management, intelligent transportation of people and goods[4].

2.2 IoT Communication Protocols in an Overview

Due to the nature of IoT devices, there are many constraints and limitations on them. The limitations vary mostly from system resources such as memory, stable storage (such as disk space), transmission capacity and computational power. As the result, such device is not very like to have a sophisticated operating system installed on it; Hence, a complete 7-layer network model (also known as OSI models) for such device will not be available, therefore cuts down the safety level that can be deployed within the device.

To examine the safety levels of existing IoT systems, analyzing the communication protocols that have been used is an essential step. Several different protocols have been used in a variety of IoT applications, some of the important ones are listed below [23]:

1. Message Queue Telemetry Transport (MQTT) - a many-to-many communication protocol for passing messages between multiple clients through a central broker. It is suited to messaging for live data and MQTT clients make a long-lived outgoing TCP connection to the broker.

2. Advanced Messaging Queuing Protocol (AMQP) - an advanced messaging queuing protocol that allows messages exchange between applications. A new standard designed to meet the needs of the institutions enables inter-operation communication and showing resources between new and legacy applications. It allows businesses to work together to provide infrastructure. Middleware acts as a conduit between applications, businesses, and shared resources. It ties together organizations and technologies across time and space.

3. Constrained Application Protocol (CoAP) - a one-to-one protocol for transferring information between clients and servers. CoAP is state-based and not purely event-based. CoAP clients and servers both send and receive UDP packets. This protocol provides built-in support for content negotiation and discovery allowing devices to probe each other to find ways of exchanging data. It is an efficient RESTful protocol, ideal for constrained devices and networks. It is especially suited for M2M applications and provides an easy mapping to/from HTTP.

4. Data Distribution Service for Real-Time Systems (DDS) - DDS is a standard developed by Object Management Group (OMG). This standard is designed as a high-performance completely decentralized architecture that provides secure publish/subscribe. DDS provides dynamic discovery so that matching peers can communicate automatically. A rich set of QoS services provided by DDS ensures that all aspects of data distribution (such as availability, resource usage, traffic prioritization, etc.) can be controlled.

5. Extensible Messaging and Presence Protocol (XMPP) - Extensible messaging and presence protocol (XMPP) is a communication protocol for message-oriented middleware based on XML (extensible markup language). The protocol is designed to be extensible and is used for publish/subscribe systems, signaling for video file transfer, gaming, Internet of Things (IoT) applications such as the smart grid, and social networking services. Unlike most instant messaging protocols, XMPP is defined as an open standard and uses an open system approach of development and application by which anyone may implement an XMPP service and interoperate with other organizations implementations.

Regardless of the choice in communication protocols for the IoT system, the security and privacy protection provided by these protocols are either not present or relying on other techniques such as Transport Layer Security (TLS) in TCP or Datagram Transport Layer Security (DTLS) in UDP. These protocols use client-server based architecture with either publish/subscribe or request/response based model, where some protocols make use of a broker in the middle to decouple the client and server. Regardless of the difference in the implementation of these communication protocols, they all share some fatal flaws in protecting the security and privacy of the system. The details will be discussed and analyzed shortly.

2.3 Details on MQTT and Related Features

For this thesis, Message Queue Telemetry Transport (MQTT) will be discussed extensively as it will be used for the simulation and application. Although MQTT differs from

other IoT messaging protocols in terms of its implementation in certain aspects, however from the security and privacy architectural point of view, all of the commonly used messaging protocols for IoT system shares some similar flaws. Before diving into the details for the security and privacy aspects, we need to first use MQTT as an example to analyze its available features, then in the later sections try to uncover some of its major weaknesses in terms of security and privacy protections.

MQTT is a Client/Server based publish/subscribe protocol created by IBM and Eurotech. MQTT provides reliable and bidirectional message delivery services, and can run over TCP/IP or other network protocols; on top of that, it also adds packet loss protection against client disconnections. The design philosophy behind MQTT is to make it small and lightweight such that it is implementable on IoT devices with the constraints such as power or memory which was mentioned previously. MQTT makes use of small headers and avoids polling to increase network efficiency, and it is an event-driven protocol that provides a near real-time notification of events. Moreover, one MQTT server/broker is sufficient for supporting a million connected devices [23]. "These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things contexts where a small code footprint is required and/or network bandwidth is at a premium" [41].

2.3.1 MQTT Protocol Overview

MQTT uses a publish/subscribe model that is event-driven and enables messages to be pushed to clients. The broker is the central point for the communication system, which is responsible for receiving and dispatching all the message exchange between the publishers and subscribers. The message has been published to the broker will have to include a topic, which is nothing more complicated than a simple string that can have multiple hierarchy levels that are separated by a slash. The topic acts as the routing information for the broker; this information is used by a broker to redirect the message to the rightful receivers who are interested in this topic. The publish/subscribe model (pub/sub) is an alternative to the traditional client-server model, where a client communicates directly with an endpoint. However, in the publish/subscribe model, the sending and receiving

ends are decoupled as shown in Figure 2.5:

- Space decoupling: Publisher and subscriber do not need to know each other (by IP address and port for example)
- Time decoupling: Publisher and subscriber do not need to run at the same time.
- Synchronization decoupling: Operations on both components are not halted during publish or receiving

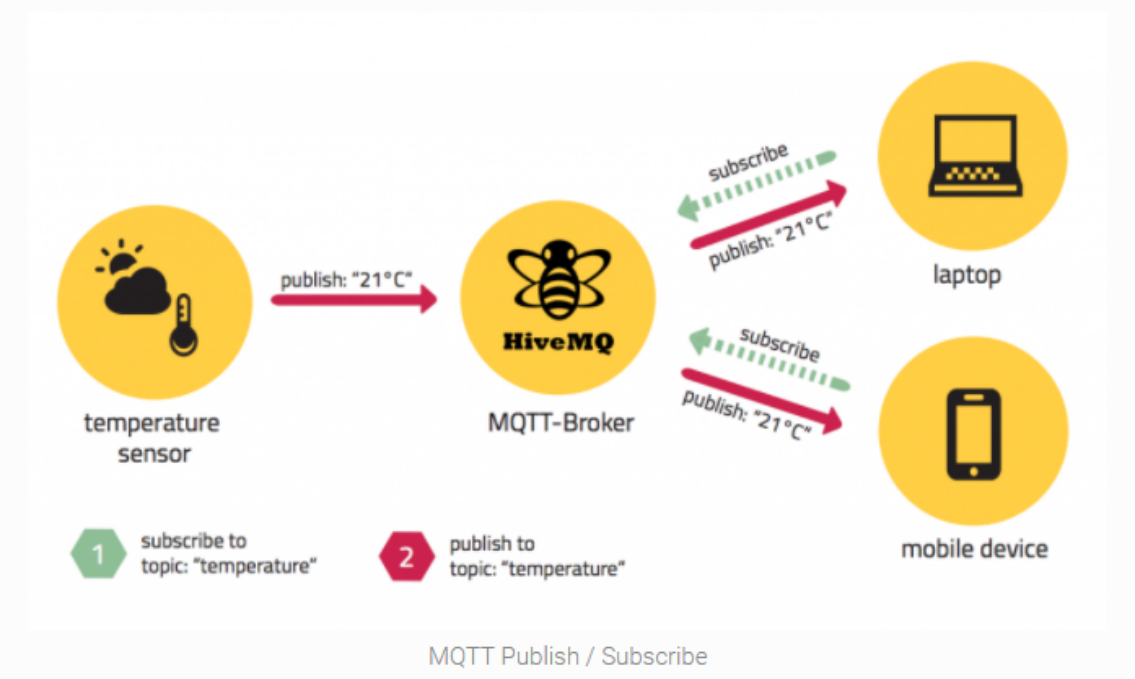


Figure 2.5: MQTT publish and subscribe model [36].

2.3.2 Publish and Subscribe

The MQTT connection is between the client and the broker, there are no direct connections between two clients. The connection is initiated through a client sending a CONNECT message to the broker, the broker then responses with a CONNACK and a status code. Once the connection is established, the broker will keep it open as long as the client does not send a disconnect command or loses the connection to the broker.

Once a connection is established between a client and a broker, clients can start to send and receive messages. MQTT has a topic-based filtering system for messages. Messages must contain a topic, and typically will have a payload that contains the actual message [36].

On the other hand, if a client wants to receive messages on certain topics, it needs to send a SUBSCRIBE message to the MQTT broker. A subscribe message contains a unique packet identifier and a list of subscriptions. The SUBSCRIBE message then will be confirmed by the broker by replying with a SUBACK message. The counterpart of the SUBSCRIBE message is the UNSUBSCRIBE message, which will request the broker to remove the existing subscriptions of a client. The broker will then acknowledge the request to unsubscribe with the UNSUBACK message [36].

2.3.3 Topics

As previously mentioned, the broker needs to use topics to decide which client receives what messages. A topic is a UTF-8 string, which can consist of one or more topic levels. Each topic level is separated by a forward slash (topic level separator). Since the topic is designed to be lightweight, therefore a client can simply publish or subscribe to a topic directly as a broker accepts each valid topic without any prior initialization.

The topic is case-sensitive, and wildcards can be used when subscribing to topics. There are types of wildcards, single level (+) or multi-level (#). As the name already suggests, a single level wildcard is a substitute for one topic level. In contrast, multi-level wildcard covers an arbitrary number of topic levels. To determine the matching topics it is required that the multi-level wildcard is always the last character in the topic and it is preceded by a forward slash. Figure 2.6 gives is a very simple example of how two kinds of wildcards are used in topics [36].

Figure 2.6: MQTT Topic wildcards [36].

2.4 Security and Privacy Requirements for the Internet of Things

In this section, we will provide an overview of the security and privacy risks that the current IoT systems are facing, then we will summarize a list of security and privacy requirements that many researchers have proposed for IoT systems. These security and privacy requirements will be described and put into different categories. Any IoT system needs to examine each category closely when measuring its security and privacy, and we will use these categories as the guideline to examine the current IoT architecture to see whether they have met the standard.

2.4.1 Security and Privacy Risks for IoT System

The IoT facilitates the development of numerous applications, Some of the IoT applications that have been widely deployed can be categorized into the following domains: personal and social, enterprises and industries, service and utility monitoring, and mobility and transportation [14]. Since such applications are likely to contain sensitive data, people's concerns in security and privacy aspects start to raise. "Privacy includes the concealment of personal information as well as the ability to control what happens with this information. The right to privacy can be considered as either a basic and inalienable human right, or as a personal right or possession"[5]. The attributes of this information may not be visible directly to the user, "individuals can be followed without them even knowing about it and would leave their data or at least traces thereof in cyberspace"[5]. Regarding the enterprise sectors, it is not only the matter of collecting respective data, but also maintaining the secrecy of the data from unauthorized people, as well as maintain a high degree of reliability for the service system. There are numerous reports on attacks being successful on existing IoT systems[19]:

1 One of the first successful attacks against industrial control systems was the Slammer worm, which infected two critical monitoring systems of a nuclear power plant in the U.S.A. in 2003.

2 In the same year, a computer virus infected the signal and dispatching control system of a major transportation network in the U.S.A. leading to a complete stop of passenger and freight trains.

3 In the following years, many security incidents affecting industrial control systems and critical infrastructure have been reported in the literature.

Many other scenarios had posed serious threats to IoT architectures if they have been attacked [10]:

- Disappearing computer or devices - the user lost the control and awareness of what is actually going on with his/her device or computer. This ubiquitous scenario can

be extended to the situation where the person does not know what data is collected about him/her, by what sensor, where and how they are sent, how processed, and whether some actuators would take an action towards him/her.

- The home medical advisor - it contains critical data that need to be transmitted securely between a patient to a health care center. There must be a secured communication path been established from the sensor on the patient to the health care center. Data must be encrypted on every part of this communication path, and can only be accessible by authorized individuals.

- Network cameras and microphones - Security of data been captured and transmitted by public web-cameras and microphones can generate voluminous data streams. The privacy of the person been breached is the issue here. Who can access the data, and what kind of operation can be performed on the data?

- The active badge and other location systems - This is another scenario regarding security and privacy on location-based systems. The data containing positions should be encrypted.

Of course, these scenarios are only a glimpse of the examples that IoT systems can be attacked or have security and privacy related concerns, there are many other discovered and undiscovered areas of security and privacy flaws in the IoT systems that need to be addressed.

2.4.2 Security and Privacy Requirement Guideline for IoT System

As the existing IoT infrastructures start to draw more attention from malicious attackers, many researchers started to look into the security and privacy guidelines for IoT systems. Many pieces of literature have defined the security and privacy requirements for IoT systems. Although there are variations among them, the following items are a good summary of what most of them have described[5]:

1. Resilience to attacks: The system has to avoid single points of failure and should adjust itself to node failures.

2. Data authentication: As a principle, retrieved address and object information must be authenticated.

3. Access control: Information providers must be able to implement access control on the data provided.

4. Client privacy: Measures need to be taken that only the information provider is able to infer from observing the use of the lookup system related to a specific customer; at least, inference should be very hard to conduct.

As previous mentioned, IoT systems are vulnerable to different types of attacks. The type of attacks can be either passive or active. Passive attacks are usually trying to sniff information from the network, but in general do not have an impact on the IoT system behaviours. However, active attacks are more direct and threatening in terms of the potential damage that can be done on the system. These two kinds of attacks can be both originated from outside of the network or from internally, where there are reports indicated that internal attacks tend to be more serious since the attacker is likely to know the system flaws better than the external attackers. "According to Computer Security Institute (CIS) and the FBI, approximately 60 percent to 80 percent of network misuse are originated from the inside network", and these kinds of attacks are usually in the form of one of the following[14]:

1. Intruder Model: A Dolev-Yao (DY) type of intruder shall generally be assumed. That is an intruder that is in effect the network and which may intercept all or any message ever transmitted between IoT devices and hubs.

2. Denial-of-service attacks (DoS): This kind of attack is an attempt to make a machine or network resource unavailable to its intended users - jamming channels, consumption of computational resources like bandwidth, memory, disk space, or processor time, and disruption of configuration information (such as node information).

3. Physical attacks: This type of attack tampers with hardware components.

4. Attacks on privacy: Since the IoT makes large volumes of information easily available through remote access mechanisms, privacy protection in IoT becomes increasingly challenging. The adversary need not be physically present to carry out surveillance, but information gathering can be done anonymously with very low risk. The most common attacks on user privacy are as follows:

    - Eavesdropping and passive monitoring: This is the most common and easiest form of attack on data privacy. If messages are not protected by cryptographic mechanisms, an adversary could easily understand the content.
    - Traffic analysis: To effectively attack privacy, eavesdropping should be combined with traffic analysis. Through effective traffic analysis, an adversary can identify certain information with special roles and activities in IoT devices and data.
    - Data mining: This enables attackers to discover information that is not anticipated in certain databases.

Referring to the three-layered IoT architecture introduced previously, we can further break down how these attacks can be threats to every layer as each IoT layer is susceptible to security threats and attacks described above. The following is a detailed presentation on the analysis of the security issues with respect to each layer [18].

1 Perception Layer: First, The wireless signals that are transmitted between sensor nodes of IoT using wireless technologies can be compromised by disturbing waves or other techniques which will result in reducing the strength of the wireless signals. Secondly, the sensor node in IoT devices are vulnerable to physical attacks due to the nature of IoT nodes usually operate in external and outdoor environments, leading to physical attacks on IoT sensors and devices in which an attacker can tamper the hardware components of the device. "Third is the inherent nature of network topology which is dynamic as the IoT nodes are often moved around different places. The IoT perception layer mostly consists of sensors and RFIDs, due

to which their storage capacity, power consumption, and computation capability are very limited making them susceptible to many kinds of threats and attacks. The confidentiality of this layer can easily be exploited by Replay Attack which can be made by spoofing, altering or replaying the identity information of one of the devices in IoT. Or the attacker might gain the encryption key by analyzing the required time to perform the encryption that is known as Timing Attack. Another confidentiality threatening attack is when the attacker takes over the node and captures all information and data which is the Node Capture attack. The attacker can add another node to the network that threatens the integrity of the data in this layer by sending Malicious Data. This can also lead to a DoS attack, by consuming the energy of the nodes in the system and depriving it from the sleep mode that the nodes use to save the energy." [18] [17] [11].

2 Network Layer: The network layer of IoT is susceptible to DoS attacks, and the adversary can also attack the confidentiality and privacy at the network layer by traffic analysis, eavesdropping, and passive monitoring [14]. These attacks have a high likelihood of occurrence as IoT systems often require the remote access mechanisms for the purpose of data exchange between devices. A Man-in-the-Middle attack is also frequently used against the network layer, the attackers then can follow up with eavesdropping on the communication channels. If the key material of the devices is stolen, the secure communication channel will be completely compromised. Attackers can also take advantage of the fact that everything is connected in order to gain more information about the users and use this information for future criminal activities [20]. "Protecting the network is important in the IoT, but also protecting the objects in the network is equally important. Objects must have the ability to know the state of the network and the ability to protect themselves from any attacks against the network. This can be achieved by having good protocols as well as software that enable objects to respond to any situations and behaviors that can be considered abnormal or may affect their security." [7].

3 Application Layer: "Since the IoT still does not have global policies and standards

that govern the interaction and the development of applications, there are many issues related to application security. Different applications have different authentication mechanisms, which makes integration of all of them very difficult to ensure data privacy and identity authentication. The large amounts of connected devices that share data will cause large overhead on applications that analyze the data, which can have a big impact on the availability of the services. Another issue that must be considered when designing the applications in IoT is how different users will interact with them, the amount of data that will be revealed, and who will be responsible for managing these applications. The users must have tools to control what data they want to disclose and must be aware of how the data will be used, by whom and when." [18].

To prevent these types of attacks or minimize the damage that can be done by them, there are several security and privacy challenges that any IoT system needs to overcome before they can declare themselves being secured. According to different systems, they may emphasize one or more of the following than the other, but overall these are the aspects that need to be focused on[14]:

1. User privacy and data protection: Things are connected, and data is communicated and exchanged over the internet, rendering user privacy a sensitive subject in many research works. Although an abundance of research has already been proposed concerning privacy, many topics still need further investigation.

2. Authentication and identity management: Authentication and IdM are a combination of processes and technologies aimed at managing and securing access to information and resources while also protecting things profiles. IdM uniquely identifies objects, and authentication entails validating the identity establishment between two communicating parties.

3. Trust management and policy integration: When many things communicate in an uncertain IoT environment, trust plays an important role in establishing secure communication between things. Two dimensions of trust should be considered in

IoT: trust in the interactions between entities, and trust in the system from the users perspective

4. Authorization and access control: Authorization enables determining if the person or object, once identified, is permitted to have the resource. Access control means controlling access to resources by granting or denying according to a wide range of criteria.

5. End-to-End Security: Security at the endpoints between IoT devices and Internet hosts is likewise important. Applying cryptographic schemes for encryption and authentication codes to packets is not sufficient for resource-constrained IoT.

2.4.3 MQTT Security Related

Since MQTT is designed to be a lightweight protocol, as the result the protocol itself does not have many security mechanisms implemented within. To provide security enhancement, MQTT can make use of other widely accepted security mechanisms at different OSI model layers to guard against different kinds of security risks. We will examine the security mechanisms that are available to MQTT, and determine whether these features are able to withstand the different types of attacks, as well as meeting the security standards that will be discussed in the next section.

1. Authentication: It is the process of which a device or user trying to obtain certain access into a system by letting the system to identify themselves. This process usually is done through a username and password scheme.

   In the case of MQTT, it identifies a client by a unique client identifier based on username and password, or other unique information such as MAC address. The MQTT server can authenticate the client on two different layers. One is done by using the TLS protocol to authenticate the client certificate on the transport layer; another part is using a password sent by the client and authenticate the client on the application layer. On the other hand, the client authenticates the MQTT server by authenticating the certificate sent back by the server. Authentication

at the transport layer is the job of TLS to authenticate the digital certificates of client and server. On the application layer, a combination of client username and password provides the means for authentication. If the default authentication mechanism provided by MQTT is used, the username and password sent to the broker are in plain text, hence giving eavesdroppers a very easy time to compromise client credentials. Only transport layer encryption can provide a completely secure transmission channel of user credentials to the broker [23].

Another possible authentication source from the client is an X.509 client certificate. A client presents its X.509 certificate to the broker during the MQTT TLS handshake. Once the TLS handshake is successful, it is also possible to use the information in the certificate for Application Layer authentication. This way, a broker can read all information in the certificate and can use this information for authentication [23].

2. Authorization: Authorization is the idea of specifying who has access to what resources. In MQTT, authorization is not done directly in the implementation of the protocol itself, rather it is usually implemented in the MQTT broker. It gives broker control on the topics of which the client can publish or subscribe to. MQTT allows a variety range of possible topic permissions. Hence, the brokers can leverage these topic permissions to specify authorization policies for clients and to limit their access in publishing/subscribing unauthorized messages.

   Another authorization mechanism available for MQTT is OAuth, which is an open standard for authorization. "OAuth provides client applications a secure delegated access to server resources on behalf of a resource owner. In OAuth implementations, an authorization server, with the approval of the resource owner, issues access tokens to the third-party clients. The client then uses the access token to access the protected resources hosted by the resource server" [23].

3. Data Integrity: An IoT system is often hosted in an environment with many untrusted clients or clients all over the place, and the system has no control over them. To avoid any potential risks this may cause, checking data integrity of re-

ceived messages is essential. With the integrity check, both client and server can ensure that nobody has modified the message, or at very least able to alert themselves that data has been tampered with. To prevent data tampering from happening, two processes need to be present to make such detection work - Encryption and Integrity check:

- Encryption: encryption works at the application layer, and TLS in comparison works on the network layer. The key function of the encryption mechanism is to protect application messages against eavesdroppers. It is still possible that an attacker can replay the entire message or parts of the message if the messages are not communicated on a secure communication channel (e.g. TLS) [23].
  - Asymmetric Encryption: also known as public/private key encryption, where the public key is used for encrypting the data and the private key is used for decrypting the data. Once data is encrypted with a public key, it can only be decrypted with a private key.
  - Symmetric Encryption: also called secret key algorithm, is an algorithm that uses the same cryptographic keys for both encrypt and decrypt the message. This key should be kept secret given by its name, as once the key is stolen all messages transmitted can be read and modified by the third party.
  - Payload Encryption: Payload encryption is the encryption of application-specific data on the application level. Payload encryption provides end-to-end encryption for application data. Message metadata remains intact while the payload of the message is encrypted [23].
- Data integrity: The mechanism available here include Digital Signatures, Message Authentication Code Algorithms (MACs), and Checksums. In short, these three mechanisms all provide the functionality for checking the integrity of the data, but they vary in terms of the complexity of the algorithm behind them. Taking Checksum as an example, if an attacker figures out the

checksum algorithm, they will be able to alter both the message content and the checksum code goes along with it. Therefore, it is a trade-off between the computational complexity and the risk of having data been tampered while you think the message is still legit.

4. Infrastructure Security

   - Firewall: It is there to implement and enforce sophisticated rules aimed at blocking attackers. The most appropriate firewall rules vary from case to case, but a firewall should allow only expected traffics in general.
   - Load Balancer: "The function of a load balancer is to distribute protocol traffic to different brokers. If brokers are operated in a cluster environment. A load balancer typically does not provide much security but can prevent overloading of brokers and can also throttle unusually high traffic" [23].
   - Demilitarized Zone (DMZ): "A DMZ is a subnetwork that protects access to all downstream objects by providing an additional firewall. The purpose of the DMZ is to stop the attacker at the DMZ level by using a dual firewall system. A DMZ should utilize two firewalls from different vendors. This is because the same exploitation techniques known for one firewall cannot be used against the second firewall" [23].

2.4.4 Security Assessment for Current IoT Systems Based on Existing communication protocol

To quickly wrap up, we need to assess whether these security mechanisms are enough to satisfy the security guidelines specified previously. Along with all of the other mechanisms mentioned above, it seems at first glance that all of the security and privacy concerns outlined in the previous section has been covered, but if we are to look at the IoT system as a whole, there are many potential threats to the security and the user privacy:

- User privacy and data protection: Need to rely on properly regulated authorization and access control provided by the system, otherwise anyone can potentially access

any information that is at rest inside the MQTT broker; the devices themselves are also exposed to physical attacks as well, which leads to information stored the device been stolen as well.

- Authentication and identity management: Current systems provide a point-to-point security channel, but not too much was done to handle man-in-the-middle attacks. For example, someone was able to get his hand on an IoT device and can modify the server's IP address on one of the devices. As the result, all authentication certificates can be stolen, and the attacker is able to become the middle man between the device and the server without been detected.

- Trust management and policy integration: The same issues with authentication and identities. We can only assure that the within the communication between device and MQTT broker, two entities can verify each others' identity; any communication beyond the point-to-point is not verifiable at the moment, and there is no good way to detect such events from happening.

- Authorization and access control: Currently, we need to fully rely on people to create and properly regulate these rules set in place. If there is a flaw in implementing these rules, sensitive information can be accessed by unauthorized users.

- End-to-End security: In terms of point-to-point security, the communication channel between two entities such as device and broker are secured with the security mechanism in place. However, if we are to look at the communication between a device and a management user on a mobile device, since there is a MQTT broker in the middle, there could be many potential attacks coming in.

To summarize, since MQTT uses TCP, the communication channel between two entities can be established using TLS which makes things in encrypted form, and it is the most essential part to establish a secured point-to-point communication channel. However, the drawback of using MQTT over TLS is the additional CPU usage and time consumed for the TLS handshake process. This CPU usage can be negligible for the MQTT brokers, but it may be significant for IoT devices that are not designed for resource-intensive

tasks. There is this session resumption technique available for TLS that can greatly improve TLS performance by allowing reuse of an already negotiated TLS session after a client reconnected to the server. This way, a much simplified TLS handshake between client and server is required instead of the full TLS handshake process. In the case where safety level is a major concern, a secured communication channel established by TLS should always be adopted.

One of the major issues for TLS is that it only provides a point-to-point security channel, any communication made beyond point-to-point can not guarantee its security. A very straight forward illustration is that we can make sure the communication channel between the IoT device and MQTT broker is secured, but if a mobile user wants to access or monitor the device through the MQTT broker, we cannot guarantee that from user end to device end we have a completely secured communication path as there are three or more points between the communication path. If any of these entities within the communication path is compromised, there is no way to detect such events from happening. This could result in scenarios such as a compromised mobile user authenticated into the server, then acquired the information of the IoT devices; but IoT devices will think the user is still legitimate and allows the user to access the data. In such a scenario, a secured channel between two points becomes meaningless, since if any point within the system is compromised, the whole system's security will just collapse. This example brings up a major issue in the IoT system, which is IoT devices are very easily been exposed to physical attacks, hence we cannot guarantee the integrity of the device part themselves. As the result, the software on the device could be modified, and furthermore risking the server security been bypassed using reverse engineer techniques, which ultimately results in all of the categories in the outline been exposed to risks. On the infrastructural side, there are many uncertainties such as DOS or DDoS attacks, this will require a well-implemented secured system outlined in the infrastructure security section to guard against such threats. To wrap up in short, a more sophisticated security infrastructure needs to be designed to detect, notify and even prevent such intrusion from happening.

In Figure 2.7 below, it outlines where the current IoT system is lacking in regards to the security and privacy requirements that have been outlined previously.

Categories	is it secured	Issues
User privacy and data protection	No	Need to rely on properly regulated authorization and access controls provided by the system, otherwise anyone can potentially access any information that are at rest inside the MQTT broker. The devices themselves are also exposed to physical attacks as well, which leads to information stored the device been stolen as well.
Authentication and identity management	No	Current systems provide a point-to-point security channel, but not too much was done to handle man-in-the-middle attacks.
Trust management and policy integration	No	Same issues with authentication and identities. We can only assure that within the communication between device and MQTT broker, two entities can verify each others' identity
Authorization and access control	No	Need to fully rely on people to create and properly regulate these rules set in place. If there is a flaw in implementing these rules, sensitive information can be accessed by unauthorized users
End-to-End security	No	Only point-to point is secured, if the communication path has three or more points, there is no good way to ensure nothing happens in between

Figure 2.7: Security assessment [36] [14].

As been illustrated here, the current IoT system cannot fulfill the security and privacy requirements that have been generally accepted, therefore improvements must be implemented to add more security levels on the current IoT system is mandatory to satisfy the increasing demand in security and privacy protections in the current society.

2.5 MQTT VS HTTP

Having introduced MQTT in the previous section, people may start to wonder why do we even need MQTT as HTTP has been so widely used in the world of internet?

According to a series of experiments comparing MQTT and HTTP conducted by Google Cloud Solution team, the results will provide an answer to why MQTT is a preferred choice by most of the people in the IoT field [35]. The test cases include the following:

- MQTT: Varying the number of messages: measured the response time for sending 1, 100, and 1000 messages over a single connection cycle each, and also captured the packet sizes that were sent over the wire. Varying the size of messages: measured the response time for sending a single message with 1, 10, and 100 property fields over a single connection cycle each, and then capture the packet size sent.
- HTTP: Measured the average response time for sending a payload with 1, 10, and 100 property fields and then capture the packet size over the wire.

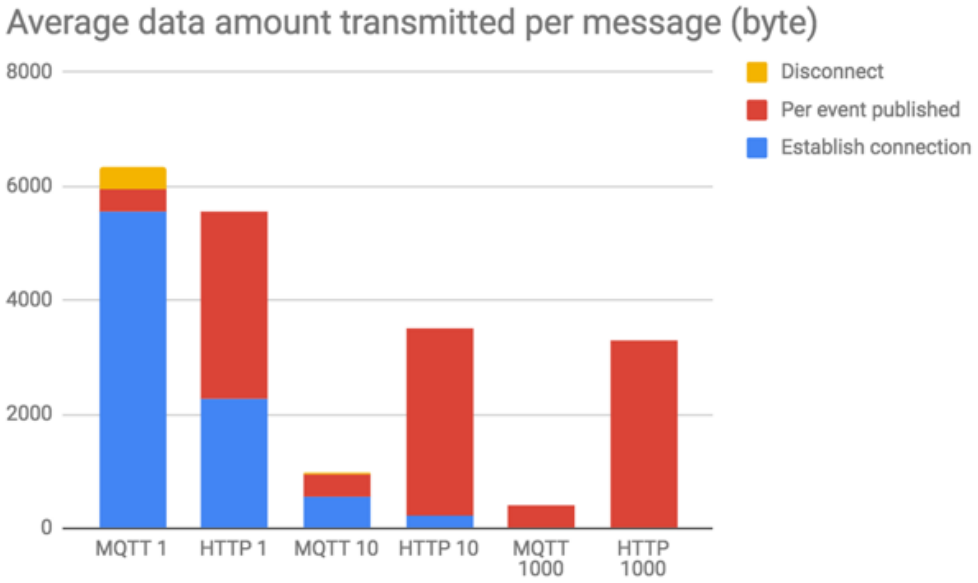


Figure 2.8: The average amount of data transmitted per message for different numbers of messages transmitted over the same connection [35].

From the result of the test as shown in Figure 2.8 and Figure 2.8, the key advantage of MQTT over HTTP according to [35] is when a single connection is used to send multiple

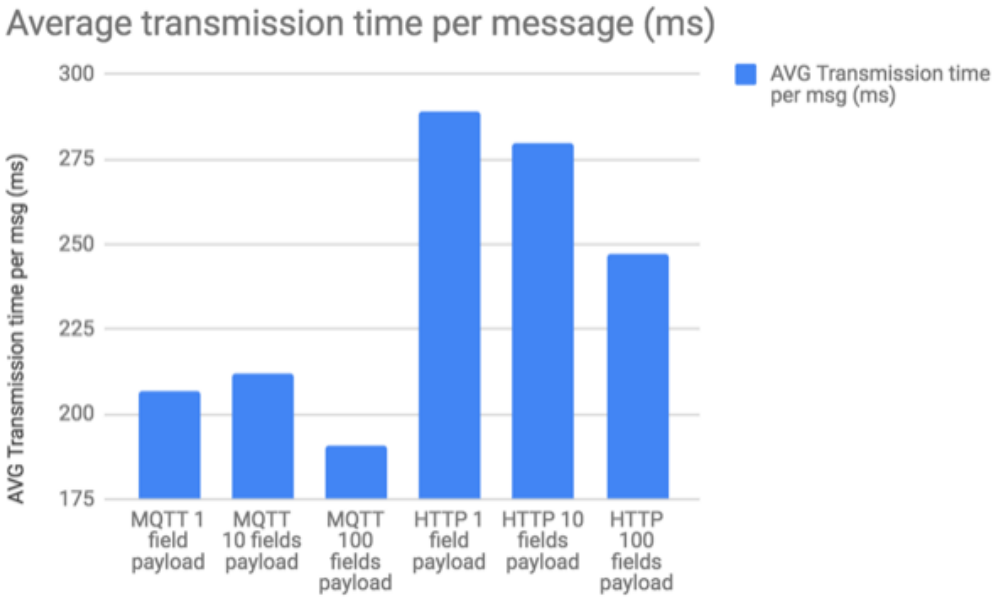


Figure 2.9: Transmission time for different message sizes [35].

messages. The conclusion can draw is that when choosing MQTT over HTTP, it is really important to reuse the same connection as much as possible. If connections are set up and torn down frequently just to send individual messages, the efficiency gains are not significant compared to HTTP. The greatest efficiency gains can be achieved through MQTT's increase in information density for each payload message. These conclusions are aligning with the MQTT description of been lightweight.

2.6 Two Factor Authentication

Given the fact that single-factor authentication mechanics such as TLS and DTLS are no longer sufficient to provide a system with end-to-end security in an IoT based architecture, as well as the computing and physical constraints that are present for IoT based devices, other methodologies need to be considered to build an end-to-end secured IoT system. Looking at the world of the Internet, which has been rapidly developed with remarkable success for the past decades. In terms of security and privacy, single-factor

authentication methods such as passwords, are no longer considered secure in the World Wide Web. "It has never been less difficult in Securing the system and remote access. The security and privacy threats through malware are always constantly growing both in quantity as well as quality. Expanded access to information increases weakness to hacking, cracking of passwords and online frauds" [26]. According to [13], out of 6,275,980 data records that were lost or stolen since 2013 breaches, only 4% of breaches were "Secure Breaches" where encryption was used and the stolen data was rendered useless. That is, every 25 transaction, one of them is breached.

One of the most common methodologies that many institutions incorporate into their security systems is Two-Factor Authentication (TFA). The TFA proposal guarantees a higher protection level by extending the single authentication factor. To summarize, Two-Factor Authentication is the process of identifying an online user by validating two or more claims presented by the user, each from a different category of factors [34]. The key towards two-factor or multi-factor authentication is that the factors are: Something the user knows, such as a password or PIN; Something the user has, such as a mobile device or security token; Something the user is, such as fingerprint, optics, voice or unique facial recognition.

There is no perfect authentication factor, any implementation for an authentication method will have its strengths and weaknesses. By incorporating one or more authentication factors into the system, they can compensate for the weakness of the other factors. For example, taking the first authentication factor been "something the user knows", such as password or pin, which can be susceptible to brute-force or social engineering attacks. Such attacks can be supplemented by adding the second authentication factor of "something you have" such as authenticating users through their mobile device or through "something you are" like a biometrics factor like fingerprint or voice. Unless the hacker has all of the factors required by the system, they will not be able to access the account [34]. Taking banking transactions as an example, when a customer wants to do an online bank transaction, TFA will send an SMS message that contains a one-time login token to the phone number provided by the customer. This way, any hackers or identity thieves will not be able to access this account as they cannot obtain

the one-time login token. "Sophisticated hacking that can access the SMS message and password is often called a man-in-the-middle attack. In general, a man-in-the-middle attack involves creating a dummy network that will trick the victim into thinking that it is a legitimate network. If the actor can intercept the user's cell phone communications, it may be possible to overcome out-of-band authentication security protocols" [44].

Method	Year	Researcher
Algorithmic method	1988	Krishan Sabnani
Petri net	1990	Ichim Suzuki
CFSM	1993	Chung-Ming Huang and Duen-Tay Huang
ECFSM	1994	Chung-Ming Huang and Jenq-Muh Hsu
Estelle	1995	Ajin Jirachiefpattana and Richard Lai
Path based approach	2000	W.-C. Liu, C.-G. Chung
Process algebra	2009	Slavomır Simonak, Stefan Hudak and Stefan Korecko
Coloured Petri nets	2010	Jing LIU, Xinming YE, Jun ZHANG, Jun LI, Yi SUN

Figure 2.10: A List of protocol verification methods[8].

2.7 Protocol Verification Using Coloured Petri Nets

According to [8] and [2], there is not much progress in the field of protocol verification techniques, especially for practical verification of communication protocols. There is a big gap between practical protocol verification techniques and the theoretical research results published in journals and conference papers. "This big gap between academic and industrial practices is the fact that academia has not been addressing the verification

issues and problems account for the fact that academic verification methods are seldom used in industry" [8]. As shown in Figure 2.10, the latest protocol verification is the Coloured Petri nets which came out back in 2010. CPN is been selected as the tool for the simulation and the verification part of this thesis is because CPN is the most recent and there are numerous existing tools that are well documented in terms of how to start it with. On the other hand, the purpose here is to pick a tool that can do the protocol verification, and all the methods listed are capable of doing such thing, therefore CPN edges out in terms of the difficulties of building the simulation models.

In the world of designing network protocols, verification is a crucial step to eliminate weaknesses and inaccuracies of effective network protocols. There are many models and tools to verify network protocols, regardless of which tool will be used, the purpose is to catch and identify the following six general types of protocol errors [8]:

- Unspecified Reception: An unspecified reception is a reception that is executable but not specified in the design. Since the required reception is not specified in the design, the occurrence of it means the subsequent behavior of the system interaction is unpredictable.

- Nonexecutable Interaction: A nonexecutable interaction is a transmission or reception that is specified in the design but never be executed. It is often called dead code.

- Deadlock: A state deadlock occurs when each and every process can only remain indefinitely in the same state. In other words, a deadlock is a global state reachable from the initial global state, with all channels empty, in which no transmissions are possible. Deadlocks must be avoided since once the system has arrived in a deadlock state, it is blocked forever.

- Livelock: A livelock is a situation where several processes keep exchanging messages but without any effective work being accomplished toward performing its services. It is sometimes called "dynamic deadlock". Livelock has to be avoided since once it occurs, the system is locked forever into a small set of global states. It is sometimes called Temp o-blocking.

- State Ambiguity: A state ambiguity exists when a state in one process can coexist stably (i.e. reachable with channels empty) with more than one state in another process. In practice, processes within a communication network often execute in a highly synchronous manner. Thus, although state ambiguities do not necessary represent errors, they should be more carefully treated.

- Overflow: An overflow is a channel state such that the number of messages in the channel is not bounded by a predefined positive integer number. In other words, any communications channel has a storage capacity that limits the amount of information it can carry at any instant. An attempt to exceed this capacity may result in loss of data being transferred among processes. Overflow may occur in two different situations: finite overflow, or infinite overflow. Finite overflow indicates that channels are bounded by one number, although not the predefined one; whereas infinite overflow indicates that channels are never bounded. While the former may be dynamically solved; the latter gives a serious sign of the existence of potential design errors.

2.7.1 Petri Nets

Petri Nets is a mathematical modeling language for describing distributed systems. It is a class of discrete event dynamic systems. According to multiple sources, Petri Nets were originally invented by Carl Adam Petri in 1939 to describing chemical processes [45]. "Petri Nets offer a graphical notation for stepwise processes that include choice, iteration, and concurrent execution. Unlike these standards, Petri nets have an exact mathematical definition of their execution semantics, with a well-developed mathematical theory for process analysis" [45].

> "A Petri Net is a particular kind of directed graphs with an initial state called the initial marking. The underlying graph N of a Petri net is a directed, weighted, bipartite graph consisting of two kinds of nodes, called places and transitions, where arcs are either from a place to a transition or from a transition to a place. In the graphical representation, places are drawn

as circles, transitions as bars or boxes. Arcs are labeled with their weights (positive integers), where a k-weighted arc can be interpreted as the set of k parallel arcs. Labels for unity weight are usually omitted. A marking (state) assigns to each place a non-negative integer. If a marking assigns to place p a non-negative integer k, we say that p is marked with k tokens. Pictorially, we place k black dots (tokens) in place P. A marking is denoted by M, an m-vector, where m is the total number of places. The p^{th} component of M, denoted by M(p), is the number of tokens in place p. In modeling, using the concept of conditions and events, places represent conditions, and transitions represent events. A transition (an event) has a certain number of input and output places representing the pre-conditions and post-conditions of the event, respectively. The presence of a token in a place is interpreted as holding the truth of the condition associated with the place. In another interpretation, k tokens are put in a place to indicate that k data items or resources are available" [1].

2.7.2 Coloured Petri Nets

Coloured Petri Nets (CPN) is a discrete-event modeling language combining the capabilities of both Petri nets and high-level programming language. "Petri Nets provide the foundation of the graphical notation and the basic primitives for modeling concurrency, communication, and synchronization. The CPN ML programming language, which is based on the functional programming language Standard ML, provides the primitives for the definition of data types, for describing data manipulation, and for creating compact and parameterizable models" [3].

Petri Nets can be categorized into low-level and high-level Petri Nets. CPN belongs to the class of high-level Petri Nets since it is the combination of Petri Nets and programming languages. Low-level Petri Nets are primarily used as theoretical models for concurrency, as well as for modeling and verification of hardware systems. High-level Petri Nets are aimed towards practical use since they can construct compact and parameterized models. The CPN modeling language is a general-purpose modeling language. It

is not only capable of modeling a specific kind of systems, but is aimed towards modeling a very broad class of systems that can be characterized as concurrent systems. Typical application domains of CPN are communication protocols, data networks, distributed algorithms, and embedded systems. "A CPN model of a system is both state and action-oriented. It describes the states of the system and the events (transitions) that can cause the system to change state. By performing simulations of the CPN model, it is possible to investigate different scenarios and explore the behaviour of the system" [3].

2.7.3 Expressing Properties of protocols using CPN

2.7.3.1 Properties of CPN

After the system or protocol is modeled by CPN, what can it prove or what properties can we get out of this modeled system? A major strength of Petri nets is their support for analysis of many properties and problems associated with concurrent systems. Two types of properties can be studied with a Petri-net model: those which depend on the initial marking, and those which are independent of the initial marking. The former type of properties is referred to as marking-dependent or behavioral properties, whereas the latter type of properties is called structural properties. Both types of properties will be described separately. First, behavioral properties include the following [1]:

- Reachability: Reachability is a fundamental basis for studying the dynamic properties of any system. The firing of an enabled transition will change the token distribution (marking) in a net according to the transition rule. A sequence of firings will result in a sequence of markings. A marking M_n, is said to be reachable from a marking M_0 if there exists a sequence of firings that transforms M_0 to M_n. A firing or occurrence sequence is denoted by $\sigma = M_0 t_1 M_1 t_2 M_2 ... t_n M_n$, or simply $\sigma = t_l t_2 ... t_n$. In this case, M_n is reachable from M_0 by σ and we write $M_0[\sigma > M_n$. The set of all possible markings reachable from M_0 in a net (N, M_0) is denoted by $L(N, M_0)$ or simply $L(M_0)$. The set of all possible firing sequences from M_0 in a net (N, M_0) is denoted by $L(N, M_0)$ or simply $L(M_0)$ [1].

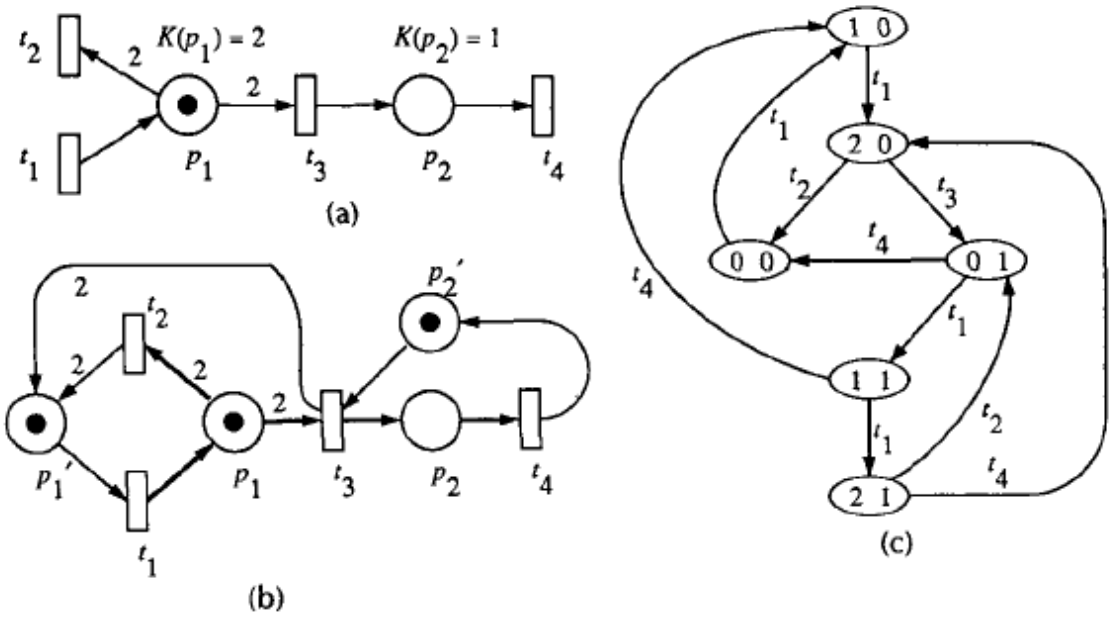


Figure 2.11: An illustration of the complementary-place transformation:
(a) A finite-capacity net (N, M_0).
(b) The net (N', M_0') after the transformation.
(c) The reachability graph for the net (N, M_0) shown in (a) [1].

- Boundedness: A Petri net (N, M_0) is said to be k-bounded or simply bounded if the number of tokens in each place does not exceed a finite number k for any marking reachable from (M_0), i.e., $M(p) \leq k$ for every place p and every marking $M \epsilon R((M_0))$. A Petri net $(N, M_0)$is said to be safe if it is 1-bounded. For example, the nets shown in Figs.2.11 are all bounded; in particular, the net in Fig.2.11 (b) is 2-bounded, and the rest of the nets are safe. Places in a Petri net are often used to represent buffers and registers for storing intermediate data. By verifying that the net is bounded or safe, it is guaranteed that there will be no overflows in the buffers or registers, no matter what firing sequence is taken.

- Livenes: liveness is closely related to the complete absence of deadlocks in operating systems. A Petri net (N, M_0) is said to be live (or equivalently M_0 is said to be a live marking for N) if, no matter what marking has been reached from (M_0), it is possible to ultimately fire any transition of the net by progressing through Some

further firing sequence* This means that a live Petri net guarantees deadlock-free operation, no matter what firing sequence is chosen.

Liveness is an ideal property for many systems. However, it is impractical and too costly to verify this strong property for some systems such as the operating system of a large computer. Thus, we relax the liveness condition and define different levels of liveness as follows. A transition t in a Petri net (N, M_0) is said to be:

1. dead $L0-live$ if t can never be fired in any firing sequence in L(M_0).
2. $L1-live$ (potentially firable) if t can be fired at least once in some firing sequence in $L(M_0)$.
3. $L2-live$ if, given any positive integer k, t can be fired at least k times in some firing sequence in $L(M_0)$.
4. $L3-live$ if t appears infinitely, often in some firing sequence in $L(M_0)$.
5. $L4-live$ or live if t is $L1-live$ for every marking M in $R(M_0)$.

A Petri net (N, M_0) is said to be $Lk-live$ if every transition in the net is $Lk-live$, $k = 0, 1, 2, 3, 4. L4-liveness$ is the strongest and corresponds to the liveness defined earlier.

- Reversibility and Home State: A Petri net (N, M_0) is said to be reversible if, for each marking M in $R(M_0)$, M_0 is reachable from M. Thus, in a reversible net one can always get back to the initial marking or state. In many applications, it is not necessary to get back to the initial state as long as one can get back to some(home)state. Therefore, we relax the reversibility condition and define a home state. A marking M' is said to be a home state if, for each marking M in $R(M_0)$, M' is reachable from M. Note that boundedness, liveness, and reversibility are independent of each other. For example, a reversible net can be live or not live and bounded or not bounded. Fig.2.12 shows examples of eight Petri nets for all possible combination of these three properties, where $\overline{\text{B}}$, $\overline{\text{L}}$, and $\overline{\text{R}}$ denote the negations of boundedness (B), liveness (L), and reversibility (R).

- Coverability: A marking M in a Petri net (N, M_0) is said to be coverable if there exists a marking M' in $R(M_0)$ such that $M'(p) \leq M(p)$ for each p in the net. Coverability is closely related to $L1-liveness$ (potential firability). Let M be the minimum marking needed to enable a transition t. Then t is dead (not $L1-live$) if and only if M is not coverable. That is, t is $L1-live$ if and only if M is coverable.

- Persistence: A Petri net (N, MO) is said to be persistent if, for any two enabled transitions, the firing of one transition will not disable the other. A transition in a persistent net, once it is enabled, will stay enabled until it fires. The notion of persistence is useful in the context of parallel program schemata and speed-independent asynchronous circuits. Persistency is closely related to conflict-free nets, and a safe persistent net can be transformed into a marked graph by duplicating some transitions and places. Note that all marked graphs are persistent, but not all persistent nets are marked graphs. For example, the net shown Fig.2.12(c) is persistent, but it is not a marked graph.

- Synchronic Distance: The notion of synchronic distances is a fundamental concept introduced by C. A. Petri. It is a metric closely related to a degree of mutual dependence between two events in a condition/event system. We define the synchronic distance between two transitions t_l and t_2 in a Petri net (N, MO) by

$$d_{1_2} = max|\overline{\sigma}(t_l) - \overline{\sigma}(t_2)|$$

  where σ is a firing sequence starting at any marking M in $R(Mo)$ and $\overline{\sigma}(t_i)$ is the number of times that transition t_i, $i = 1, 2$ fires in σ. For example, in the net shown in Fig. 2.12(d) $d_{1_2} = 1, d_{3_4}, = 1, d_{1_3} = \infty$, etc.

- Fairness: Many different notions of fairness have been proposed in the literature on Petri nets. We present here two basic fairness concepts: bounded-fairness and unconditional (global) fairness. Two transitions tl and $t2$ are said to be in a bounded-fair (or B-fair) relation if the maximum number of times that either one can fire while the other is not firing is bounded. A Petri net (N, MO) is said to be a B-fair net if every pair of transitions in the net are in a B-fair relation. A firing sequence

σ is said to be unconditionally (globally) fair if it is finite or every transition in the net appears infinitely often in σ. A Petri net (N, MO) is said to be an unconditionally fair net if every firing sequence σ from M in $R(M_0)$ is unconditionally fair. There are some relationships between these two types of fairness. For example, every B-fair net is an unconditionally-fair net and every bounded unconditionally-fair net is a B-fair net. The net shown in Fig.2.12(h) is a B-fair net as well as an unconditionally fair net. The net shown in Fig2.12(d) is neither a B-fair net nor an unconditionally fair net since t_3 and t_4, will not appear in an infinite firing sequence $\sigma = t_2t_1t_2t_1....$ The unbounded net is shown in Fig. 2.12(c) is an unconditionally fair net but not a B-fair net since there is no bound on the number of times that t_2 can fire without firing the others when the number of tokens in p_2 is unbounded.

"Structural properties are those that depend on the topological structures of Petri nets. They are independent of the initial marking MO in the sense that these properties hold for any initial marking or are concerned with the existence of certain firing sequences from some initial marking". These properties are as following [1]:

- Structural Liveness: A Petri net N is said to be structurally live if there exists a live initial marking for N.

- Controllability: A Petri net N is said to be completely controllable if any marking is reachable from any other marking.

- Conservativeness: A Petri net N is said to be (partially) conservative if there exists a positive integer y(p) for every (some) place p such that the weighted sum of tokens,$M^Ty = M_0^Ty =$ a constant, for every M ϵ, RM_0 and for any fixed initial marking M_0.

- Repetitiveness: A Petri net N is said to be (partially) repetitive if there exists a marking M_O and a firing sequence σ from M_O such that every (some) transition occurs infinitely often in σ.

- Consistency: A Petri net N is said to be (partially) consistent if there exists a

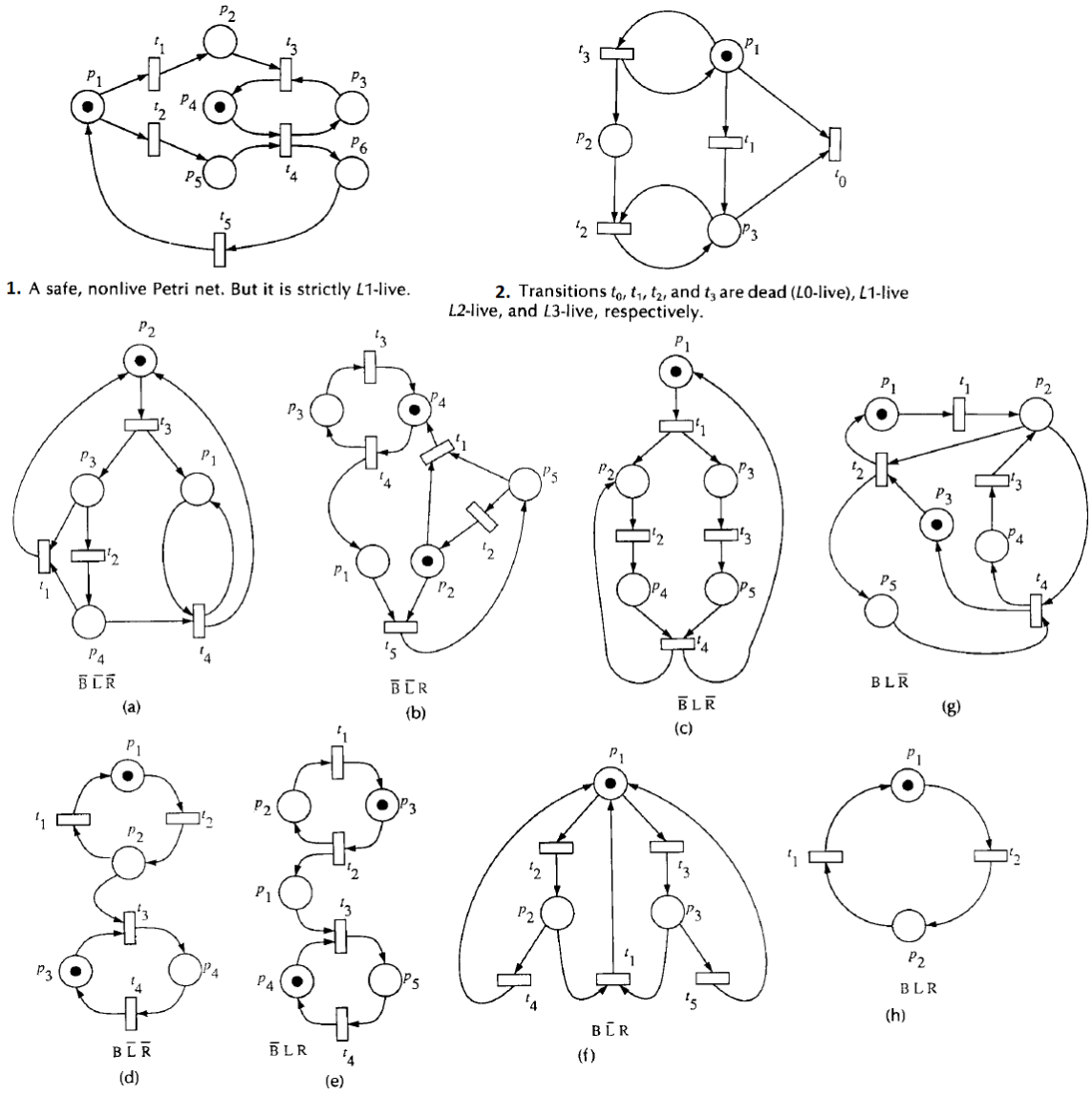


Figure 2.12: Example nets of CPN properties [1].

marking M_O and a firing sequence σ from M_O back to M_O, such that every(some) transition occurs at least once in σ .

- S- and T-invariants: An m-vector y (n-vector x) of integers is called an S-invariant (T-invariant) if $A_y = 0(A^T x = 0$.

- Structural B-Fairness: The concept of B-fairness discussed previously can be extended to the following structural properties. Two transitions are said to be in a structural B-fair relation if they are in a B-fair relation for any initial marking. A Petri net is said to be structurally B-fair if it is a B-fair net for any initial marking.

2.7.4 Verifying protocols using CPN Tool

The tool features incremental syntax checking and code generation, which take place while a net is being constructed. A fast simulator efficiently handles untimed and timed nets. Full and partial state spaces can be generated and analyzed, and a standard state-space report contains information, such as boundedness properties and liveness properties [33].

2.8 Summary

Based on the information provided in previous sections, it is clear that the existing security techniques are only suitable for architectures like MQTT systems which only contains three entities - publisher, subscriber, and a broker. Any more complicated architecture for IoT based systems cannot guarantee end-to-end security. Therefore, this thesis will describe in detail how to incorporate Two-Factor Authentication protocol into IoT based system architectures, and verify that this protocol indeed can be expressed and simulated using CPN.

Chapter 3

Two Factor Authentication Protocol for General IoT Architecture

In the previous chapter, the lack of security and privacy protection for existing IoT systems was explained and illustrated. The biggest issue is that current IoT systems can only provide a point-to-point secured communication channel using techniques such as TLS or DTLS. However, in order to satisfy the security and privacy requirements described in the previous chapter, an end-to-end secured system must be implemented.

3.1 TLS Quick Review

TLS 1.0 was first introduced back in 1999 and has been constantly evolving. TLS 1.3 was defined in RFC 8446 in August 2018 based on the specification for TLS 1.2. The purpose of this section is to give a quick review on the sequence of the TLS handshake protocol, which is a fundamental building block for the Two Factor Authentication protocol. This whole TLS handshaking sequence will be denoted as a single step called "TLS handshake process" in later sections for simplicity and cleaner illustration purposes [37].

Client-server applications use the TLS protocol to communicate across a network in a way that designed to prevent eavesdropping and tampering by creating a secured communication channel between the client and server. Applications generally will indicate whether they want to communicate with or without TLS before they establish the communication channel. It is necessary for the client to tell the server that a TLS connection is required. An alternative mechanism is that the client need to make a protocol-specific request to the server to switch the connection to TLS by making a STARTTLS request when using the mail and news protocols [37].

"Once the client and server have agreed to use TLS, they negotiate a stateful connection by using a handshaking procedure. The protocols use a handshake with an asymmetric cipher to establish not only cipher settings but also a session-specific shared key with which further communication is encrypted using a symmetric cipher. During this handshake, the client and server agree on various parameters used to establish the connection's security "[37] [46]:

1. The handshake begins when a client connects to a TLS-enabled server requesting a secure connection and the client presents a list of supported cipher suites (ciphers and hash functions).

2. From this list, the server picks a cipher and hash function that it also supports and notifies the client of the decision.

3. The server usually then provides identification in the form of a digital certificate. The certificate contains the server name, the trusted certificate authority (CA) that

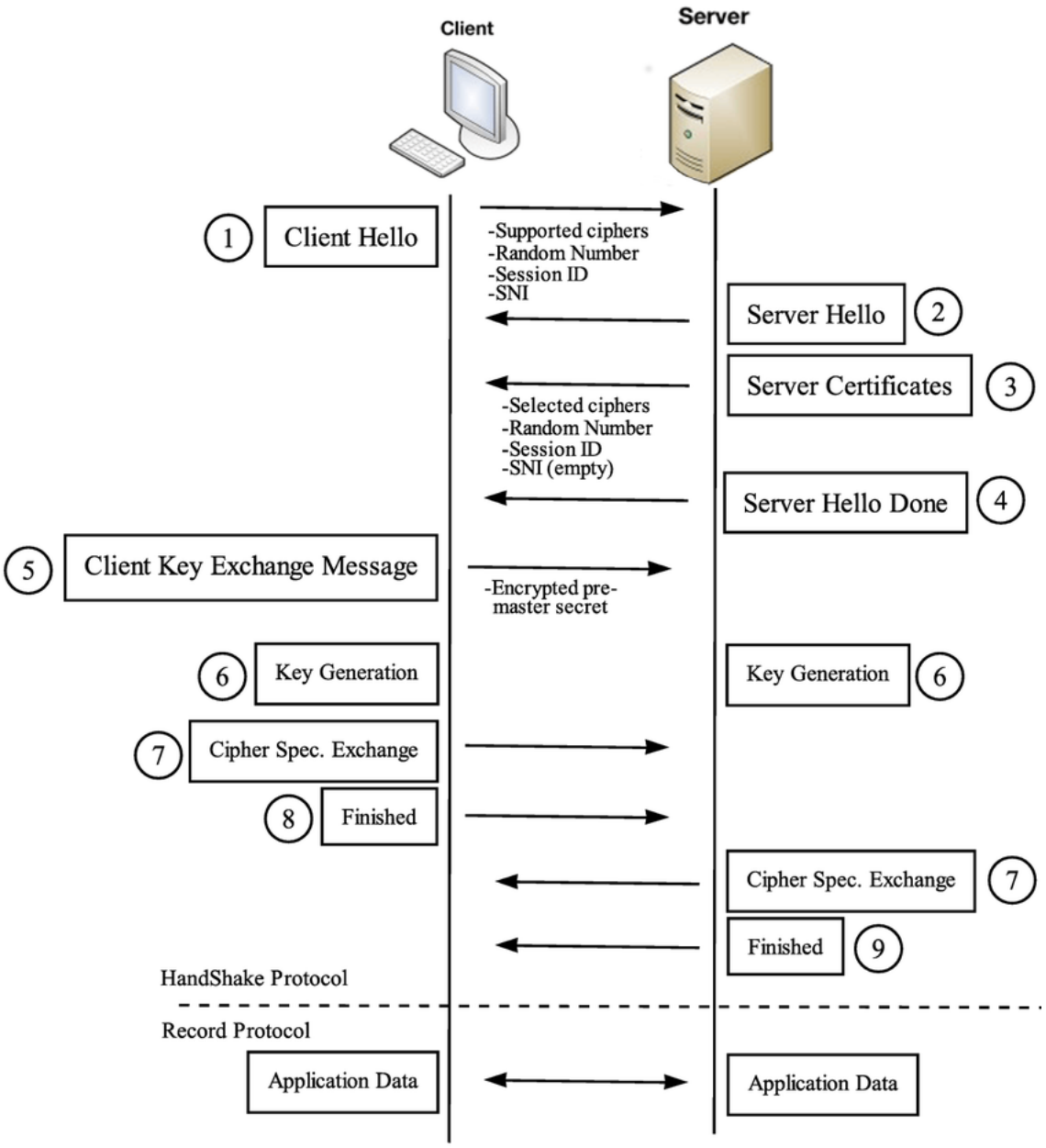


Figure 3.1: TLS handshake protocol [25]

vouches for the authenticity of the certificate, and the server's public encryption key.

4. The client confirms the validity of the certificate before proceeding.
5. To generate the session keys used for the secure connection, the client either:
    - encrypts a random number with the server's public key and sends the result to the server (which only the server should be able to decrypt with its private key); both parties then use the random number to generate a unique session key for subsequent encryption and decryption of data during the session
    - uses Diffie–Hellman key exchange to securely generate a random and unique session key for encryption and decryption that has the additional property of forward secrecy: if the server's private key is disclosed in the future, it cannot be used to decrypt the current session, even if the session is intercepted and recorded by a third party.

The procedure described above concludes the TLS handshake process and has created a secured communication channel, which is encrypted and decrypted with the session key until the connection closes. If any of the above steps fails, then the TLS handshake fails and the connection is not created [37]. In Figure 3.1, it illustrates the complete TLS handshake procedure, which is denoted as a single round trip process "TLS handshake process".

3.2 Two Factor Authentication Protocol Architecture

Existing IoT systems have already adopted some of the most advanced techniques such as TLS or DTLS for establishing a point-to-point secured communication channel, however it is not sufficient to make the whole system end-to-end secure. To achieve end-to-end security for the whole system, a more sophisticated system infrastructure needs to be developed in order to achieve this objective.

Since there is very little room to make some progress on the point-to-point secured communication channel, the focus for achieving end-to-end security will be eyed on the

system architecture. In the other words, the new infrastructure needs to build on top of the existing point-to-point secured communication channels using them as a safety blanket, but leverage the existing point-to-point secured communication channel infrastructure to achieve the goal of end-to-end secured. The building block for achieving end-to-end security is to ensure each entity within the system is legitimate. To do so, we can make uses of Two-Factor Authentication (TFA) methods. TFA is an authentication process of confirming a user's identity. The user's access privilege is granted access only after successfully presenting at least two pieces of evidence (or factors) to an authentication mechanism: "knowledge (something they and only they know), possession (something they and only they have), and inherence (something they and only they are)" [47]. The authentication mechanism used should originate from two different networks or channels in order to fully utilize the protection granted by TFA. TFA authentication significantly reduces and prevents many kinds of fraud and hacking due to the double or even multiple layers of protection, and guard against attackers that may only have access to one of the channels. Although the authentication mechanism used in each channel can be just the commonly accepted methods, with the addition of the second layer of authentication layer, the probability that both authentication channels been successfully breached will be reduced multiplicatively in comparison with just single channel authentication. Two-Factor Authentication can effectively block many of the most common kinds of hacking and identity theft in online banking, which has been adopted in many real-world applications.

In Figure 3.2, it illustrates the authentication process for existing an IoT system using MQTT as the messaging protocol from system architectural point of view, where both Things and Users need to authenticate into the MQTT broker as the first step, then they can pub/sub messages through brokers or upload/download data to/from the server through the connection channel provided by the Broker; furthermore, the Users can obtain the access privilege to the things in order to directly access the data on things. One thing to notice is that only the Administrator of the system can directly access the Server, hence have the control on the entire system, any other parties can only access the Server via the Broker.

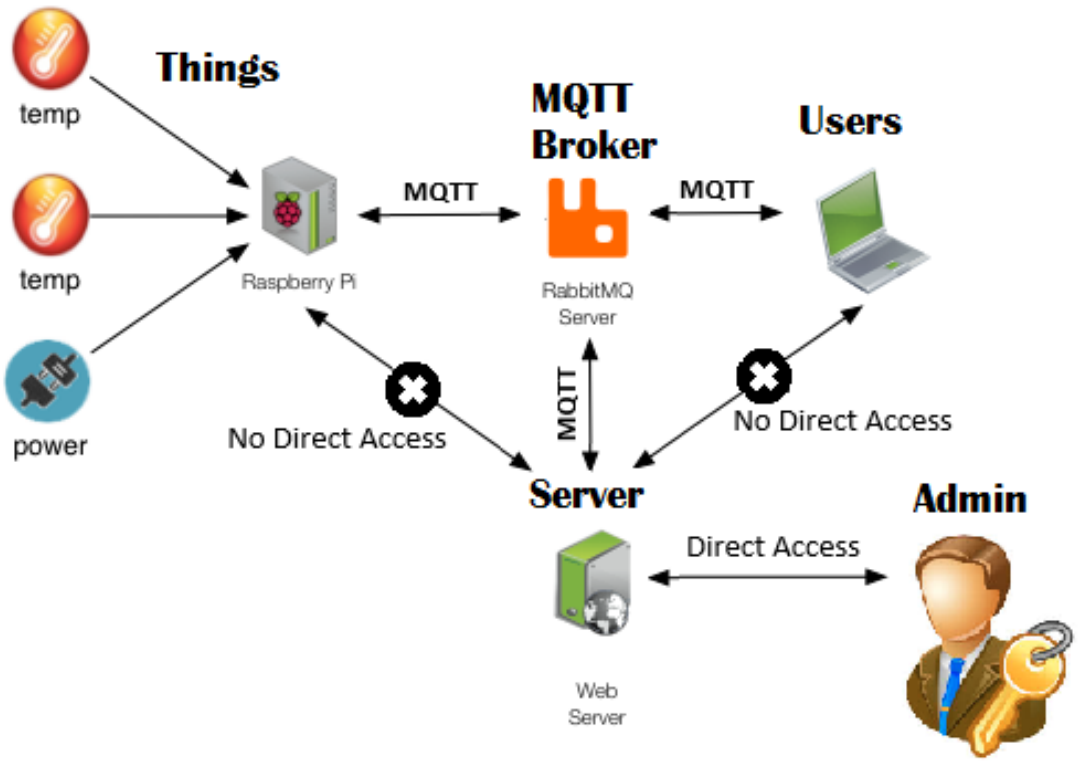


Figure 3.2: Authentication process for existing IoT system

In Figure 3.3, TFA has been adopted into the IoT system. In this architecture, both Things and Users can now create a communication channel to the Server as their secondary authentication channel for the system. Both Things and Users now have to authenticate into both the Broker and the Server before they can obtain their access privilege on the system. Again, only the Administrator can directly access the Server without using the TFA protocol.

By deploying the TFA protocol into the existing IoT system, the probability for an attacker to successfully breach into the system becomes significantly lower in comparison with the single-factor authentication. As the result, any authenticated entity can be considered as a legitimate user or things, hence the whole system can be considered end-to-end secured. Now, if we examine the TFA IoT system against the security and privacy requirement for IoT system, it is safe to mark three more categories as been secured by TFA as illustrated in Figure 3.4.

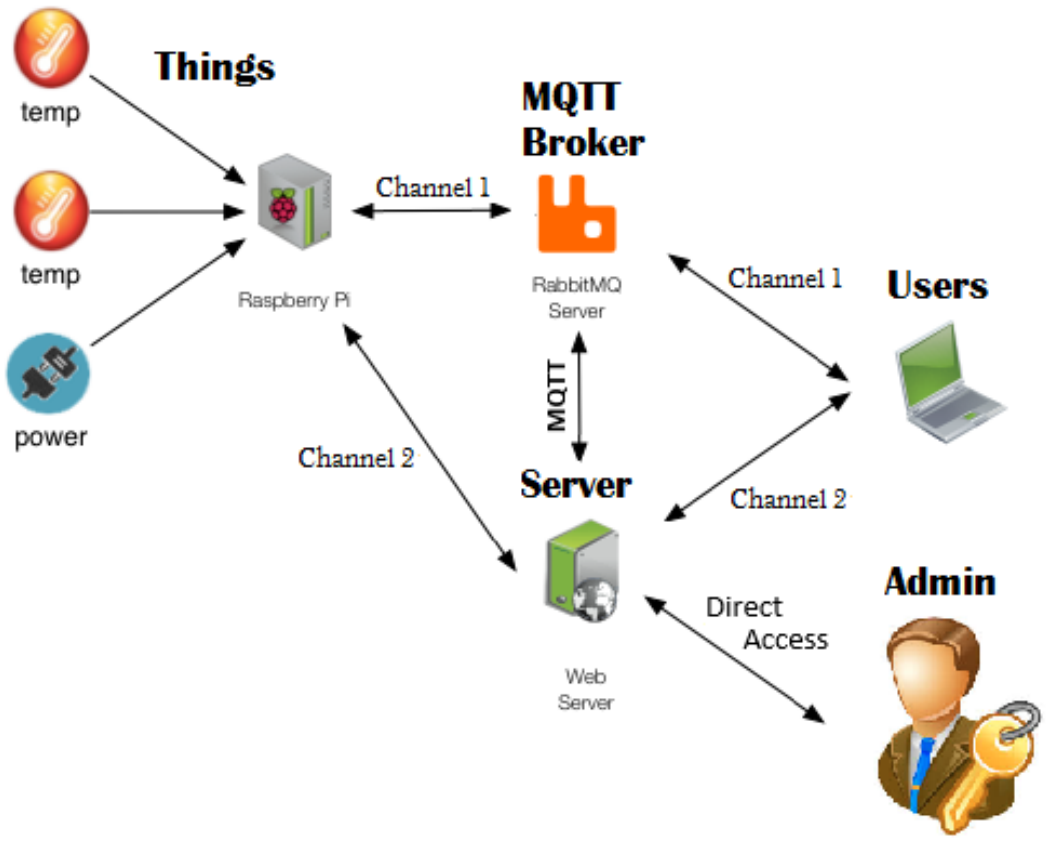


Figure 3.3: Two-Factor Authentication solution for IoT system

3.2.1 Two Factor Authentication Protocol without Broker Variation

The solution described above has taken into consideration of the messaging protocols that incorporate a broker in the architecture.

The solution has been illustrated using MQTT as the example since MQTT is chosen for the simulation and evaluation for this thesis. In the previous chapter, it has already explained that all of the commonly used IoT messaging protocol are client/server based using either publish/subscribe or request/receive models. All these protocols either do not have any built-in security mechanisms or they are making use of the mechanisms in other network layers such as TLS or DTLS. Hence, all the security mechanisms are still

Categories	is it secured	Issues
User privacy and data protection	No	Need to rely on properly regulated authorization and access controls provided by the system, otherwise anyone can potentially access any information that are at rest inside the MQTT broker. The devices themselves are also exposed to physical attacks as well, which leads to information stored the device been stolen as well.
Authentication and identity management	**Yes**	
Trust management and policy integration	**Yes**	
Authorization and access control	No	Need to fully rely on people to create and properly regulate these rules set in place. If there is a flaw in implementing these rules, sensitive information can be accessed by unauthorized users
End-to-End security	**Yes**	With out-of-band authentication in place, we can guarentee that every point within the system is trustable as the porability of breaking two channels at once is very very low

Figure 3.4: Security assessment with OOBA

on the point-to-point level. The TFA solution presented will be effective and usable in all of the other systems that uses different messaging protocols, which will enhance their system to an end-to-end secured level.

3.3 Two Factor Authentication Protocol with Certificate Authority - Further Enhancing Authorization and Access Controls

By incorporating Two Factor authentication protocol into the system, we can put check marks on Authentication and identity management, Trust management and policy integration, and end-to-end security in the security and privacy requirement as illustrated in Figure 3.4. The categories that are still left uncovered are user privacy and data protection, and authorization and access control.

These two categories are actually closely related to one another, where if an unintended user has gained access to information and data that are not supposed to be available to this user, then user privacy and data protection is lost in this case. Taking an airline company as an example, where the mechanics for the airplane can remotely view the engine turbine and other sensors' information on the plane, but all the other flight's information such as number of passengers should not be accessible by the airplane mechanics, and should only be available to airline's management team.

To overcome the flaws that still exist in the system, we have proposed a solution that utilizes the Certificate Authority (CA) to manage user access related issues. The main purpose of adopting CA is that the digital signature provided by CA can further improve the point-to-point security for a secured channel communication even if there is a man-in-the-middle, the attacker will not be able to decrypt the information they intercept since they do not have the private key to decrypt the messages. On top of this, a user's access permission can be embedded within the Authorization certificate so that each end device can only access its permitted resources or information, hence solve the authorization and access control issues within the existing IoT systems. Overall, user privacy and data protection requirements are met as well in this case.

3.3.1 Solution without CA

In the case where CA is not available for the system implementation or on the resource constraint device where the digital signature is not feasible, a replacement solution is required.

In Patent[6], it described a solution that incorporates a micro-controller that provides the functionality for securing information exchanged between two calling parties. Essentially, the micro-controller replaces the functionality of the Central Authority and is able to work with resource constrained devices or environments.

Article [27] summarizes all the key components, and it will not be discussed in detail here.

Categories	is it secured	Issues
User privacy and data protection	**Yes**	Need to rely on properly regulated authorization and access controls provided by the system, otherwise anyone can potentially access any information that are at rest inside the MQTT broker. The devices themselves are also exposed to physical attacks as well, which leads to information stored the device been stolen as well.
Authentication and identity management	**Yes**	
Trust management and policy integration	**Yes**	
Authorization and access control	**Yes**	Need to fully rely on people to create and properly regulate these rules set in place. If there is a flaw in implementing these rules, sensitive information can be accessed by unauthorized users
End-to-End security	**Yes**	With out-of-band authentication in place, we can guarentee that every point within the system is trustable as the porability of breaking two channels at once is very very low

Figure 3.5: Security assessment after incorporating CA

3.4 Two Factor Authentication Protocol - Step by Step

The ultimate goal of using TFA in the IoT architecture is to create an end-to-end secured IoT system. Therefore any unknown devices and users are not permitted to access any information in the system by default. Each end device (including things and end-users) should by default have the public keys of the Server and the Broker, and the unique ID of every devices and users (unique identifiers for devices and login credentials for users) will need to be stored in both the Server and the Broker initially. Since things here are not like intelligent human users, the unique identifier used in the authentication process will vary, but the steps for the protocol should remain almost identical.

Before diving into the Two Factor Authentication protocol, the existing authentication protocol for IoT systems can be visualized as simple as shown in Figure 3.6, which only consists of the TLS handshake protocol shown in Figure 3.1. However, this TLS handshake process is crucial to ensure the point-to-point communication is secured. All

the messages that have been exchanged within the TFA protocol are running under the encapsulation of the TLS, effectively using TLS as the safety blanket to defend against any intrusions. The TFA protocol is essentially building on top of the existing TLS protocol, but put an extra layer of the two factor authentication.

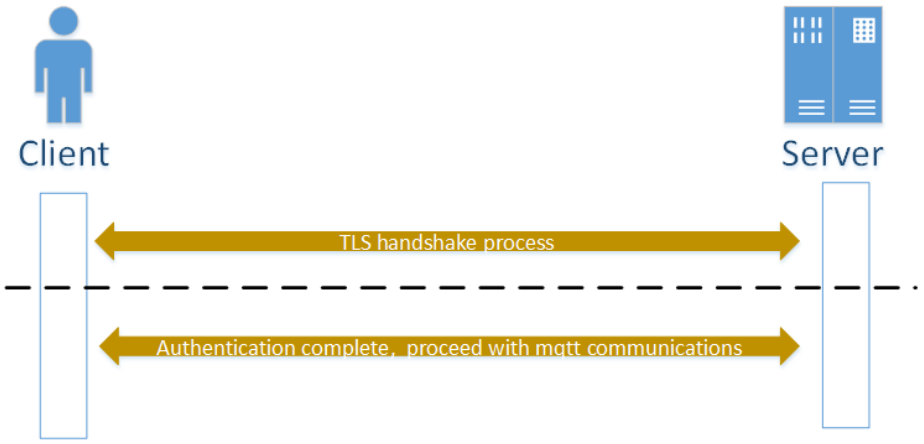


Figure 3.6: Existing TLS authentication protocol for IoT system

Since things are not like intelligent human users, and in most of the use cases its purpose is to upload data or provide information for other parties; hence access control will not be discussed here, or we can treat the things been having the minimal access rights. In the case where access control is a concern for the things, the technique that is designed for Users can be used.

From the protocol description point of view, the authentication procedure for both things and users will be the same, the difference lies within the information been exchanged in each step. To differentiate between the protocol used for things and users, the following two sections will be discussing the detailed steps used in the protocol.

3.4.1 Authentication Procedures for Things

The following steps are the exact procedure will be used in TFA for establishing a secured channel for both the Server and the Broker. The design philosophy for users will include

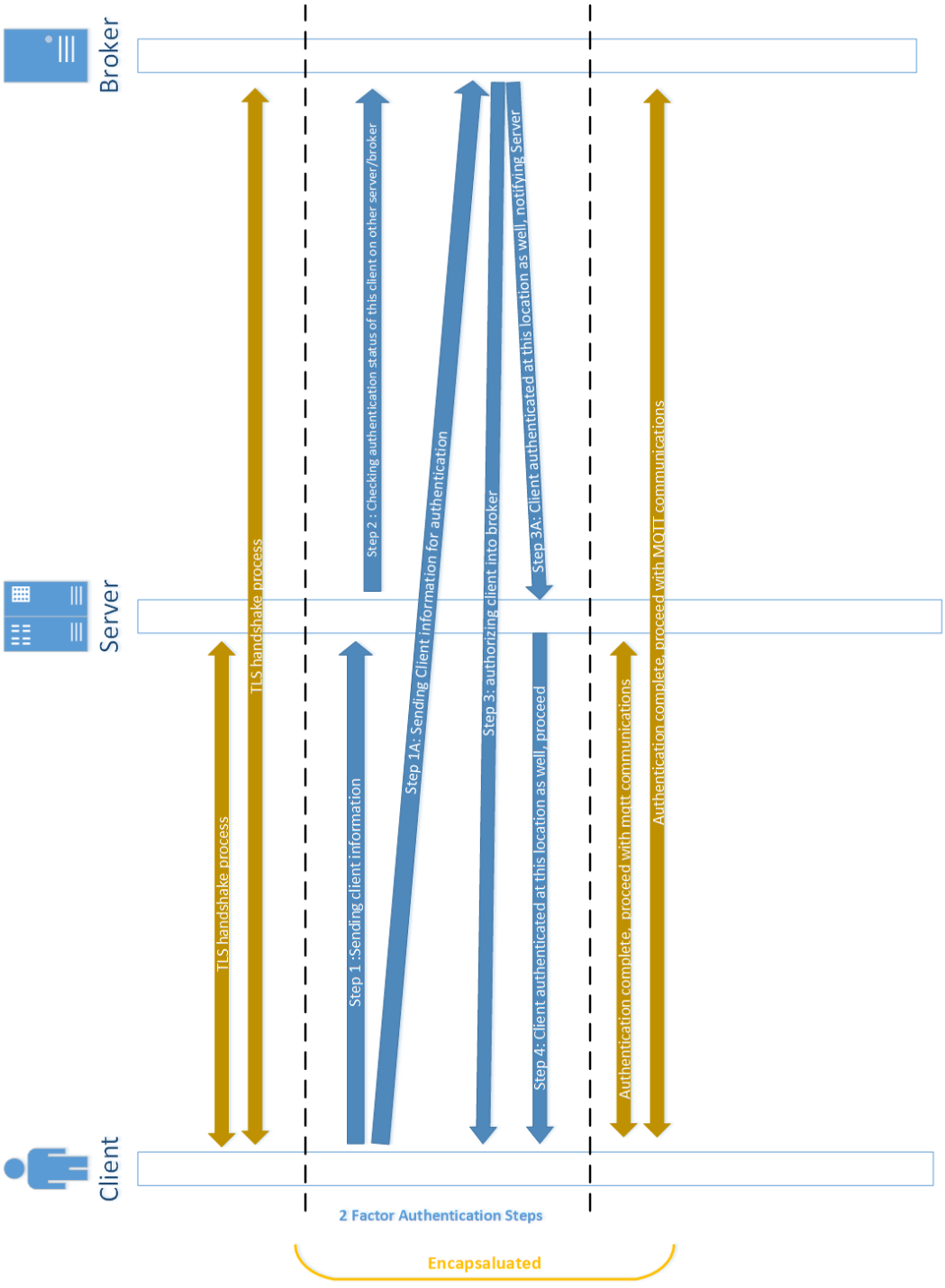


Figure 3.7: Two-factor Authentication solution for IoT system

easy access from multiple locations or devices including the user's cell phone, PC or other devices depending on the user's convenience. Hence sometimes it is hard to give a single unique ID for one specific user; in comparison, things can have a unique ID hardcoded to that specific device. This is where techniques like username and password authentication come in handy. For users, we will also take access control into consideration, hence the procedure will differ in certain steps compare with the procedure for Things:

1. A Thing initiates the connection to the server by sending a request. Within this initial handshake request, the Thing must uses the public key of the server to encrypt this request. The request needs to include the unique ID of the Thing and the public key for the Thing (if the public key of the Things is not already available to the server by default). With this unique ID, the server can uniquely identify this Thing.

   In this step, we have the option to have the CA signs this request before sending it to the Server. By doing this, it will further secure the authentication procedure, but it is not mandatory.

2. When the Server received this initial handshake request, it will first check with the CA for the signature that comes from the request and verifies the identity of the Thing in the case when CA is used in step one. Then the server decrypts the request using the server's private key and verifies if the ID within the request matches with one of the available keys stored in the Server.

3. Once the ID is verified, the Server will need to communicate with the Broker and wait until this Thing is authenticated at both the Server and the Broker before granting access permission. The Server will notify the Broker that this Thing is already authenticated at the Server. Once the Thing is authenticated at the Broker, the Broker will also notify the Server that this Thing is authenticated at both sides. Then both Server and Broker can proceed to the next step.

   This step is mirrored in the case for the authentication procedure for the Broker. The goal is to satisfy the TFA explained in the previous section, and the communication between the Server and the Broker will ensure the TFA has satisfied.

Essentially, this is the first layer of protection, when we add a second channel for TFA, the probability that the system security been breached will be multiplicatively lower.

4. After the Broker has notified the Server that the Thing is authenticated at the Broker as well, the Server will generate a secret key that will be used for encrypting the communication channel between the Thing and the server. The Server will encrypt the secret key in the response using the Thing's public key that is sent to the Server in the request back in step one and have the CA sign the response before sending back to the Thing. The digital signature of the CA is to tell the thing that the origin of the response is legitimately from the Server. In this case, even if there is a man-in-the-middle attack, they can only intercept the message but not able to decrypt the messages as they are encrypted by the server's public key and the private key is not available to them.

5. When the Thing received the response from the Server, it can verify that the message is indeed from the server by verifying the signature of the response with the CA. Once the signature is verified, the handshake is complete at this point. Any further communication between the Server and the Thing will be using the secret key for encryption.

There are a few dependencies within the protocol Figure 3.7, meaning the next step cannot be executed unless the previous step has been completed, and these dependencies do not differ between things and users:

- Step 2 depends on Step 1
- Step 3 and Step 3A depend on Step 2 and Step 1A
- Step 4 depends on Step 3A

The authentication methods are not limited to the username/password. There are other methods such as soft tokens that can be used to replace or in combination with username and password as well. The method here can be flexible depending on the use-case and the level of security a system wants to achieve.

3.4.2 Authentication Procedures for Users

1. The User initiates the connection to the Server by sending a request. Within this initial handshake request, the User must use the public key of the Server to encrypt this request. The request needs to include the username and password of the User and the public key associated with the device that the User is using for authentication. Username/password are encrypted so they can only be decrypted by the Server, the Server can also verify the User's identity through the username/password.

   In this step, we have the option to have the CA signs the request before sending it to the server, and include a unique ID for the device that the User is using like the procedure described for Things. Having a unique ID sets a limit that the User can authenticate using certain devices. By doing this, it will further secure the authentication procedure, but it is not mandatory.

2. When the Server received this initial handshake request, it will first check with the CA for the signature on the request and verifies the identity of the User in the case when CA is used in step one. Then the Server decrypts the request using the Server's private key and verifies the username/password.

3. Once the username/password verified, the Server will need to communicate with the Broker and wait until this User is authenticated at both the Server and the Broker before granting access permission. The Server will notify the Broker that this Thing is already authenticated at the Server. Once the Thing is authenticated at the Broker, the Broker will also notify the Server that this Thing is authenticated at both sides. Then both the Server and the Broker can proceed to the next step. This step is mirrored in the case for the authentication procedure for the Broker.

4. After the Broker has notified the Server that the user is authenticated at the Broker as well, the Server will generate a secret key that will be used for encrypting the communication channel between the User and the Server. The Server will encrypt the Secret key in the response using the User's public key that is sent to the Server

in the request back in step one and has the CA sign the response before sending back to the Thing.

5. When the User received the response from the Server, it can verify that the message is indeed from the Server by verifying the signature of the response with the CA. Once the signature is verified, the handshake is complete at this point. Any further communication between the Server and the User will be using the secret key for encryption.

3.4.3 Out-of-Band Authentication Option - An Enhancement for Building Secured Channel

Out-of-Band Authentication (OoBA) is an enhancement for the TFA method described previously. The key concept behind OoBA is that it provides extra strength for each factor in the Two-factor or multi-factor authentication by using different communication channels. Putting this into context, we will take the architecture shown in Figure 3.3 as an example. The first communication channel would be the channel that users & things that are used to authenticate into the Server, and the communication media can be the internet. For the Out-of-Band communication channel or secondary factor authentication channel, it needs to avoid using the same type of media - the internet. Instead, it should use networks such as a cellular network to meet the requirement of OoBA. The advantage of this strategy is that it gives the system an even higher level of security since the attackers have to simultaneously break into two different networks, which makes the probability for the system been breached even lower. The disadvantage would be the cost of implementing such a system will be higher depending on the choice of communication channel. For example, the choice between 4G and 5G are for higher bandwidth and smaller delays. Depending on the requirement of the system, sometimes the upgrade from 4G to 5G are not worth the cost.

3.4.4 TFA on Other Messaging Protocols

Although the TFA protocol was introduced around MQTT as the messaging protocol, however the TFA protocol is not limited to systems that use MQTT as the messaging protocol. It is designed to be very generic such that it can be implemented to the systems that are using other messaging protocols such as AMQP or CoAP.

Essentially, the TFA protocol can be implemented on any system that requires some sort of authentication schema that lies in between the clients and the source of the messages. If the system requires a midpoint for authentication purposes, then the TFA protocol is applicable as all it requires is a secondary authentication mechanism that mimics the first one that is in place.

3.4.5 Exception Handling

The previous sections had described the sequence of execution of the TFA protocol and how each party within the protocol works on its own. However, things will not work out as smoothly as it was described in a real world scenario. There will be many unexpected scenarios such as delays between communications, lost of messages, or even system crashes during the authentication process. To ensure the authentication system can still be functional during the event of unexpected behaviours, exception handling and system state recovery process have to be implemented.

To better illustrate how exceptions are handled, the finite state machines for both Server/Brokers and Users/Things are presented in Figure 3.8 and Figure 3.9 respectively. The Server/Broker has seven different states, state 1A - 1B represents the states when the party is the first one that received the authentication request; on the other hand, 2A - 2B represents the states of the party after the second factor authentication:

1. **State 0** - represents the initial state of the Server/Broker. This will be the default state or the state that the system will be reset to if the Server/Broker system crashes. The event that associated will be either the system received the authentication request from the Client, which leads to state **1A**; or requesting authentication status of the Client from the secondary factor authentication, which leads to state

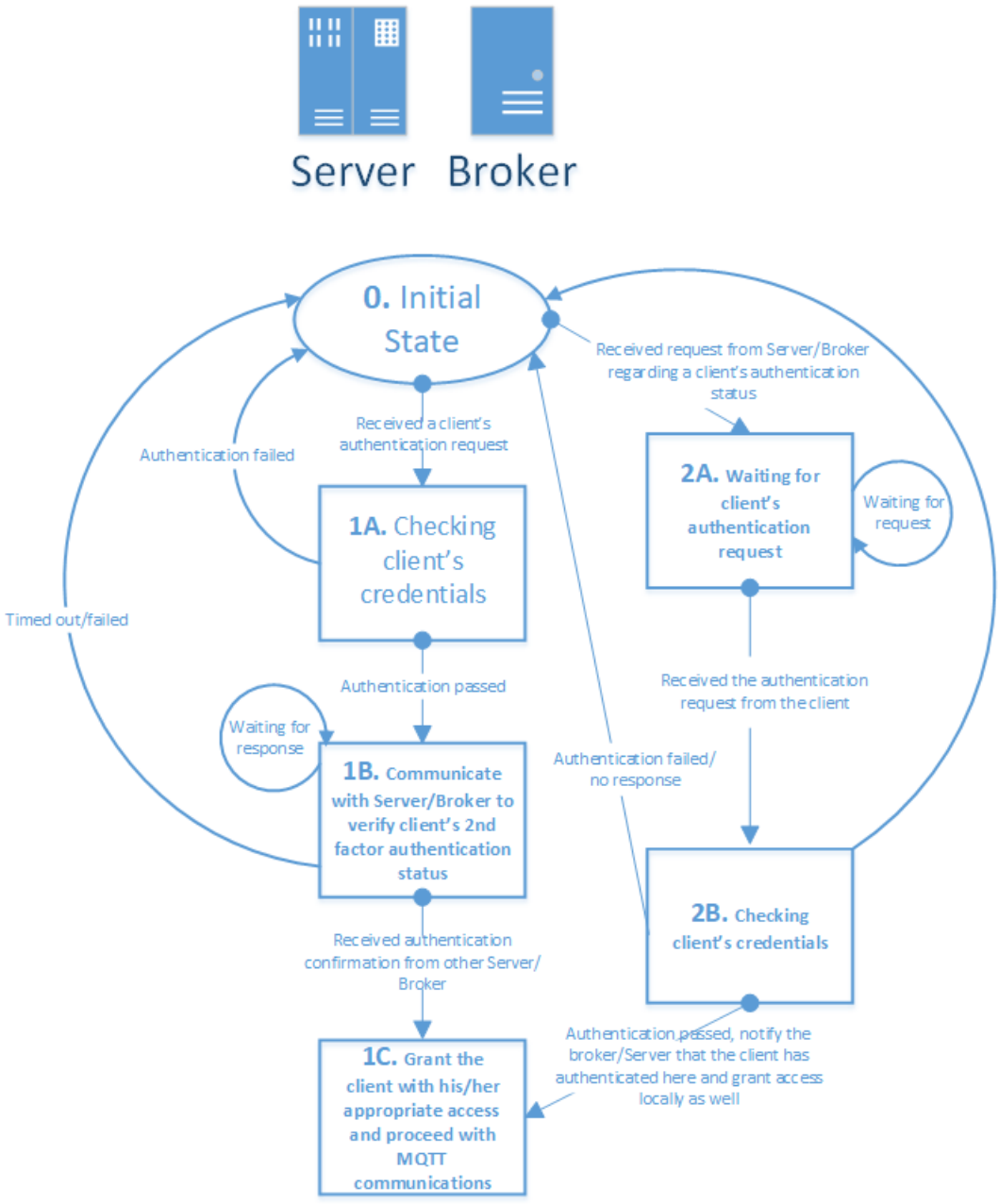


Figure 3.8: Finite state machine for Server/Broker

2A

2. **State 1A** - represents the state when the Server/Broker after the primary factor authentication. In the event when authentication credential is invalid or the system crashes, the Server/Broker will reset the state to **0**. In the event if the credential is valid, it will proceed to state **1B**

3. **State 1B** - represents the state when the Server/Broker is waiting for an authentication response from the second factor authentication. In the event when a time out period has reached, the Server/Broker will reset the state to **0**. If the authentication response is received, it will proceed to state **1C**;

4. **State 2A** - represents the state when the Server/Broker after the second factor authentication. In the event when it receives the authentication credential from the same client as the primary factor authentication, it will proceed to state **2B**; otherwise it will remain in the current state.

5. **State 2B** - If the credential from the same Client is valid, proceed to state **1C**; otherwise reset the state to **0**. There are some possible tweaks for this state, possibly make several attempts before the state is reset to accommodate possible human error

6. **State 1C** - the final state after authentication been accepted. The Server/Broker can now grant the access privilege to the Client

The Client has three different states:

1. **State 3A** - represents the initial state of the Client. This will be the default state or the state that the system will be reset to if the Client system crashes. The event that is associated with this state sends authentication information to the Server/Broker

2. **State 3B** - represents the state when the Client is waiting for the authentication response. In the event when authentication is rejected or after a time out period, the Client will reset the state to **3A**. This state involves two possible events, first,

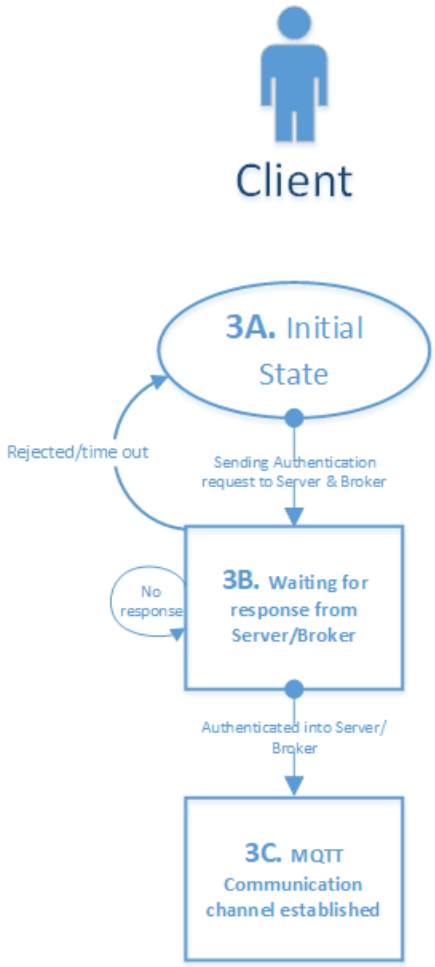


Figure 3.9: Finite state machine for Things

one been authentication accepted, which then the Client will proceed to **3C**; second, the event is waiting for a response, and the Client will remain in the current state until it either reaches the time out period or a response of either authentication has been accepted or rejected

3. **State 3C** - the final state after authentication been accepted. The Client can now proceed to pub/sub messages with the system using MQTT protocol

As shown in the finite state machines, during the states of waiting for a signal message or an acknowledgement message, time out has to be taken into consideration as a to receive such a message is possible.In comparison, other states only have to consider

things happening internally. In this case, these states need to be able to reset themselves so that the system will not get stuck in a deadlock.

3.5 Summary

Based on my research, there is not any generic solution that can completely fulfill the security and privacy requirements outlined in Figure 2.7. Some of the solutions that have been proposed are very application-specific and cannot work in other scenarios. By implementing this TFA protocol, the IoT system can achieve end-to-end system security, and it is not limited to MQTT as the messaging protocol.

Chapter 4

System Security Solution for Fleet Management IoT Architecture

In the previous chapter, a Two Factor Authentication Protocol was introduced. Since the protocol was designed to accommodate the general IoT architecture, therefore it should be able to deployed into variety of IoT systems. This chapter will discuss in detail on how this TFA protocol works for the Fleet Management IoT Architecture.

4.1 IoT Architecture for Fleet Management System

For the purpose of this thesis where the experiments and evaluations are carried out, an IoT architecture that is specifically designed for fleet management needs to be explained first.

The architecture for fleet management IoT system is an extended variation from the general IoT architecture. This fleet management architecture divides itself into 3 layers: perception, middleware and application layers, as shown in Figure 4.1. "The perception layer consists of various types of sensors and actuators allowing the IoT network to interconnect physical things that harvest information from the environment (sensing) and to interact with the physical world (actuation/command/control). It uses current Internet standards to provide a simple interaction between the physical world and the virtual world, by integrating a large number of things into the Internet" [30] [31].

This architecture uses the idea of fog to make the connections between the three layers. The term fog is a resource-constraint cloud near the edge of the network. The fog supports low latency and lightweight computing, and meets some IoT requirements that are not satisfied by the cloud. The fog lies between the middleware layer and the Perception layer, it provides the cloud with an interface to the lower IoT layers [15] [22].

Due to the specific use-case of the IoT system, the platform is divided into two planes: control plane and data plane. The control plane is used to manage all the vehicles and to implement Things to Things (T2T) communication between vehicles in the fleet management system. Data plane controls all the data flow from harvesting data all the way from vehicles to the local database, then towards the middleware layer, and finally to the application layer. The middleware layer receives fleet data in real-time and stored them in the distributed database nodes [30].

4.1.1 Control Plane

The Control Plane is implemented using the Meshcentral technologies [24], which is a Peer-To-Peer (P2P) technology supporting a mesh network architecture. This architecture is been picked over the client-server based architecture due to the potential bot-

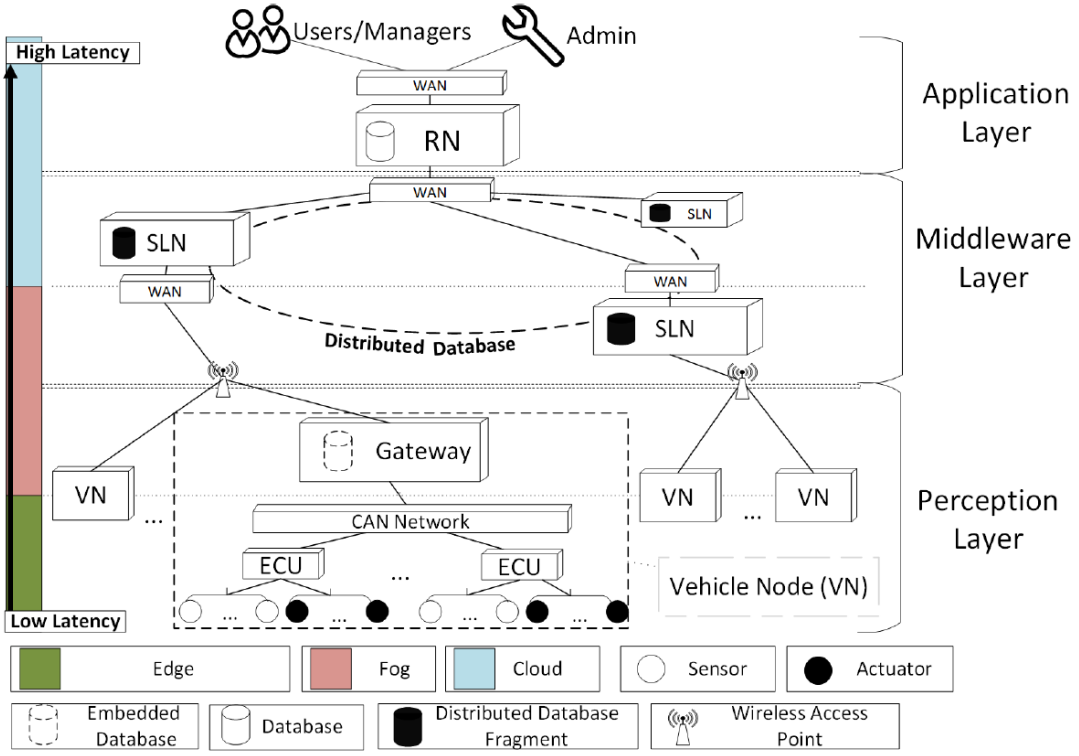


Figure 4.1: Fleet management IoT architecture.[31].

tleneck arises when a large number of things are involved in the T2T communication architecture. In this P2P architecture using Meshcentral, all participating nodes have equivalent capabilities and responsibilities. The nodes can directly exchange resources and services between each other without the need for centralized servers, which will greatly reduce the resource competition on the centralized server by spreading out the workload among different participating nodes. On the other hand, this P2P architecture using Meshcentral allows the fleet management system to connect large clusters of vehicles to a web service and implement T2T communication. In the proposed mesh network, the nodes are divided into three types: Root Node, Server Leader Node, and Vehicle Node. As shown in Figure 4.1, the Server Leader Node and Vehicle nodes are both referred to as mesh nodes in the context of this thesis. The three types of nodes are separately located in the application layer, middleware layer, and perception layer [30].

Meshcentral Technologies aims at giving web control and access to all Internet-enabled devices. In order to use Meshcentral Technologies, the mesh agent must be installed on all devices that want to be part of the management system. A symbol representing the device will show up on the Meshcentral web site and is immediately manageable once the agent is running on the device. These technologies are used for a wide ranging set of tasks such as remote terminal access, remote file access, remote monitoring of devices, remote power control, and remote device control. The set of technologies works as a client-server model, where the server is made of the Meshcentral and a myriad of clients which are mesh agents. The meshcentral structure is mainly divided into two parts: web service and mesh agent. The mesh agent runs in the background of each device that is part of the mesh network and generally has no user interface. Mesh agents communicate with other agents that are part of the same mesh network and to the web service. The web service runs on the Internet and serves as a master control for minute-by-minute operations [24].

We use the Meshcentral architecture for two purposes: One, providing a scalable way to connect servers and vehicles to the cloud; and Two, making mesh nodes discoverable, searchable and available even when they are powered off. For this thesis, the primary focus will be on the Control Plane as this is the plane where all the authentication procedures are carried out.

4.1.2 Data Plane

The main function of the data plane is to manage and store a large amount of data. First, local data is harvested from vehicles through sensors. SQLite [42] is used to temporarily store the local data. Second, all local data is sent periodically to a distributed database. The aggregation of local data from all vehicles is called fleet data. Third, the fleet data is sent to the information discovery layer to be analyzed and processed. Finally, the results of critical events are sent to the application layer. Operators can then act on the data based on the results received [30].

4.1.3 Different Nodes in the Architecture

This fleet management system has three types of nodes: Root Node, Server Leader Node (SLN) and Vehicle Node as shown in Figure 4.2 [30].

1 Root Node: The Root node will host the application layer, providing one central point of administration for the entire fleet system. It will need a redundant crash recovery mirror in the event the root node fails, since all the fleet will rely on this node for any initial information discovery when first connected into the system. The Root node will bridge the application handle to the server leader nodes' vehicle networks. To speak to the SLNs the Root Node will multiplex all the SLN MQTT networks into a connection pool. From an implementation point of view, an application will only need to communicate with the Root Node through the application layer's interface, the Root Node's MQTT broker.

   This offers a way for mesh agents to receive live commands from the mesh administrator and delegate administrators. The Meshcentral web service can be used to support many different usages for cloud connecting and managing mesh nodes. The Meshcentral web service provides many benefits:

   – It is a central synchronization point for all mesh nodes and acts as the root of the whole architecture.

   – It is also attached to a large database and can keep historical data about the mesh network, since, left alone, mesh agents keep track of each other, but don't keep any past data.

   – It offers a way for mesh agents to receive live commands from the mesh administrator and delegate administrators.

2 Server Leader Node: Mesh nodes can periodically synchronize and report state information to the web service, this is sufficient for passive monitoring of mesh nodes. However, a mesh cluster leader is needed to keep a permanent connection to the web service on behalf of all of the mesh nodes on the same physical network.

For example, we may divide individual countries into different regions, with each region forming a cluster. We call such a mesh cluster leader Server Leader Node. The implementation of this node in the system allows us to decentralize the system into different regions. In the case where data transferring is immense, using multiple Server Leader Node will greatly reduce the workload on the Root Node.

3 Vehicle Node: In the mesh network, there are a larger number of vehicles where each vehicle is representing a vehicle node. The Vehicle will first communicate with the Root node to discover who is the SLN that is in charge of its area. Then the Vehicle Nodes can report to their Server Leader Nodes regard to their states and the state of their vehicles. We adopt 4G LTE to connect Vehicle Nodes and Server Leader Nodes. Each mesh node keeps as many signed blocks of data as it can.

The relationship between different nodes are showcased in Figure 4.2. This architecture is highly scalable in order to meet the geographical distribution requirement, in which the architecture is designed in a hierarchical manner. As a fleet grows, additional SLNs can be deployed in a geographical region to partition the region's VNs into smaller sub-fleets to balance the workload in a specific region by adopting one or more SLNs [31].

4.2 Two-Factor Authentication Protocol for the Fleet Management Architecture

Having introduced the fleet management architecture, now it is the time to explain how the TFA works in this specific architecture. The biggest deviation of the fleet management architecture from the general one is that the Root Node will serve as a central agent that redirects each device or vehicle in this context, to its local server leader node (SLN). All the actual MQTT message transmission is done between each vehicle and the respective SLN. The Root node will only directly communicate with the devices or vehicles during the initial authentication process, effectively working as the central authentication guardian. Once the device is authenticated at the Root node using

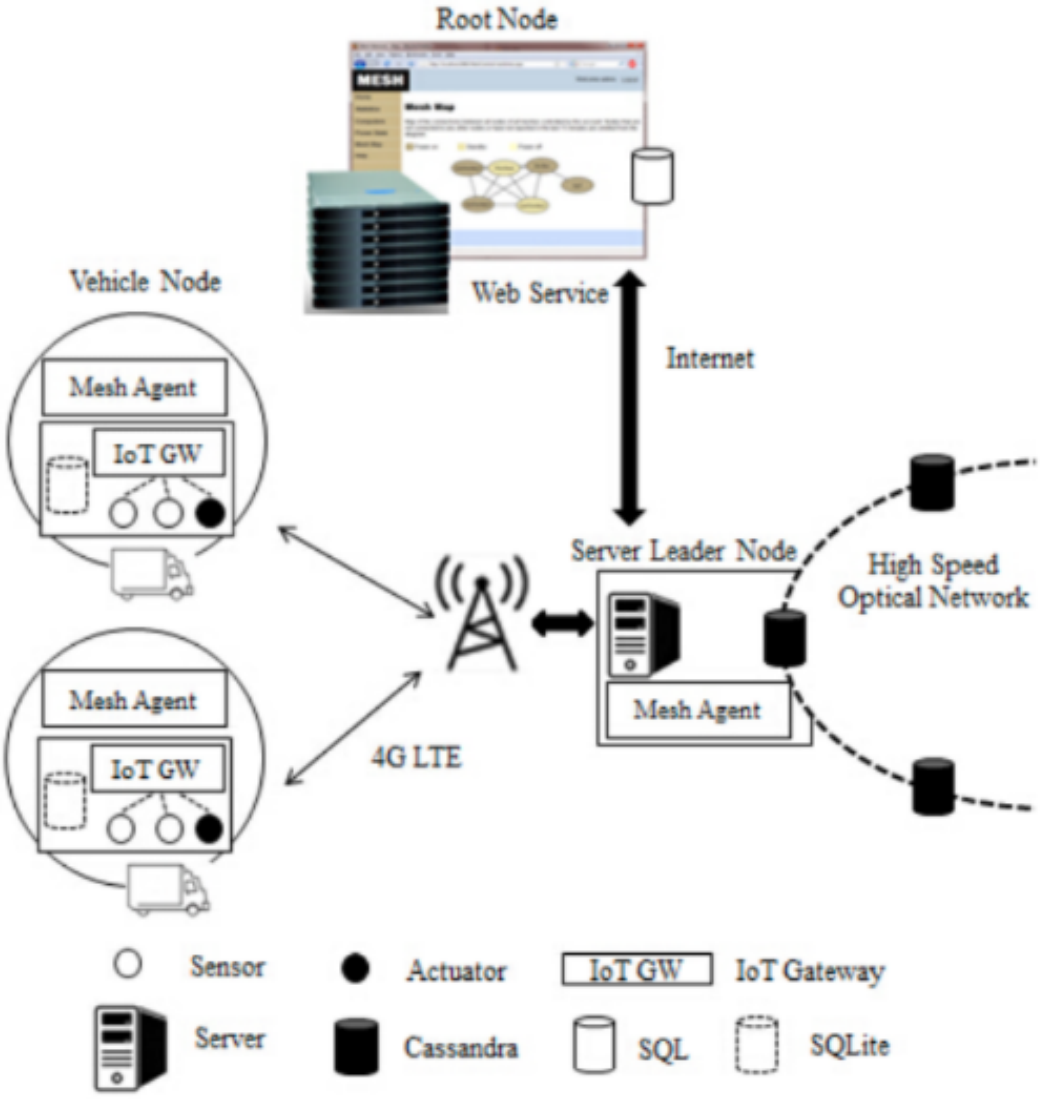


Figure 4.2: Relationship between different nodes in Fleet management IoT system [30].

TFA, then the Root node will generate an unique session ID that is only known to this vehicle and its respective SLN. The Root node will send the unique session ID and the address of the SLN to the vehicle, and only the unique session ID of this vehicle to the SLN. After this step, there will be no direct communication between the Root node and the vehicle unless the vehicle is disconnected from the system and requires a new unique session ID. In comparison, there will be communication between the Root node and the SLN from time to time depending on the information that needs to be shared between these two servers, this is also depending on how the system is designed based on the

specific requirements. Once the vehicle and the SLN all have the unique session ID, the vehicle will initiate another TLS handshake with the SLN using the unique session ID as the identifier; The SLN, on the other hand, will only accept connections with unique session ID as the identifier, any request without a unique session ID will be rejected by the SLN. The unique session ID in this case, is a generic term that can be varied based on the system requirement. It can be in the form of a security token or unique session key that the client can use to authenticate into the SLN. This security token can be a public key pair or just a secret key that will be used between the client and SLN. Essentially, providing this token eliminates the process for the client to do the same authentication process for the SLN again. Once the unique session ID is verified on the SLN, then the vehicle and the SLN can start using MQTT to pub/sub messages.

Doing a quick comparison between Figure 4.3 and Figure 3.7, before step 5, the authentication procedure is identical between the two figures. Step 5, Step 5A, and Step 6 are the three extra steps in the authentication protocol for the fleet management system. These steps represent the communication and authentication between the vehicle and the SLN. After step 4, which is when the client has authenticated into the Root node,the Root node will generate a unique session ID the client and the respective SLN. Then the server will send the unique session ID and the IP address of the SLN to the vehicle in Step 5, and the unique session ID to the SLN in Step 5A. Then the vehicle will try to establish the connection with the SLN using the unique session ID in Step 6. Once the TLS handshake completes between the client and the SLN, the client can start to transmit data to the SLN via MQTT or other communication protocols introduced earlier indicated in yellow arrow below Step 6.

The dependencies for the fleet management protocol has a few extra dependencies on top of the regular TFA protocol

- Step 2 depends on Step 1
- Step 3 and Step 3A depend on Step 2 and Step 1A
- Step 4 depends on Step 3A
- Step 5 and Step 5A depend on Step 3A - Additional

- Step 6 depends on Step 5 and Step 5A - Additional

4.3 Security and Privacy Related Issues with Fleet Management Architecture

The main difference between the Fleet management architecture and the general IoT architecture is the introduction of the Server Leader Node. Having the Server Leader Node means that there will be the extra steps for the client to authenticate into the Server Leader Node, where the Root node provides a security token to the client for the authentication process. In this scenario, the communication between the Root node and Server Leader Node been discussed is when the Root node sends the unique session ID to the Server Leader Node, we will assume that they have adopted a secured communication scheme to keep each other updated on the information about the security token been used. In reality, the communication channel between the Server Leader node and Root node plays a very important in measuring the system's end-to-end security as they are part of the system as well. The communication channel between these two nodes should use mechanisms such as a private key for identification purposes, the Server Leader node only contacts Root node when there is local information that needs to be transferred over to the Root node. On the other hand, the Server Leader node should reject all communication requests from clients unless an accepted security token such as a private key is present.

4.4 Summary

One thing that needs to take into consideration is that the scenario presented in the previous section is a very simple and generic case. In any real-world application, the communication paths will be much more complicated and frequent. For example, the Root node will need to keep track of all the Server Leader nodes' security token updated-to-date. To enhance the security strength of the communication channel for the Root

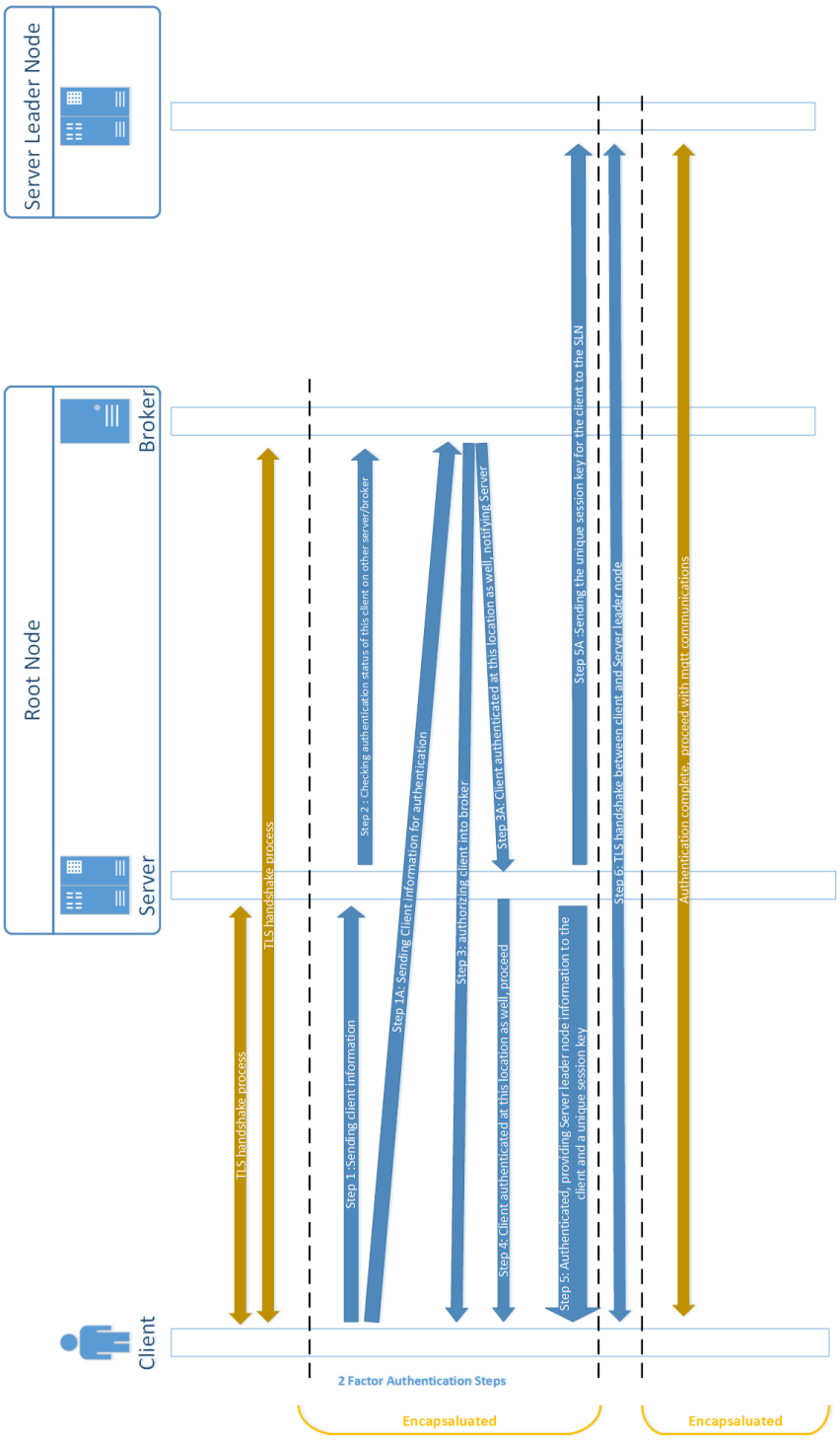


Figure 4.3: Two-Factor-Authentication sequence for Fleet management IoT system

node and Server Leader node, we can also adopt the TFA authentication protocol for their communication channel based on the system requirement. The drawback of using TFA for the Root node and Server Leader node is the increases in complexity of the system, and if the security token is stolen due to a physical attack, then the server leader node will be vulnerable to attacks. These will have to be more carefully considered as the physical attack is a very hard problem to prevent from the system architect point of view.

Chapter 5

Experiments and Evaluations

In this chapter, a detailed explanation of the system architecture that was used to simulate the Two-Factor-Authentication protocol that was described in previous chapters will be given in section 5.1. In section 5.2, a detailed analysis, inspections, and tests that have been conducted on the system will be presented and explained. Lastly, the evaluation on the work that has been done will be presented in terms of whether it is a good representation of the Two Factor Authentication solution and whether the results provided can prove that this solution can make an IoT system end-to-end secure.

5.1 Out-of-Band Authentication Implementation

5.1.1 Pre-implementation Work

Before we dive into the actual implementation of the TFA protocol, several preparation tasks need to be completed. The following is a list of essential software and tools that were used to develop and test the system, some additional items might be neglected:

- Mosquitto message broker: Since MQTT has been selected as the messaging protocol for testing and evaluation purposes, the message broker that will be used in the system will be Mosquitto since it is an open-source message broker that implements the MQTT protocol.
- Ubuntu 14.02 Operating System: All of the coding and testing will be conducted in one or multiple virtual machines of this OS. For tests that require more than IP addresses, additional virtual machines with identical environment setup will be deployed
- Python 2.7.2: Major part of the coding for this system will be done in Python 2.7.2
- Ubuntu built-in certificate generation tools: Since we need a Central Authority to test the solution, the certificate generation fulfilled this purpose by allowing the local machine to be the Central Authority by sign the server and client certificates.
- VMware Player: used for running Ubuntu operating systems. In the case when multiple IP addresses are required for testing purposes, multiple instances of VMware can be launched to serve such function
- mosquitto_pub and mosquitto_sub tools: mainly used for testing purposes that verifies the message broker can publish and subscribe correctly when the coded client did not function as expected
- Wireshark: gives the user the capability to listen and capture packets on a specific IP address or a specific port. This is the tool that can verify the solution satisfies the requirement for end-to-end security

- CPN Tools: A tool for editing, simulating, and analyzing Colored Petri nets, the purpose of using this tool is to verify the correctness of the protocol [33].

Given the list of items that are essential to the implementation of the system, the following list contains some core tasks that need to be completed before coding the different components of the system:

1. Set up Ubuntu 14.02 on VMware Player: Ubuntu 14.02 OS image was obtained from [38], after lanching up the Ubuntu image, python 2.7.2 was installed using the apt-get command from Ubuntu OS

2. The procedure on how to install Mosquito message broker and mosquitto_pub and mosquitto_sub was provided from [40]

3. In order to create our own Central Authority, the procedures of how to create our own certificates, how to setup Mosquito with TLS enabled, and how to verify that TLS is set up correctly are all provided in [43], this source also provides the command to test and validate that the Mosquito message broker can correctly allow clients to publish and subscribe to specific topics with TLS option enabled.

5.1.2 Implementation

After completing all the environment setup and software configuration tasks, we can now start the implementation of the system. The operation of the system revolves around three major components which were explained in Section 3.3: Server, Broker, and Client. The purpose of implementing this prototype is to demonstrate that the TFA protocol is indeed end-to-end secure, therefore these three components are implemented specifically to demonstrate such purpose. As a result, each component is not implemented exactly according to all of the TFA protocol standards and requirements, but coded to showcase the TFA protocol:

- Server: For demonstration purposes, the server will always be referred to as the first entity that receives the authentication request from the client. However, in reality,

either the Server or the Broker can be the first party to receive the authentication request, then the other party will act as the second factor of authentication.

Once the server receives the login request from the Client, if the credentials from the client match with the record on the Server, the Server will then send a message to ask the Broker whether this client is also authenticated at the Broker side. This message contains a unique identifier of this client known to both the Server and the Broker. The Server will not grant any access privileges to the Client until the Broker responds a message to the Server saying this same Client has also authenticated at the Broker side. Once the Server receives this response from the Broker, it will then grant the access privilege to the Client.

For demonstration purposes, Client's login credentials will be just a unique ID that is stored on Server and Broker, this ID needs to be same on Server and Broker, if this ID matches the record on the Server, the Server will send this ID to the Broker and wait for the response.

- Broker: The Broker in this system will act as the second factor of authentication. When the Server first receives the authentication request, it will ask the Broker about the authentication progress of a specific Client by sending the ID of the Client to the Broker. The Broker will only reply to such a request message from the Server when the Client has to try to authenticate at the Broker using the same Client ID at the Server. If the Client's ID exists on the Broker's record, the Broker will reply to the Server's message by sending another message containing the Client's ID to inform the Server that this Client has authenticated at the Broker.

  At this point, the Client is in a state where it has the access privilege at both the Server and the Broker side.

- Client: The Client can represent both the users and things as explained in the previous chapter since both of them shares the same authentication procedure, it is only the credentials that used for them are different. For testing purposes, the client's login credentials will be represented by a unique ID that is stored on both

the Server and the Broker. This unique ID will be used extensively as the message content between these three entities.

For demonstration purposes, the Client will always try to authenticate at the Server first, then after a pre-configured delay period (usually a few seconds), the Client will try to authenticate at the Broker.

To quickly summarize the flow of the authentication process:

- Step 1: The Server and Broker will be launched and waiting for the authentication request from the Client.
- Step 2: Client will be launched to send an authentication request to the Server first; upon receiving this request, Server will send a request to the Broker about the authentication status of this Client; The Broker will not take any action at this point
- Step 3: Client now will send another request to the Broker, and the Broker will check this Client's ID against the request from the Server, if they match the Broker will inform the Server that the same client has authenticated at the Broker side, and granting the Client access privilege at the Broker's side
- Step 4: Once the Server receives the confirmation from the Broker, the Server will grant the Client's access privilege at the Server as well

To visualize the sequence been described in the steps above, figure 5.1 is a sequence diagram for a client to successfully acquire access privilege.

5.2 Simulation Results and Evaluation of Two-Factor-Authentication Implementation

To illustrate that the solution described in the previous chapter has met the expectation, the simulation results will be presented in this section.

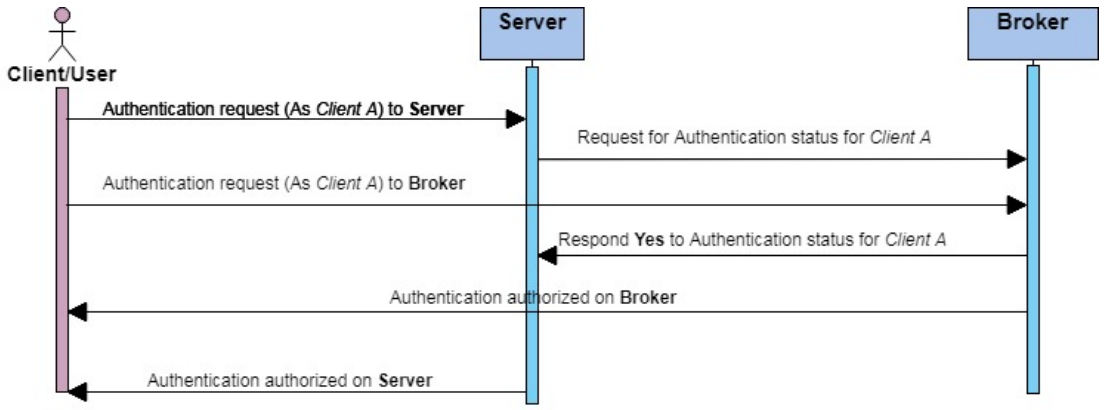


Figure 5.1: Authentication protocol sequence diagram

In Figure 5.2, it shows the results for one of the three components fail to connect to the mosquito broker due to connection related issues such as the broker been down or broker has blocked the connections coming from a specific IP address. Figure 5.3illustrates the error for SSL certificate verification related issues during the initial TLS handshake process. Specifically for this case, the certificate generated for this client is no longer valid due to its IP address has been updated. As illustrated in Figure 5.4, for a specific component to connect to Mosquitto broker using TLS encryption mode, it needs to specify the certificate in the source code, as well as the destination host IP address.

When all three components (Server, Broker, and Client) can connect to the Mosquitto broker in the sequence described in 5.1.2, the simulation results are shown in Figure 5.5, 5.6, and 5.7. The steps shown in each figure are referring to the steps been described in section 5.1.2.

Figure 5.8 and Figure 5.9 are the screenshots from Wireshark for capturing the packets been transmitted during the non-encrypted communication between the three components. The transmitted messages are shown above the red highlighted lines; the protocol shown for the captured event is MQTT in this case.

Figure 5.10 is the screenshot from Wireshark for capturing the packets been transmitted during the TLS encrypted communication between the three components. The transmitted messages are encrypted as shown above the red highlighted lines, therefore the message content is unknown to us; the protocol shown for the captured event is

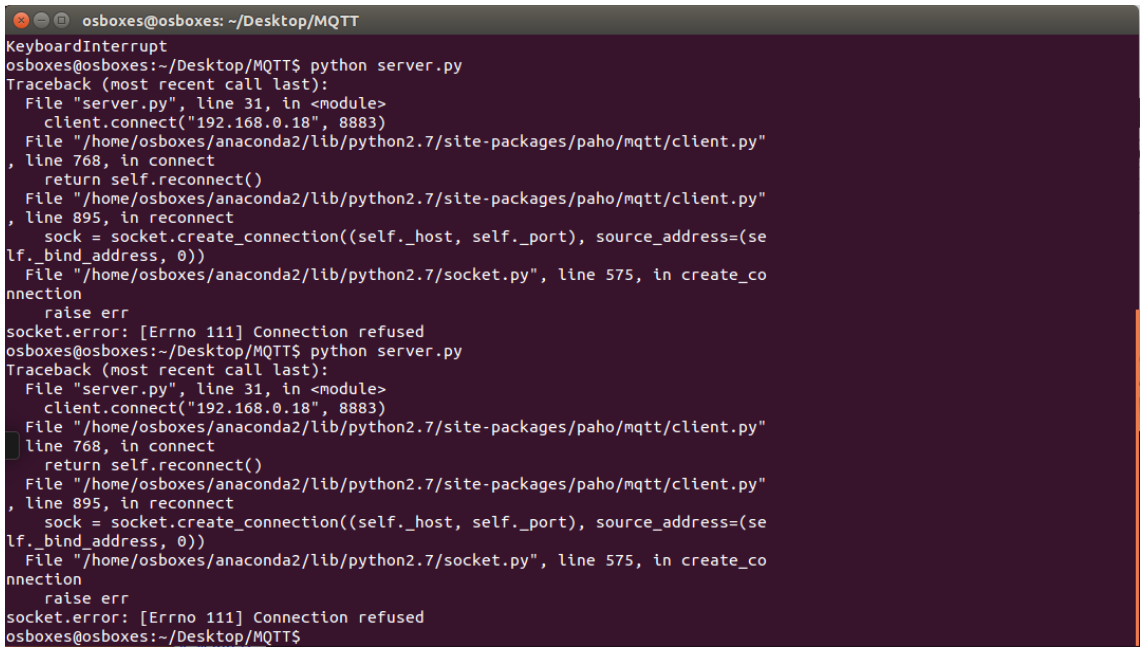


Figure 5.2: Connection refused by the server.

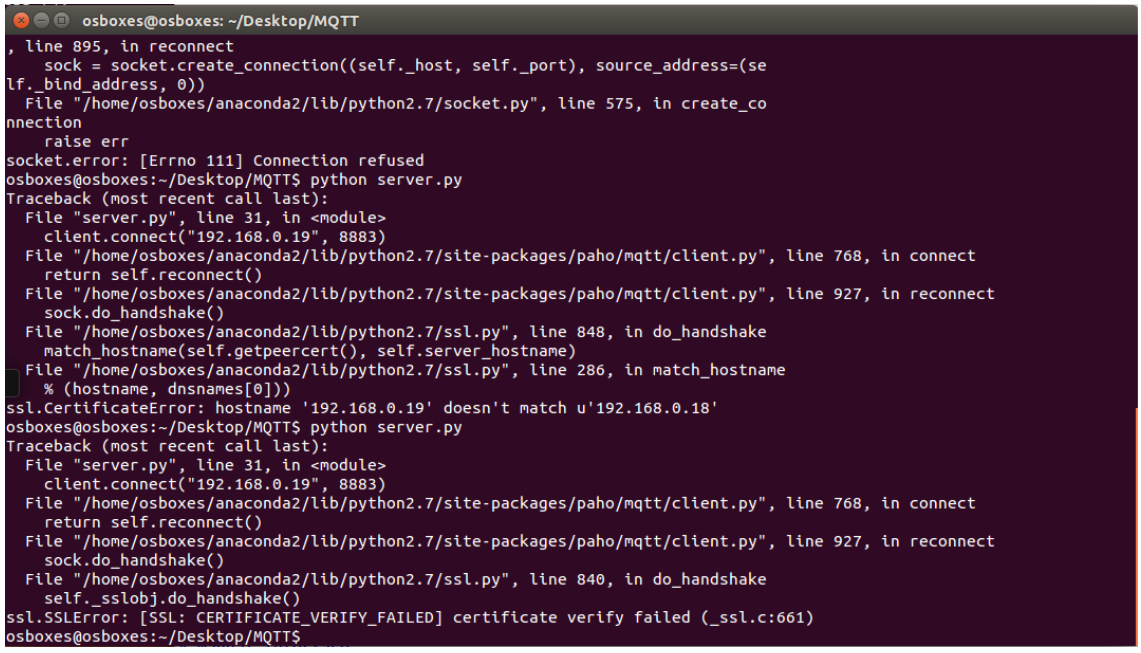


Figure 5.3: Connection refused by the server due to SSL related error.

```python
def on_connect_2(client, userdata, flags, rc):
    print("Connected to remote broker with result code "+str(rc))
    # Subscribing in on_connect() means that if we lose the connection and
    # reconnect then subscriptions will be renewed.
    message = "2nd instance"
    #print("rc: " + str(rc))
    #client.subscribe("login");
    #client2.publish("client",message);

    print("publishg msg: " + str(message))


# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    print("msg " +msg.topic+" ... "+str(msg.payload))



client = mqtt.Client(client_id="client", clean_session=True, userdata=None, protocol=3, transport="tcp")
#client = mqtt.Client(client_id="client")
#client.tls_set('/home/osboxes/Desktop/Certs/ca.crt')
client.tls_set('/home/osboxes/Desktop/Certs/18/server.crt')
client2 = mqtt.Client(client_id="client_copy", clean_session=True, userdata=None, protocol=3, transport="tcp")

client2.on_connect = on_connect_2
client.on_connect = on_connect


client.on_message = on_message

#client2.connect("192.168.0.19",1883, 60)
#client.connect("192.168.0.15", 8883, 60)
client.connect("192.168.0.18", 8883)

message = "2nd instance"
#client2.publish("client",message);

# Blocking call that processes network traffic, dispatches callbacks and
# handles reconnecting.
# Other loop*() functions are available that give a threaded interface and a
# manual interface.
client.loop_forever()
```

Figure 5.4: Sample implementation source code.

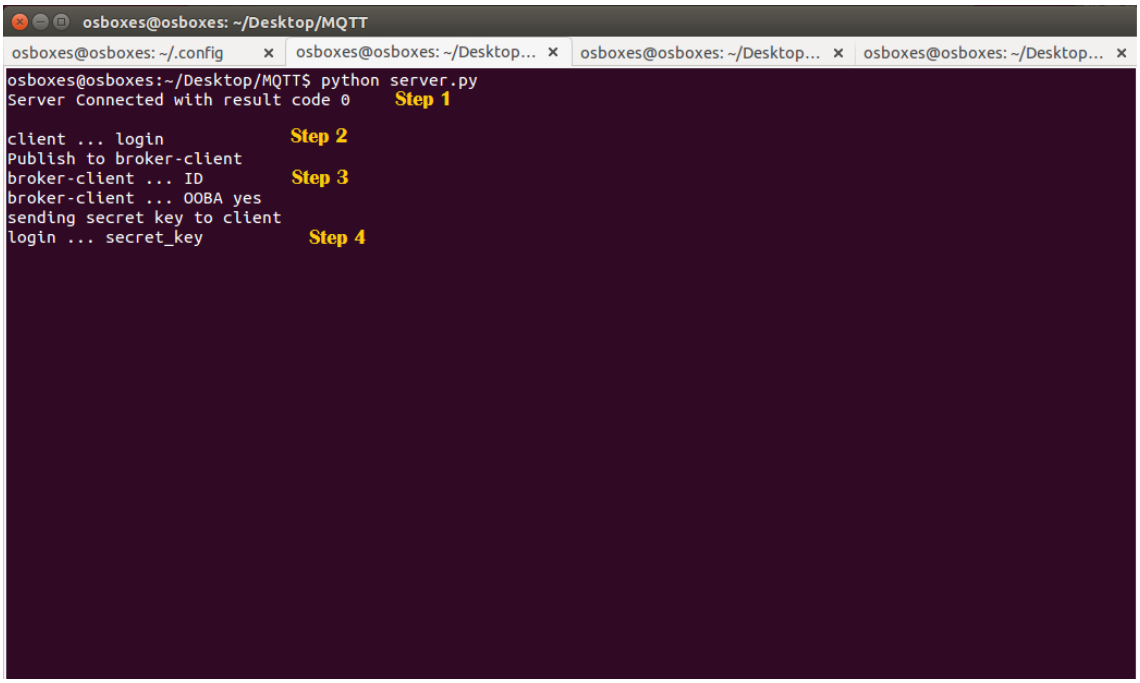


Figure 5.5: Output result for Server component.

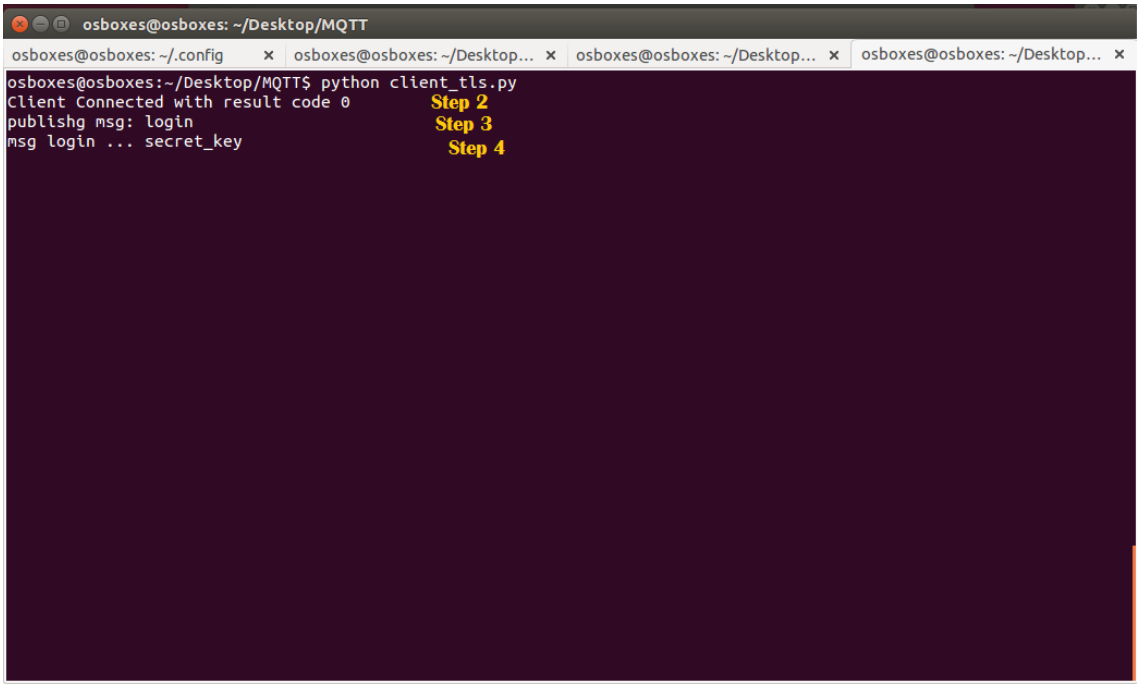


Figure 5.6: Output result for Client component.

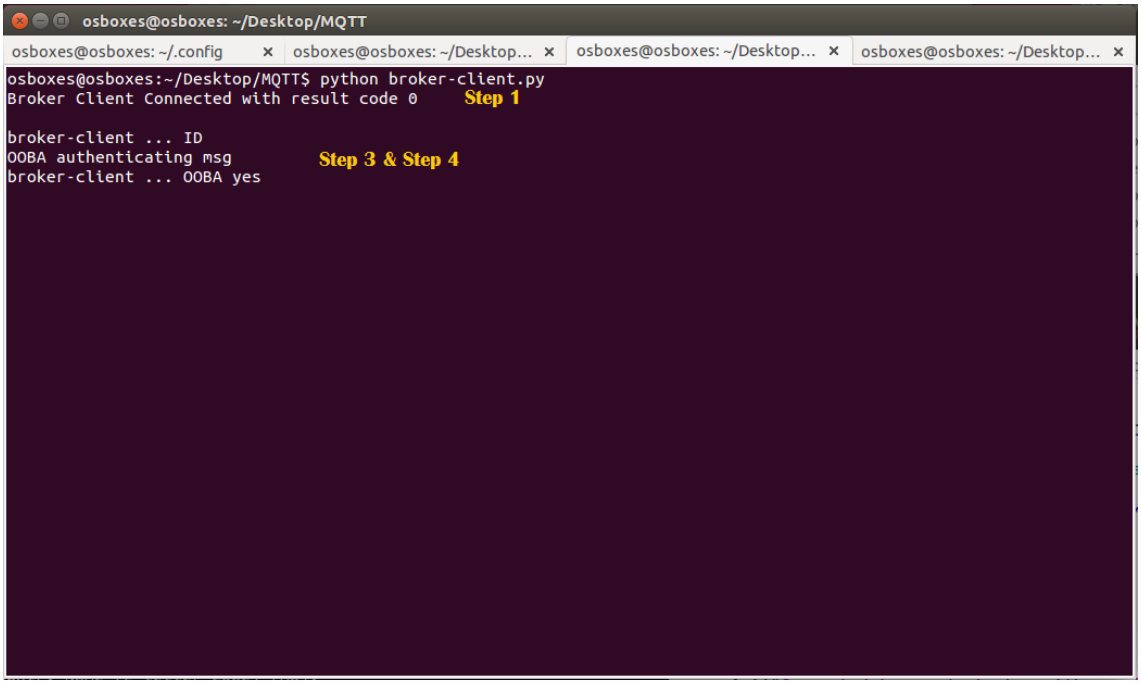


Figure 5.7: Output result for Broker component.

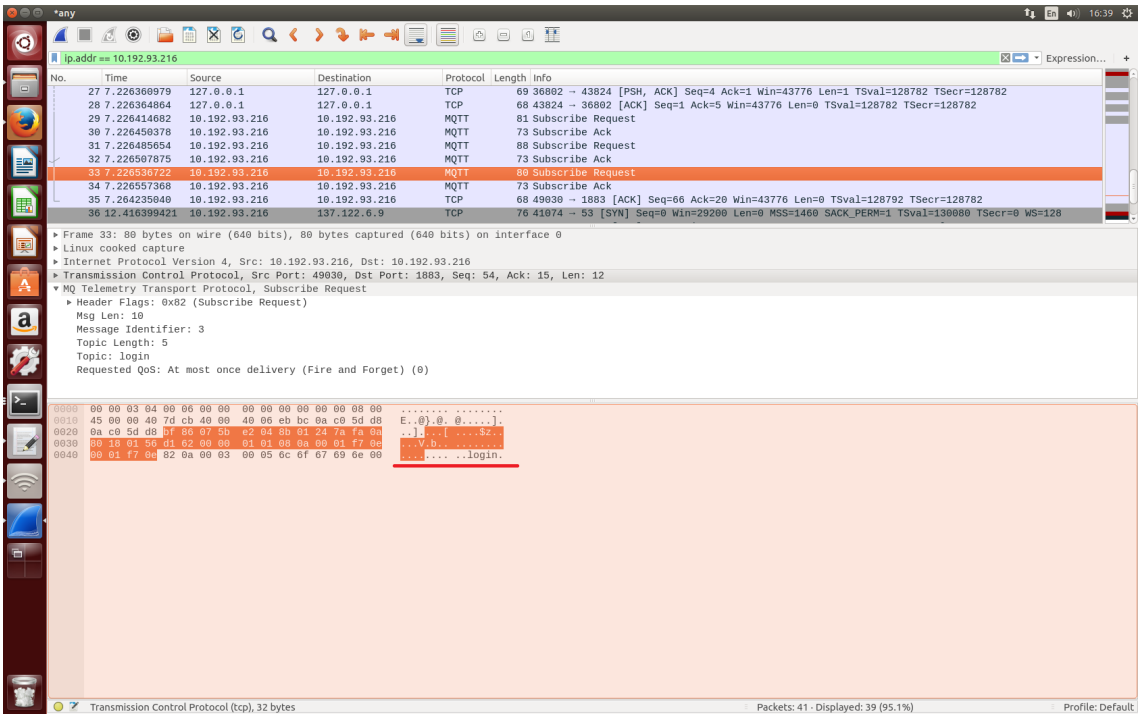


Figure 5.8: Wireshark packet capture for non-encrypted MQTT connections.

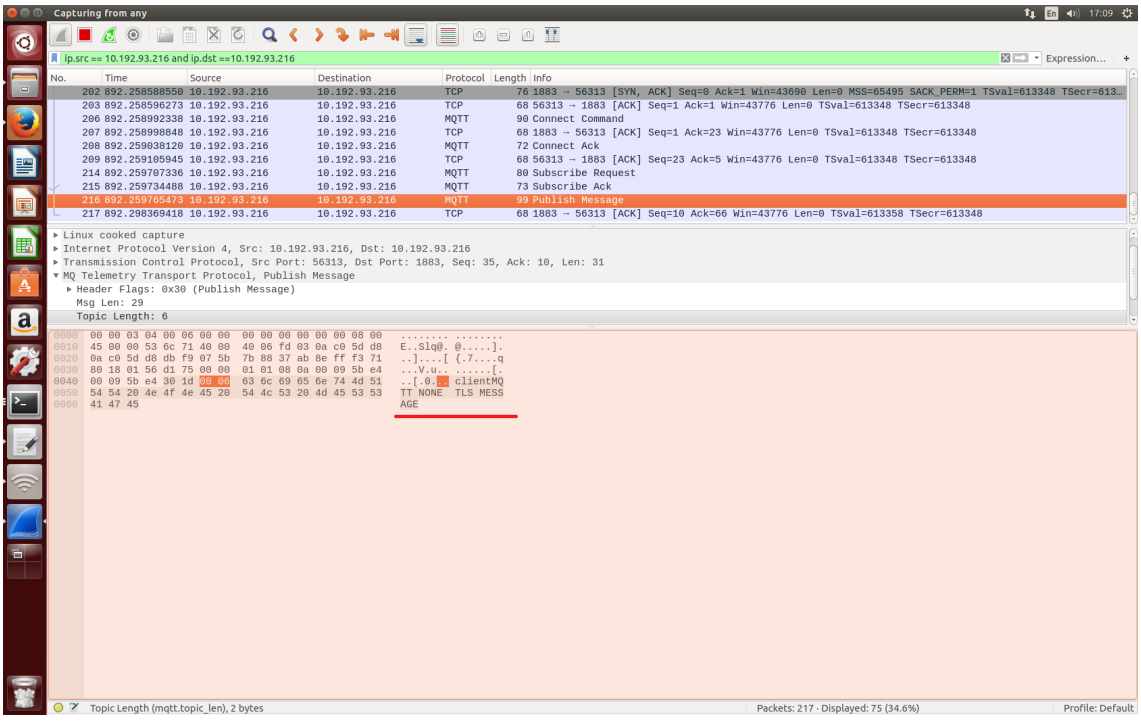


Figure 5.9: Wireshark packet capture for non-encrypted MQTT connections.

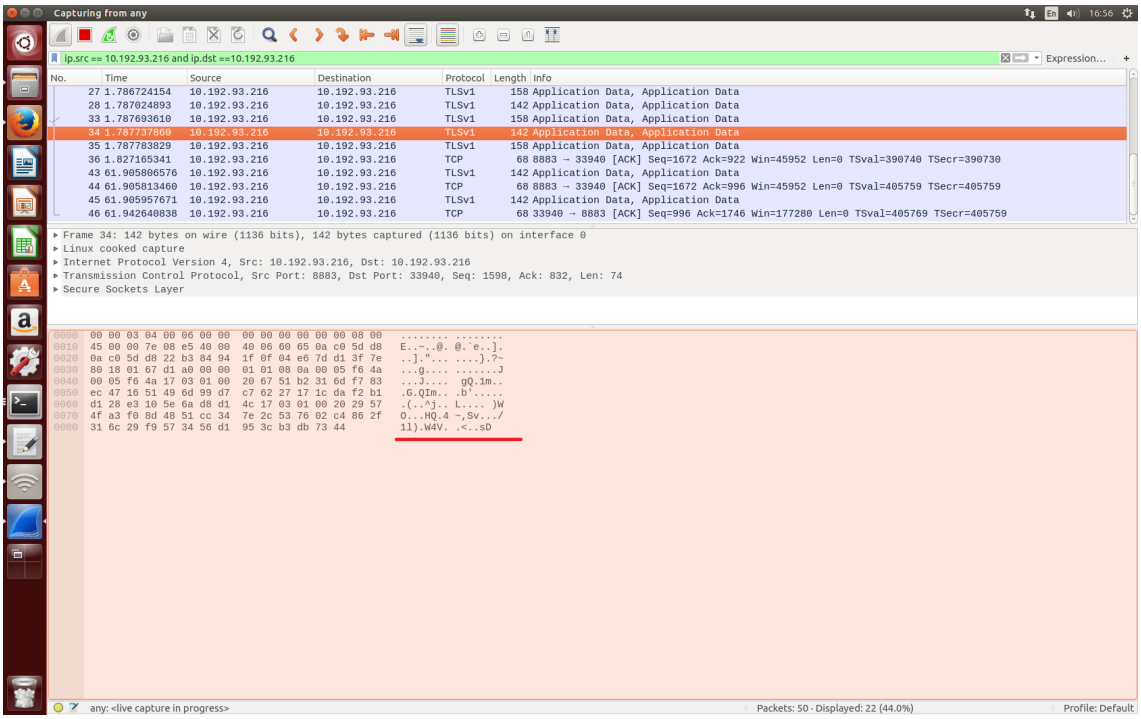


Figure 5.10: Wireshark packet capture for encrypted TLS MQTT connections.

TLS rather than MQTT because TLS is been used on top of MQTT so that Wireshark interpret the protocol as TLS is as expected.

5.3 Coloured Petri Nets Simulation

This section will explain how the CPN Tools were used to simulate the TFA protocol that has been discussed extensively for this thesis.

5.3.1 Expressing the TFA Protocol Using CPN

Before diving into the actual CPN simulations, the specific setup of this CPN needs to be explained in order to understand how it represents and can fully simulate the TFA protocol.

As shown in Figure 5.11, this is the initial setup of the Petri Nets that represents the intial state of the TFA protocol. Each component that involved in the TFA protocol is represented using a place, where two extra components are used to represent the state

transitioning of the intial components:

- Client: represents the initial state of the user/clients in the protocol when it is about to send the authentication request to both the Server and Broker. It holds one token at the beginning, which will be fired to Server and Broker once the simulation starts.

- Server: represents the initial state of the Server in the protocol. It is waiting for the authentication request token from the Client; upon receiving the token from the Client, it will fire the token to the Broker and "Server stage 2", which represents the server is now requesting the client's authentication status on the broker, and have now transitioned itself into a stage that waits for the confirmation from the Broker;

- Broker: represents the initial state of the Broker in the protocol. It is waiting for authentication tokens from both the Client and the Server. Since in the TFA protocol we have assumed that Server will always receive the authentication request first to simplify the process, once the Broker receives both tokens it will fire 2 tokens to "Server stage 2".

- Server stage 2: represents the state when the Server has already sent a message to Broker and it is waiting for the Broker's response. It will only fire when it receives 2 tokens from Broker and 1 token from Server.

- Done: represents that both Server and Broker have granted the Client authorization into the system, and the authentication process is complete once the three tokens have reached this place.

One thing to note is that a Place in CPN does not actually fire a token. The Transition that connects with the Place is the one that does the firing of the tokens.

5.3.2 Simulate the TFA Protocol using the CPN Tools

After setting up the CPN, the next step is to simulate the nets using the CPN Tools. Figure 5.11 - Figure 5.15 illustrate the authentication process of the TFA protocol using

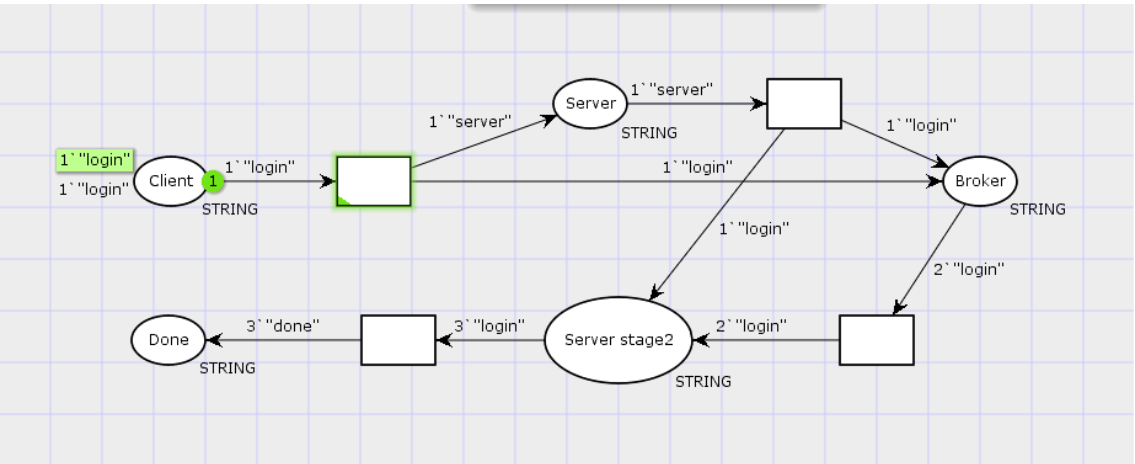


Figure 5.11: CPN - initial state

CPN:

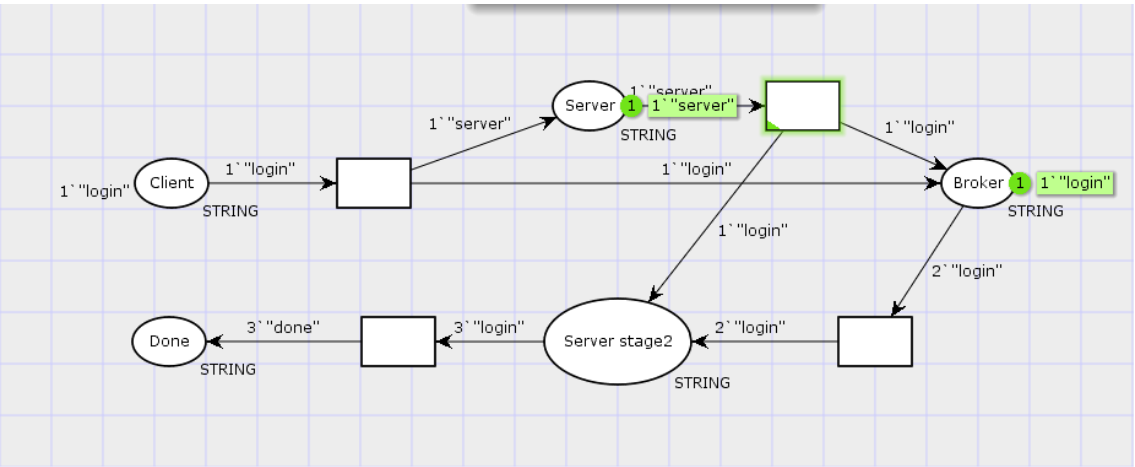


Figure 5.12: CPN simulation - step 1

- Figure 5.11 is the initial state, where the Client is about to send the authentication request.
- In Figure 5.12, both Server and Broker had received the authentication request from the client, where each place have one token; then Sever will fire its token to Broker to verify Client's authentication status, and another token to Server stage 2 to represent Server has transitioned its state to waiting for Broker's response state.

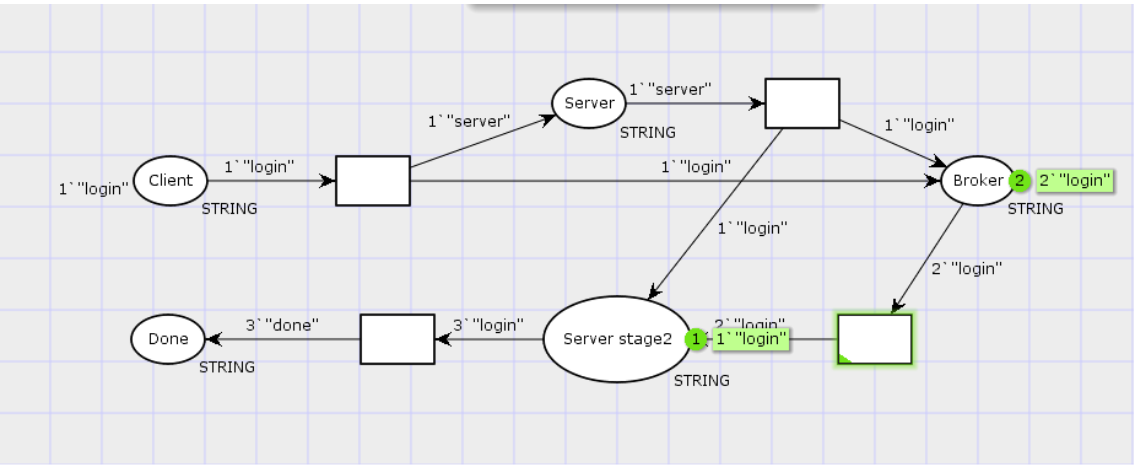


Figure 5.13: CPN simulation - step 2

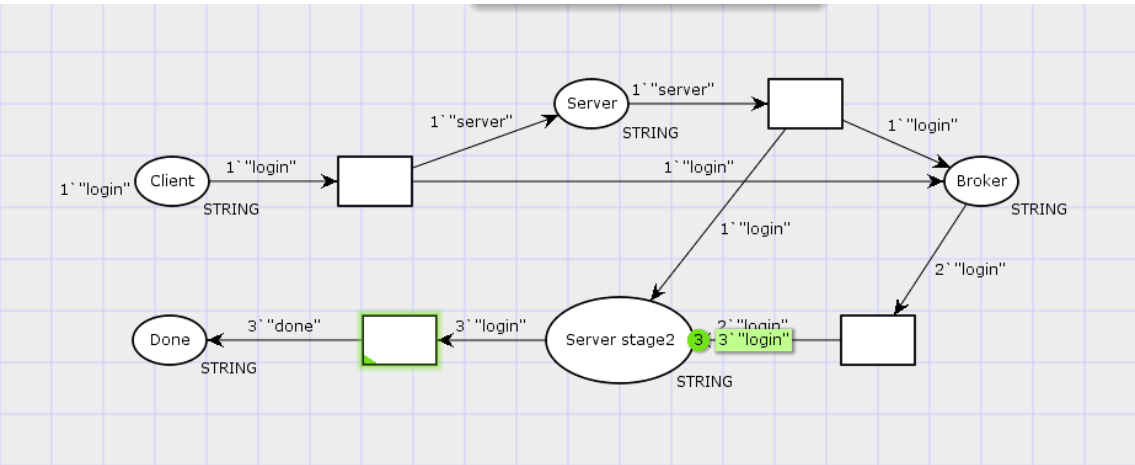


Figure 5.14: CPN - step 3

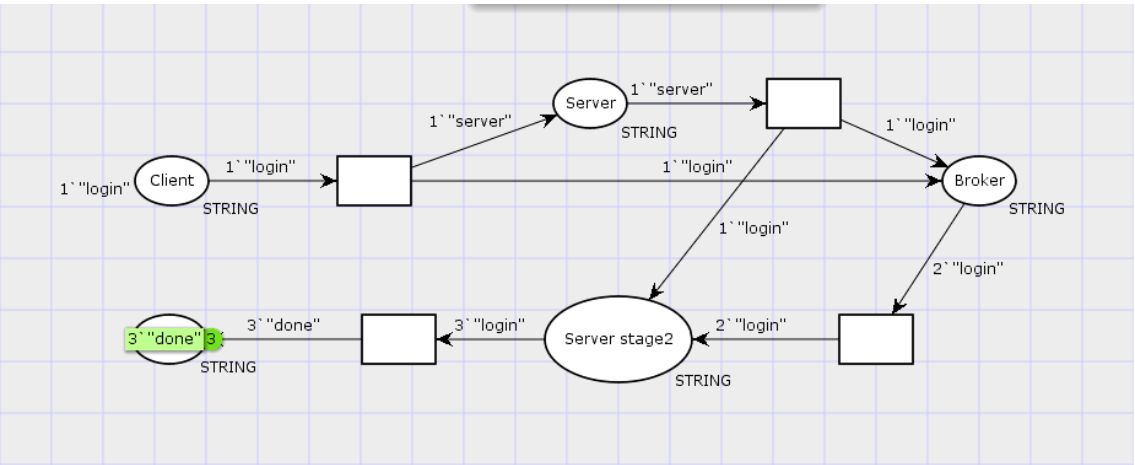


Figure 5.15: CPN - step 4

- In Figure 5.13, upon receiving two tokens from Client and Server, which represents Client has authenticated, and the Server is requesting Client's authentication status; Broker now has two tokens, and will not fire to "Server stage 2" until it has received two tokens.

- In Figure 5.14, Broker fired two tokens to "Server stage 2", representing Server has received confirmation from the Broker, and can grant the Client its appropriate access. "Server stage 2" will not fire until it has three tokens

- Figure 5.15 represents when the authentication has completed by having three tokens at the Done stage; At this point, the CPN simulation is completed and can no longer continue.

By simulation this CPN, it demonstrates that the TFA protocol possesses the properties of CPN, including no deadlock or livelock, and the simulation is consistent when repeating the process. Therefore, the simulation proves the stability of the protocol using CPN as the verification tool.

5.4 Summary

Through the simulation results presented in the previous section, Wireshark can be treated as the third party attacker trying to hack into the system by intercepting the communication message between each component. TLS encryption is the first layer of defense against the attackers in the system. Even if the attacker manages to intercept and decrypt the message between the Client and the Server, it will still need to do the same thing for the message between the Client and the Broker as well. In this case, the probability for both communication channels been hacked all at once becomes immensely smaller in comparison with one single communication channel using TLS encryption, which is the state of the art in the IoT security field.

By introducing Out-of-Band TFA into an IoT system architecture, it covers many flaws that are not covered by TLS. Because of the probability been extremely rare for any hacker to successfully hack into both authentication channels, hence making the TFA

protocol is end-to-end secure for IoT system architecture in comparison with the existing IoT infrastructure. By using CPN to simulate the TFA protocol, it proves the simulated TFA nets possesses the basic properties of the CPN, and ensures that the protocol can be executed according to the specification.

Now referring back to Figure 2.7 and 3.4 which outlines the security and privacy requirements for IoT system. Since we have achieved end-to-end security with our Out-of-Band TFA architecture, hence the categories for Authentication and Identity management, Trust management and policy integration, and End-to-security are satisfied for IoT systems.

Chapter 6

Conclusion and Future Work

6.1 Conclusions and Limitation

The purpose of this thesis is to design an end-to-end secured system architecture for IoT related applications. Although there are many existing techniques such as TLS or DTLS been used extensively, these techniques can only provide a point-to-point secured communication channel. In terms of overall system security, there is no guarantee since if any of the entities within the communication path is compromised, there is no way to detect such events from happening, and the whole system is exposed.

The system architecture proposed in this thesis is very generic, which means it can be deployed into virtually any IoT system. The solution utilizes the Two Factor Authen-

tication protocol explained in Chapter 3, which solves the issues in "the Authentication and identity management" and "Trust management and policy integration" in the IoT security requirements. Furthermore, we can also implement a Central Authority in the Server to solve "the User privacy and data protection" and "Authorization and access control" categories as the client's access can be embedded with their access certificate. TFA protocol can greatly reduce the risk from outside attacks, while the Central Authority can help mitigate the risk originated from inside the system; hence, creating an end-to-end secure IoT system.

In theory, this architecture is able to solve all the issues outlined in Security and Privacy Requirements for the Internet of Things back in Chapter 2, but there are still certain limitations and drawbacks to it. One of the major issues is that it has limited capability in dealing with physical attacks and tampering with the hardware/software in the device. Some other drawbacks are the increase in the system's complexity; depending on the system requirements and the amount of authentication requests, using TFA may result in slowing down the whole authentication process and reducing the system's performance. In the case when we are discussing systems that need to handle the huge number of authentication requests, it may results in crashing the Server.

6.2 Future Work

The TFA protocol was designed to work for any IoT system. However, when other messaging protocols other than MQTT is used, especially if the messaging protocol is not broker based, the overall design of the architecture may need to alter accordingly to meet the specific requirement of the system. Nonetheless, the concept of TFA protocol is applicable regardless of any messaging protocol been discussed in this thesis.

Some other future work directions could be to address the physical attack and hardware/software tampering issues mentioned in the previous section, a security tool suite called Cloakware developed by Irdeto can be incorporated into the system to help address the issue [39]. Cloakware can help Safeguard critical functionality in modules and devices from perimeter security breaches, as well as guarding against Anti-tamper, anti-reverse

engineering tools.

In comparison, other issues mentioned are more complicated, it will involve carefully designing the system to minimize the wait time added by the extra steps in the TFA protocol. Another aspect is the load balancing on the Server when the amount of request becomes large. In some extreme cases, a firewall will be required to guard against DDoS attacks.

To summarize, the TFA protocol presented in this thesis is still an overview picture of the architecture, each system will need to design accordingly in order to maximizing system security without sacrificing the overall system performance.

References

[1] Tadao Murata. "Petri nets: Properties, analysis and applications". In: *Proceedings of the IEEE* 77.4 (1989), pp. 541–580.

[2] Wolfgang Reisig and Grzegorz Rozenberg. *Lectures on petri nets i: basic models: advances in petri nets*. Springer Science & Business Media, 1998.

[3] Kurt Jensen and Lars M Kristensen. *Coloured Petri nets: modelling and validation of concurrent systems*. Springer Science & Business Media, 2009.

[4] Luigi Atzori, Antonio Iera, and Giacomo Morabito. "The internet of things: A survey". In: *Computer networks* 54.15 (2010), pp. 2787–2805.

[5] Rolf H Weber. "Internet of Things–New security and privacy challenges". In: *Computer law & security review* 26.1 (2010), pp. 23–30.

[6] Tet Hin Yeap et al. *Adapter for secure VoIP communications*. US Patent 7,660,575. 2010.

[7] Rodrigo Roman, Pablo Najera, and Javier Lopez. "Securing the internet of things". In: *Computer* 44.9 (2011), pp. 51–58.

[8] Ms Veena Bharti and Sachin Kumar. "Survey of network protocol verification techniques". In: *International Journal of Scientific and Research Publications* 2.4 (2012), pp. 228–231.

[9] Rafiullah Khan et al. “Future internet: the internet of things architecture, possible applications and key challenges”. In: (2012), pp. 257–260.

[10] Denis Kozlov, Jari Veijalainen, and Yasir Ali. “Security and privacy threats in IoT architectures”. In: (2012), pp. 256–262.

[11] Quangang Wen, Xinzheng Dong, and Ronggao Zhang. “Application of dynamic variable cipher security certificate in internet of things”. In: *Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*. Vol. 3. IEEE. 2012, pp. 1062–1066.

[12] Jayavardhana Gubbi et al. “Internet of Things (IoT): A vision, architectural elements, and future directions”. In: *Future generation computer systems* 29.7 (2013), pp. 1645–1660.

[13] Lili Yang, Shuang-Hua Yang, and Linda Plotnick. “How the internet of things technology enhances emergency response operations”. In: *Technological Forecasting and Social Change* 80.9 (2013), pp. 1854–1867.

[14] Mohamed Abomhara and G. M. Køien. “Security and privacy in the Internet of Things: Current status and open issues”. In: (2014), pp. 1–8. DOI: 10.1109/PRISMS.2014.6970594.

[15] Flavio Bonomi et al. “Fog computing: A platform for internet of things and analytics”. In: *Big data and internet of things: A roadmap for smart environments*. Springer, 2014, pp. 169–186.

[16] Shuang-Hua Yang. “Internet of Things”. In: *Wireless Sensor Networks: Principles, Design and Applications*. London: Springer London, 2014, pp. 247–261. ISBN: 978-1-4471-5505-8. DOI: 10.1007/978-1-4471-5505-8_12. URL: https://doi.org/10.1007/978-1-4471-5505-8_12.

[17] Muhammad Umar Farooq et al. “A critical analysis on the security concerns of internet of things (IoT)”. In: *International Journal of Computer Applications* 111.7 (2015).

[18] Rwan Mahmoud et al. “Internet of things (IoT) security: Current status, challenges and prospective measures”. In: (2015), pp. 336–341.

[19] Ahmad-Reza Sadeghi, C. Wachsmann, and M. Waidner. “Security and privacy challenges in industrial Internet of Things”. In: (2015), pp. 1–6. ISSN: 0738-100X. DOI: 10.1145/2744769.2747942..

[20] Sabrina Sicari et al. “Security, privacy and trust in Internet of Things: The road ahead”. In: *Computer networks* 76 (2015), pp. 146–164.

[21] Charles Bell. *MySQL for the Internet of Things*. Springer, 2016.

[22] Manuel Dıéaz, Cristian Martıén, and Bartolomé Rubio. “State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing”. In: *Journal of Network and Computer applications* 67 (2016), pp. 99–117.

[23] Amir Manzoor. “Securing Device Connectivity in the Industrial Internet of Things (IoT)”. In: *Connectivity Frameworks for Smart Devices: The Internet of Things from a Distributed Computing Perspective*. Ed. by Zaigham Mahmood. Cham: Springer International Publishing, 2016, pp. 3–22. DOI: 10.1007/978-3-319-33124-9_1. URL: https://doi.org/10.1007/978-3-319-33124-9_1.

[24] *Meshcentral Technologies*. [Online; accessed 12-March-2019]. 2016. URL: https://meshcentral.com/.

[25] Wazen Shbair et al. “A multi-level framework to identify HTTPS services”. In: Apr. 2016. DOI: 10.1109/NOMS.2016.7502818.

[26] Asif Amin, Israr ul Haq, and Monisa Nazir. “Two Factor Authentication”. In: *International Journal of Computer Science and Mobile Computing* (2017).

[27] Nitin Naik. “Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP”. In: (2017), pp. 1–7.

[28] Pallavi Sethi and Smruti R Sarangi. “Internet of things: architectures, protocols, and applications”. In: *Journal of Electrical and Computer Engineering* 2017 (2017).

[29] Mohammad Salah Uddin, Jannat B. Alam, and Suraiya Banu. "Real time patient monitoring system based on Internet of Things". In: *2017 4th International Conference on Advances in Electrical Engineering (ICAEE)* (2017), pp. 516–521.

[30] Patrick Killeen et al. "IoT-based predictive maintenance for fleet management". In: *Procedia Computer Science* 151 (2019), pp. 607–613.

[31] Patrick Killeen. "Knowledge-Based Predictive Maintenance for Fleet Management". PhD thesis. Université d'Ottawa/University of Ottawa, 2020.

[32] *Ben Armstrong's Virtualization Blog.* [Online; accessed 28-May-20198]. URL: https://www.getkisi.com/blog/internet-of-things-communication-protocols.

[33] *CPN Tools.* [Online; accessed 25-April-2019]. URL: http://cpntools.org/.

[34] *GlobalSign Blog.* [Online; accessed 12-January-2019]. URL: https://www.globalsign.com/en/blog/benefits-of-multi-factor-authentication/.

[35] *Google Cloud.* [Online; accessed 11-June-2019]. URL: https://cloud.google.com/blog/products/iot-devices/http-vs-mqtt-a-tale-of-two-iot-protocols.

[36] *HiveMQ.* [Online; accessed 12-March-2019]. URL: https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt.

[37] *HiveMQ.* [Online; accessed 12-March-2019]. URL: https://en.wikipedia.org/wiki/Transport_Layer_Security.

[38] *HiveMQ.* [Online; accessed 12-March-2019]. URL: https://blogs.msdn.microsoft.com/virtual_pc_guy/2014/06/09/ubuntu-14-04-in-a-generation-2-vm/.

[39] *Irdeto.* [Online; accessed 12-March-2019]. URL: https://irdeto.com/cloakware-by-irdeto/.

[40] *JasonBrazeal Blog.* [Online; accessed 16-October-2018]. URL: http://jasonbrazeal.com/blog/how-to-build-a-simple-iot-system-with-python/.

[41] Donald Knuth. *MQTT Version 3.1.1. Edited by Andrew Banks and Rahul Gupta.* [Online; accessed 12-March-2019]. URL: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html.

[42] *Lynda*. [Online; accessed 17-April-2019]. URL: http://www.lynda.com/SQLite/.

[43] *Steves-internet-guide*. [Online; accessed 21-November-2018]. URL: http://www.steves-internet-guide.com/mosquitto-tls/.

[44] *techopedia*. [Online; accessed 11-May-2019]. URL: https://www.techopedia.com/definition/29532/out-of-band-authentication-ooba.

[45] *Wikipedia*. [Online; accessed 12-January-2019]. URL: https://en.wikipedia.org/wiki/Petri_net.

[46] *Wikipedia*. [Online; accessed 21-May-2019]. URL: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc785811(v=ws.10).

[47] *Wikipedia*. [Online; accessed 11-May-2019]. URL: https://en.wikipedia.org/wiki/Multi-factor_authentication.