

DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

**IDENTITY VERIFICATION IN INTERNET OF
THINGS DEVICES**

by
Hakan DALKILIÇ

April, 2022
İZMİR

IDENTITY VERIFICATION IN INTERNET OF THINGS DEVICES

**A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Doctor of
Philosophy in Computer Engineering**

by

Hakan DALKILIÇ

April, 2022

İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled “**IDENTITY VERIFICATION IN INTERNET OF THINGS DEVICES**” completed by **HAKAN DALKILIÇ** under supervision of **ASSOC. PROF. DR. MEHMET HİLAL ÖZCANHAN** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

.....
Assoc. Prof. Dr. Mehmet Hilal ÖZCANHAN

Supervisor

.....
Asst. Prof. Dr. Özlem AKTAŞ

Thesis Committee Member

.....
Asst. Prof. Dr. Reyat YILMAZ

Thesis Committee Member

.....
Prof. Dr. Mehmet Süleyman ÜNLÜTÜRK

Examining Committee Member

.....
Prof. Dr. Murat KOMESLİ

Examining Committee Member

Prof.Dr. Okan FISTIKOĞLU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENT

I would like to thank my advisor Associate Professor Dr. Mehmet Hilal ÖZCANHAN for his guidance throughout this journey in my thesis.

I extend my thanks to the members of my committee, Assistant Professor Dr. Özlem AKTAŞ and Assistant Professor Dr. Reyat YILMAZ for their contribution to this study and sharing their ideas during development and writing of this thesis.

I would like to thank my brother Gökhan DALKILIÇ and his precious wife Feriştah DALKILIÇ for all their support during my doctoral studies.

Finally, I would like to thank my parents who are always there for me. They have always supported me.

Hakan DALKILIÇ

IDENTITY VERIFICATION IN INTERNET OF THINGS DEVICES

ABSTRACT

The scope of services provided to the citizens in smart cities is expanding with the global spread of Internet of things (IoT) devices. However, the security of personal data may be at risk, especially due to the use of IoT devices in areas such as healthcare, transportation, and education, in smart cities.

In this thesis, a five-step mutual authentication protocol (FSMAP) has been developed to ensure the security of message exchanges between IoT devices. The developed protocol was implemented on a custom made smart textile device. The created smart textile device can be worn on the arm of a person and the inner angle of the arm can be measured to determine the mobility of the arm. The smart textile data collected by the sensors is first encrypted using the AES encryption algorithm and stored on the custom made IoT device. Afterward, the stored data is transferred securely to another IoT device through a strong wireless mutual authentication protocol.

Informal and formal analysis of the developed authentication protocol was carried out. According to the informal analysis result, the proposed protocol is safe against replay, full-disclosure, ID-theft, man-in-the-middle, de-synchronization, identity tracing, impersonation, DoS, clone, and malicious code injection attacks. In addition, Scyther and AVISPA tools were used, in formal analysis. Both analyses proved that the developed authentication protocol provides secure message exchange between IoT devices. Comparison with previous works showed that FSMAP is stronger and more secure.

Keywords: Internet of things, authentication, smart city, smart textile devices

NESNELERİN İNTERNETİ CİHAZLARINDA KİMLİK GÜVENLİĞİ

ÖZ

Akıllı şehirlerde kentlilere sağlanan hizmetlerin kapsamı Nesnelerin İnterneti (Internet of Things, IoT) cihazlarının küresel olarak yaygınlaşmasıyla genişlemektedir. Bununla birlikte, IoT cihazlarının akıllı şehirlerde özellikle sağlık, ulaşım, eğitim gibi alanlarda kullanılması nedeniyle kişisel verilerin güvenliği risk altında olabilmektedir.

Bu tezde, IoT cihazları arasındaki mesaj alışverişlerinin güvenliğini sağlamak için beş adımlı bir karşılıklı kimlik doğrulama protokolü (FSMAP) geliştirilmiştir. Geliştirilen protokol, kendi tasarımı olan bir akıllı tekstil cihazına uygulanmıştır. Oluşturulan akıllı tekstil cihazı kişinin koluna takılabilmekte ve kolun hareket kabiliyetini belirlemek için kolun iç açısı ölçebilmektedir. Sensörler tarafından toplanan akıllı tekstil verileri, önce AES şifreleme algoritması kullanılarak şifrelenmekte ve tasarlanan IoT cihazında saklanmaktadır. Ardından, depolanan veriler, kablosuz bir karşılıklı kimlik doğrulama protokolü aracılığıyla başka bir IoT cihazına güvenli bir şekilde aktarılmaktadır.

Geliştirilen kimlik doğrulama protokolünün informal ve formal analizi yapılmıştır. Informal analiz sonucuna göre, önerilen protokol replay, full-disclosure, ID-theft, man-in-the-middle, de-synchronization, identity tracing, impersonation, DoS, clone, ve malicious code injection saldırılarına karşı güvenlidir. Ayrıca, formal analizde Scyther ve AVISPA araçları kullanılmıştır. Her iki analiz de, geliştirilen kimlik doğrulama protokolünün IoT cihazları arasında güvenli mesaj alışverişini sağladığını kanıtlamaktadır. Önceki çalışmalarla yapılan karşılaştırma, FSMAP'in daha güçlü ve daha güvenli olduğunu göstermektedir.

Anahtar kelimeler: Nesnelerin interneti, kimlik doğrulama, akıllı şehir, akıllı tekstil cihazları

CONTENTS

	Page
Ph.D. THESIS EXAMINATION RESULT FORM	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZ	v
LIST OF FIGURES	xi
LIST OF TABLES	xiv
 CHAPTER ONE - INTRODUCTION	 1
1.1 Overview	1
1.2 Smart Cities	2
1.3 Internet of Things (IoT)	4
1.4 Internet of Things Applications for Smart Cities	5
1.4.1 Smart Parking	5
1.4.2 Smart Homes	5
1.4.3 Smart Grid	6
1.4.4 Smart Vehicular Traffic System	6
1.4.5 Smart Transportation	6
1.4.6 Smart Energy System	7
1.4.7 Smart Waste Management	7
1.4.8 Smart Health	8
1.5 Motivation	9
1.6 The Goal of the Thesis	9

1.7 Thesis Organization	10
CHAPTER TWO - BACKGROUND AND RELATED WORKS	12
2.1 Architecture of Embedded Systems	12
2.2 Architecture of IoT	14
2.3 Architecture of Wearable IoT Device	16
2.4 IoT Devices Functionality Layers	17
2.4.1 Three-Layer Architecture	18
2.4.1.1 Perception Layer	19
2.4.1.2 Network Layer	19
2.4.1.3 Application Layer	20
2.4.2 Five-Layer Architecture	22
2.4.2.1 Processing Layer	23
2.4.2.2 Business Layer	23
2.5 IoT Devices Security Requirements	23
2.5.1 Confidentiality	23
2.5.2 Integrity	24
2.5.3 Availability	24
2.5.4 Authentication	25
2.5.5 Authorization	25
2.5.6 Data Freshness	26
2.5.7 Non-Repudiation	26
2.6 Encryption Methods	26
2.7 AES (Advanced Encryption Standard) Encryption Algorithm	28
2.8 Types of Attacks in IoT	31

2.8.1 Replay Attack.....	31
2.8.2 Denial of Service (DoS) Attack	32
2.8.3 Man-in-the-Middle Attack	32
2.8.4 Full-disclosure Attack	33
2.8.5 ID-theft Attack	33
2.8.6 De-synchronization Attack.....	34
2.8.7 Identity Tracing Attack	34
2.8.8 Impersonation Attack.....	34
2.8.9 Clone Attack.....	34
2.8.10 Node Tampering Attack.....	35
2.8.11 Malicious Code Injection Attack	35
2.8.12 Jamming Attack	36
2.9 Literature Review.....	37

CHAPTER THREE – DESIGN AND IMPLEMENTATION OF A TYPICAL IOT DEVICE..... 54

3.1 Overview.....	54
3.2 Materials.....	54
3.3 Vulnerabilities in IoT	58
3.4 The Hardware Used in the Designed IoT Device	59
3.4.1 Arduino UNO.....	59
3.4.2 MPU 6050 Accelerometer & Gyro Sensor	61
3.4.3 Bluetooth Module.....	62
3.4.4 Woven Fabric	63
3.5 Programming of Our Designed Microcontroller Hardware	64

CHAPTER FOUR – PROPOSED SECURITY DESIGN AND IMPLEMENTATION ON OUR DESIGNED IOT DEVICE..... 66

4.1 Overview	66
4.2 Methods.....	66
4.2.1 Data Encryption	69
4.2.1.1 AES Implementation.....	70
4.2.1.2 AES Implementation Performance	70
4.2.1.3 Power Performance of Our AES Implementation.....	72
4.2.2 Five-Step Mutual Authentication Protocol (FSMAP)	74
4.3 Testing Platform.....	78
4.4 Testbed	80

CHAPTER FIVE – SECURITY ANALYSIS OF OUR DESIGN AUTHENTICATION METHOD 85

5.1 Overview	85
5.2 Informal Analysis.....	85
5.2.1 IoT Attacks.....	86
5.2.1.1 Replay Attack.....	86
5.2.1.2 Denial of Service (DoS) Attack	86
5.2.1.3 Man in the Middle Attack	87
5.2.1.4 Full-disclosure Attack	87
5.2.1.5 ID-theft Attack	87
5.2.1.6 De-synchronization Attack.....	88
5.2.1.7 Identity Tracing Attack	88
5.2.1.8 Impersonation Attack.....	89

5.2.1.9 Clone Attack.....	89
5.2.1.10 Node Tampering Attack.....	89
5.2.1.11 Malicious Code Injection Attack	90
5.2.1.12 Jamming Attack	90
5.3 Formal Analysis	90
5.3.1 AVISPA as a Formal Analysis Tool	91
5.3.1.1 Verification of Our Proposed Protocol Using AVISPA	93
5.3.2 Scyther as a Formal Analysis Tool	100
5.3.2.1 Verification of Our Proposed Protocol Using Scyther.....	104
5.4 Performance and Security Comparison with Previous Works.....	110
CHAPTER SIX - CONCLUSION	114
6.1 Conclusion	114
6.2 Future Works.....	116
REFERENCES.....	117

LIST OF FIGURES

	Page
Figure 1.1 IoT based services in smart cities	3
Figure 2.1 Vonn Neumann & Harvard architecture.....	12
Figure 2.2 Basic architecture of embedded systems	13
Figure 2.3 Embedded system subset	14
Figure 2.4 IoT architecture.....	16
Figure 2.5 Wearable IoT device architecture.....	17
Figure 2.6 Three-layer architecture.....	18
Figure 2.7 MQTT protocol.....	21
Figure 2.8 CoAP protocol	21
Figure 2.9 Five-layer architecture	22
Figure 2.10 CIA triad	25
Figure 2.11 Illustration of symmetric encryption	27
Figure 2.12 Illustration of asymmetric encryption.....	28
Figure 2.13 AES encryption and decryption process.....	30
Figure 2.14 Replay attack demonstration.....	31
Figure 2.15 DoS attack demonstration.....	32
Figure 2.16 Man-in-the-middle attack demonstration	33
Figure 2.17 Clone attack demonstration	35
Figure 2.18 Malicious code injection attack demonstration	36
Figure 2.19 Jamming attack demonstration	37
Figure 2.20 Proposed schema by Kalra & Sood (2015).....	47
Figure 2.21 Proposed schema by Tewari & Gupta (2017)	49
Figure 2.22 Proposed schema by Fu et al. (2010).....	51

Figure 2.23 Desynchronization attack on the proposed schema by Fu et al. (2010) .	52
Figure 3.1 MPU 6050 sensors	55
Figure 3.2 Smart textile	56
Figure 3.3 Smart textile circuit diagram	56
Figure 3.4 Smart textile prototype	57
Figure 3.5 Windows application	58
Figure 3.6 Arduino UNO rev3 microcontroller	60
Figure 3.7 MPU 6050 accelerometer & gyro sensor.....	62
Figure 3.8 HC-05 bluetooth module	62
Figure 3.9 Woven fabric	63
Figure 3.10 Schematic diagram of the woven fabrics.....	64
Figure 3.11 Arduino IDE software.....	65
Figure 4.1 Smart healthcare in smart city	67
Figure 4.2 Proposed design block diagram	69
Figure 4.3 Graphical illustration of Arduino UNO time measurement	71
Figure 4.4 Current measurement on Arduino UNO.....	73
Figure 4.5 Our five-step mutual authentication protocol (FSMAP)	75
Figure 4.6 Mathematical calculations in our protocol design	76
Figure 4.7 Designed smart textile prototype	79
Figure 4.8 Proposed design hardware architecture	80
Figure 4.9 Windows application for IoT data communication	81
Figure 5.1 Summary of cyber-attacks in IoT	85
Figure 5.2 SPAN tool for AVISPA.....	92
Figure 5.3 IoT sensor role implementation	93
Figure 5.4 IoT gateway role implementation	94
Figure 5.5 HLPSSL code role definition for IoT sensor	95

Figure 5.6 HLP SL code local variables definition for IoT sensor	95
Figure 5.7 HLP SL code transition section for IoT sensor	96
Figure 5.8 HLP SL code role definition for IoT gateway	97
Figure 5.9 HLP SL code local variables definition for IoT gateway	97
Figure 5.10 HLP SL code transition section for IoT gateway	98
Figure 5.11 Security evaluation result with AVISPA	99
Figure 5.12 Scyther tool	101
Figure 5.13 Global declarations	103
Figure 5.14 The Scyther code of the FS MAP	105
Figure 5.15 Scyther code for variable declarations	106
Figure 5.16 Scyther code for data transfer on the sensor side	107
Figure 5.17 Scyther code for security properties on the sensor side	107
Figure 5.18 Scyther code for data transfer on the gateway side	107
Figure 5.19 Variable definitions Scyther codes used by the gateway	108
Figure 5.20 Scyther code for security properties on the gateway side	108
Figure 5.21 Authentication protocol result by Scyther tool	109

LIST OF TABLES

	Page
Table 3.1 Arduino UNO hardware part description.....	61
Table 3.2 The specifications of yarns	63
Table 4.1 Arduino UNO time measurement	71
Table 4.2 Arduino UNO memory consumption.....	72
Table 5.1 Comparison of security features	113



CHAPTER 1

INTRODUCTION

1.1 Overview

The technological devices used today have become an indispensable part of our daily life. Devices, many of which have become mobile, communicate with each other over the internet and exchange data. The number of devices called the Internet of Things (IoT) devices that communicate over the Internet has reached billions all over the world. It is predicted that the number of mobile devices to be used will increase rapidly in the future. Especially in smart cities, the IoT devices, which are much more numerous, are used significantly in many areas such as health, transportation, and education.

IoT devices are also used on wearable items. Data is collected through fabrics on which sensors are placed, called smart textiles, and then these data are transferred wirelessly to the IoT gateway. With this created smart textile fabric, a wearable smart textile device is obtained. Thus, the process of collecting and evaluating the health data of the person is accelerated. In particular, the health status of elderly patients can be determined quickly, and measures can be taken.

One of the important problems encountered in the use of IoT devices is to ensure data security, because their data exchanges are accessible over the internet. Many IoT devices are connected to the internet at certain intervals during the day, such as devices used in healthcare for data collection from sick people, wireless sensors used in transportation and parking areas, and wireless sensors used in smart home systems that collect information from different utensils. The data security of the exchanged information through the internet is important, in ensuring the data security of the personal data. There are risks such as data modification and deletion as a result of

unauthorized access to IoT devices. There are some methods to minimize these risks. One of these methods is to authenticate IoT devices to communicate with each other.

Data in IoT devices can be transferred wirelessly to a central location such as cloud storage systems. Afterward, being able to access data quickly and safely is very important for smart cities in providing good quality services.

Authentication protocols should be used so that IoT devices are not affected by penetration attacks and can continue to work securely. Authentication protocols to be implemented should not cause serious losses in performance, by using the limited resources efficiently. However, authentication protocols should be designed to work in harmony and compatibly with developing technologies. In cases where the encryption algorithms used are out of date, the applicability of up-to-date encryption algorithms can also be used to ensure the applicability of the designed system for a longer period. Thus, people who benefit from the services offered by IoT devices will be able to access shared data and transfer information securely over the IoT device.

1.2 Smart Cities

Due to the developments in the field of information technologies, a change is taking place, including the social, cultural, and economic structure of cities (Köseoğlu & Demirci, 2018). As a result of this change, the concept of a smart city emerges. Smart cities enable the design of sustainable cities by using information technology infrastructures. By 2050, more than 55% of the world's population is expected to live in cities (Hernández et al., 2020). It will be inevitable for cities to quickly adapt to the smart city concept. Information and communication technologies are leading the way in the development of the smart city concept.

It is seen that it contributes to the management of cities using innovative solutions in technology and has become a city where problems are solved with faster and more

effective methods. It is seen that concepts such as cloud technologies, artificial intelligence, IoT, and the wearable devices are frequently used in smart city applications. Using these technologies, cities can offer application services such as elderly assistant, smart transportation, smart home, smart healthcare, smart agriculture, smart parking, smart surveillance system, smart vehicular traffic system, and smart grid.

With the IoT technology used in smart cities, many sensors can be remotely managed and monitored. With a remotely managed central structure, it is possible to intervene in a very short time in natural disasters, fires, traffic accidents, epidemics, and similar problems that may occur in smart cities. Thus, the quality of service received by the citizens increases. Figure 1.1 shows technology-based services that have an impact on the development of smart cities.

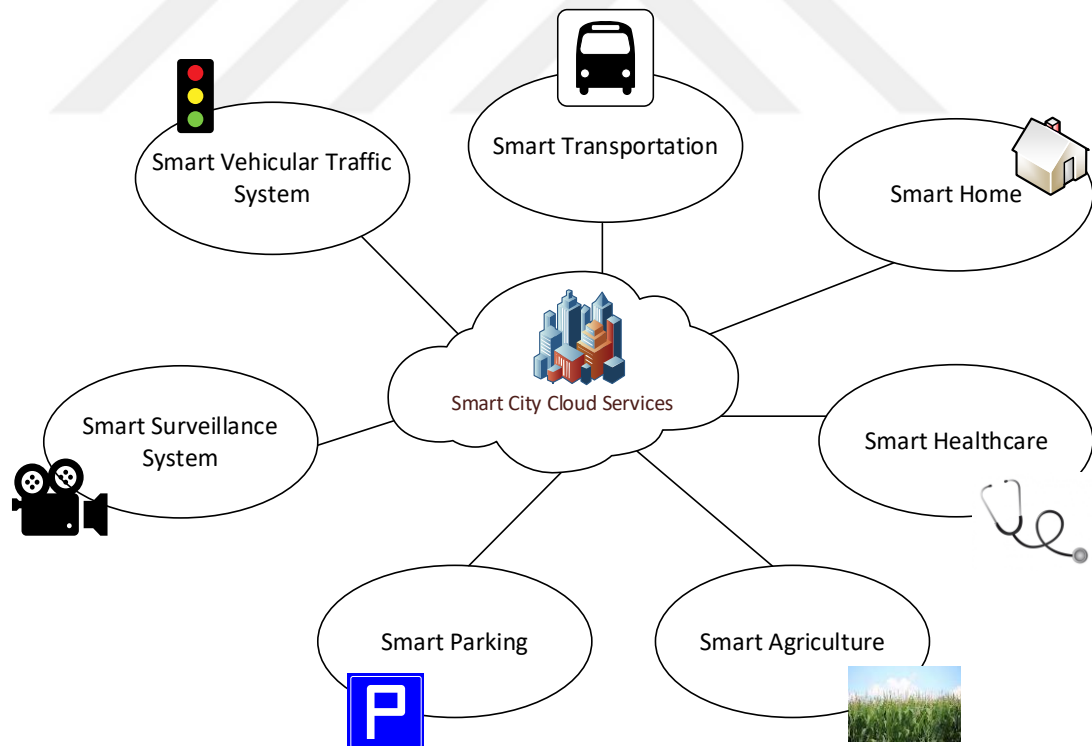


Figure 1.1 IoT based services in smart cities

In addition to the opportunities offered by technological innovations, there are some disadvantages. One of these disadvantages is ensuring data security. With the increase of wearable and mobile devices connected to the Internet in today's smart cities, the privacy and security of data is gaining importance. Data transfers should not be in a way that everyone can read but should be transferred in a way that confidentiality can be ensured. For this, security should be prioritized in IoT devices to be used in the technological infrastructure of smart cities. Thus, the quality of technology services received by the citizens living in smart cities will also increase and the security of personal data in IoT devices used in smart cities will be protected. In recent years, there are many studies on the security of IoT technologies used in smart cities (Chakrabarty & Engels, 2016; Malche, Maheshwary & Kumar, 2019; Reddy, Suresh, Phaneendra, Shin & Odelu, 2018).

1.3 Internet of Things (IoT)

The concept of IoT was invented by Kevin Ashton in 1999 to support the RFID concept involving sensors (Tawalbeh, Muheidat, Tawalbeh & Quwaider, 2020). The concept of IoT refers to many objects that connect to the internet, such as smart watch, smart home applications, smart phone, and so on. With the increasing use of technology in many areas such as health, transportation, and education, the concept of IoT is increasing its popularity. It is known that the number of devices connected to the internet is 7 billion in 2018 and it is thought that this number will be 19.8 billion in 2023 (Sisodia, 2020).

IoT devices are not very expensive hardware, have low resources, and generally do not have security mechanisms (Alladi, Chamola, Sikdar & Choo, 2020). The security of IoT devices, which consume less power and have fewer resources than computers, should be considered as much as the security of computers. Since IoT devices are widely used especially in smart cities, they can be located at critical points. For example, when there is a problem with security vulnerabilities in devices used in the healthcare field, even the slowdown of IoT device-based services can have very

important consequences. For this reason, we can talk about a more secure IoT network by taking the necessary precautions as much as possible.

1.4 Internet of Things Applications for Smart Cities

The effect of IoT technologies in the formation of smart cities is gaining great importance. Many different infrastructure services in the city provide services to the citizens by connecting to the internet via IoT devices. People living in smart cities benefit from the offered IoT device technologies by using wireless and mobile devices. Thus, it is possible to access many data from different locations and at desired times. IoT applications used in smart cities are described below.

1.4.1 *Smart Parking*

Managing parking lots is an important factor in regulating the vehicle density of smart cities. Analyzing the vehicle density and stopping data on the streets, giving the closest and appropriate parking lot information to the vehicle users will help in regulating the urban traffic. By monitoring the car parks in smart cities in a central structure with the automation system and presenting these data to the citizens, the car parks in the city can be used more efficiently. The fact that vehicle users can quickly and easily find the parking lots in the city that are suitable for them through their mobile devices will reduce the driving time of the vehicle user, and accordingly, the traffic density will decrease.

1.4.2 *Smart Homes*

Electronic equipment used in smart homes can be monitored and managed remotely with the help of sensors. Heating/cooling, blinds, lighting, and similar systems inside the house can be controlled via the internet with the help of sensors and other IoT

devices. For example, the air conditioning of the house can be done autonomously with the temperature values determined by the user. Many devices can be managed with different sensors to be connected to the central control panel at home. Thus, through mobile devices, IoT devices in the house can be managed from anywhere and remotely intervened when necessary. By calculating the energy consumption of electrical devices through IoT sensors, the operating times of electrical devices in the house can be determined and it can contribute to energy savings.

1.4.3 Smart Grid

Smart grid provides smart management of energy distribution and consumption. With the increase in energy need day by day, the use of smart grid technologies is also increasing. With the use of IoT technologies in this area, monitoring, detection, and intervention of faults become easier. At the same time, more comprehensive information about the energy they use, and energy consumption tariffs can be offered to consumers.

1.4.4 Smart Vehicular Traffic System

Vehicle traffic density, which is one of the important problems of cities, can be monitored with cameras and sensors. Information such as traffic lights and street traffic density data in the city can be collected in a central structure with cloud computing technology and analyzed and solutions can be presented with decision support systems. Citizens can have the chance to determine the appropriate roads to go by obtaining the urban road status information.

1.4.5 Smart Transportation

The transportation services offered in smart cities may become more efficient with mobile applications and IoT technologies (Dalkılıç, 2015). The transportation services

data, which are detected by sensors and analyzed with the central cloud structure, are made available to the citizens, enabling them to benefit from transportation services more efficiently. By acquiring urban transportation data with their mobile devices, citizens will be able to see which transportation route would be more efficient for them and save time.

1.4.6 Smart Energy System

The search for clean energy sources is increasing day by day. As a result of this search, renewable energy sources gain importance. Today, the most common renewable energy sources are solar energy and wind energy. Solar energy is obtained by converting sunlight into electricity. Energy collection time varies according to regions and sunlight intensity. It is important to get the maximum benefit from solar energy in a limited time.

By using sensor technologies, the intensity of sun rays can be determined according to regions and time, and more efficient energy can be obtained. To benefit more from wind energy, wind direction and strength can be detected by sensors, and wind energy can be used in suitable areas. With the contribution of IoT technology, the efficiency of energy resources in different locations can be measured and resources can be used more efficiently.

1.4.7 Smart Waste Management

Waste management contributes to improving the sustainability of a city. The integration of the latest technological innovations into the existing waste management system contributes to the provision of smart waste management solutions in smart cities.

By using developing sensor technologies, it is possible to separate more easily according to waste types. In addition, the occupancy rates of the trash cans can be determined remotely via sensors, and the optimal determination of the trash bin cleaning frequency can be achieved. Thus, by creating smart trash cans, it contributes to the smart waste management of smart cities. By using IoT technologies, waste management can be done quickly and efficiently, and environmental pollution can be prevented.

1.4.8 Smart Health

Central automation systems are used to provide faster and more efficient health services in smart cities. For example, by controlling all the emergency services in the city from a central location, the possible densities can be balanced. Floods, disasters, and similar situations can be detected quickly with IoT technologies and the response time to incidents can be shortened. Elderly people need to receive fast service in case of health problems that may occur. IoT technologies to be used in smart cities can be of great help when elderly people need urgent access to health services, such as falling (Dalkılıç, 2016).

In addition, some health data of people can be obtained with wearable smart textile devices. For example, in wearable smart textiles, body movements of people can be detected by using accelerometer and gyro sensors (such as walking, running, sitting). In addition, with the pressure sensors integrated into the socks, the amount of pressure applied by the people while they step on the ground can be measured. With the obtained foot pressure data, some diseases can be detected in advance and measures can be taken (Lin & Seet, 2016). A wearable heart monitoring design has been proposed by adding inductive sensors to the fabric of the clothes (Brezulianu et al., 2019). As a result of this study, the evaluation of the heart data rate was provided.

Health data of people can be obtained with mobile phones and smart watches. By means of smart watches, the heart rate can be measured and the user can be informed.

By synchronizing the smart watches with the mobile phone, the data obtained with the health of the user is analyzed. Wireless technologies such as mobile devices and wireless sensors on the person communicate with each other and become a part of the wireless body area network (WBAN). When a structure including more comprehensive equipment is formed, it will become a part of the personal area network (PAN).

1.5 Motivation

The motivation for writing this thesis is given below.

- The increasing use of IoT devices used in smart cities and, accordingly, the importance of the security of personal data contained in these devices.
- Due to the limited resources of IoT devices, security measures to be taken should consume fewer resources than computers.
- Security measures must be compatible with new technological devices.
- The need for IoT devices used in areas such as transportation, health, and education to be used in different areas day by day and the increase in security problems that may arise.
- To enable people living in smart cities to use the IoT-based services they receive more securely.

1.6 The Goal of the Thesis

One of the aims of this thesis is to provide secure data communication of IoT devices, which are increasingly used in smart cities. Smart watches, mobile devices, and wearable devices containing IoT technologies can contain personal data. This data is transferred wirelessly between IoT devices, and personal information is constantly shared between devices. These information shares, which often contain personal data, are transmitted between devices in unencrypted and clear text.

Our second goal is to ensure that the security protocol to be implemented works efficiently and with performance on IoT devices with limited resources. With this work, we want to ensure the secure transfer of wirelessly transmitted data between IoT devices. With the developing technology, the size of IoT devices is getting smaller and they need less energy consumption. The security protocols to be used should also provide fast and performance services to users by using the resources of IoT devices at a minimum level.

Our third goal is to introduce a strong mutual authentication protocol. We proposed a five-step mutual authentication protocol (FSMAP) to open the way to secure wireless data transmission between IoT devices. The FSMAP was simulated through the smart textile device we prototyped and the Windows application we developed. We also analyzed the authentication protocol by coding it for Automated Validation of Internet Security Protocols and Applications (AVISPA) and Scyther tools.

1.7 Thesis Organization

This thesis consists of five chapters. In this chapter, the definition of smart cities, the concept of the IoT, IoT applications in smart cities, our motivation, and the purpose of the thesis are mentioned. The rest of the thesis is organized as follows.

Chapter 2 explains the architecture of embedded systems, architecture of IoT, architecture of wearable IoT device, IoT devices functionality layers, IoT devices security requirements, encryption methods, AES encryption algorithm, types of attacks in IoT and literature survey on security of IoT devices.

In Chapter 3, the stages of creating the smart textile hardware used in the thesis, vulnerabilities in IoT, the materials contained in the smart textile hardware, and the software in which the Arduino UNO microcontroller is programmed are presented.

Chapter 4 explains the methods used in this thesis study, performance evaluation of FS MAP on IoT device, testing platform, and testbed with Windows application we developed.

In Chapter 5, informal analysis, analysis with AVISPA and Scyther formal verification tool, and performance and security comparison with previous works are given.

In the last chapter, the conclusion and future works are mentioned.



CHAPTER 2

BACKGROUND AND RELATED WORKS

2.1 Architecture of Embedded Systems

The embedded system, which forms the basis of IoT technologies, consists of two parts. These are embedded hardware and embedded software. Embedded hardware includes components such as I/O, controller, memory, while embedded software includes embedded operating system, applications, and drivers. Basically, Harvard architecture and Von Neumann architecture are used in embedded systems. Figure 2.1 shows Vonn Neumann and Harvard architecture.

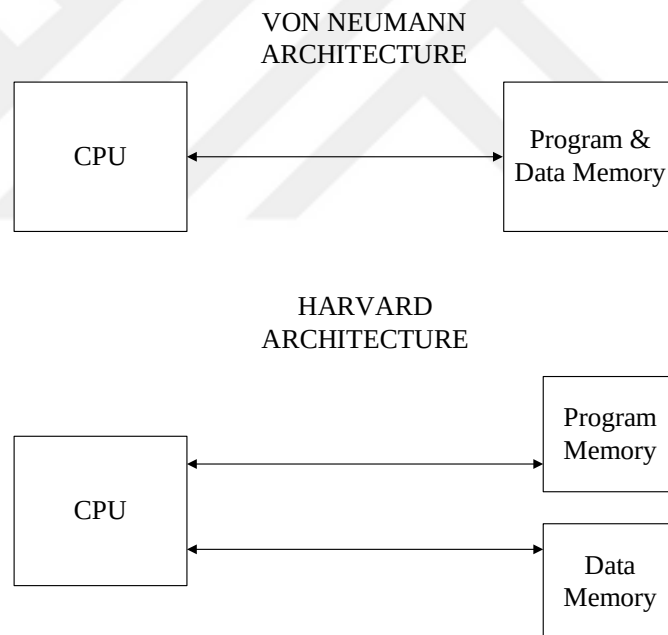


Figure 2.1 Vonn Neumann & Harvard architecture

There are some differences between the two architectures. In Vonn Neumann architecture there is a single bus connection to the CPU for Program and data memory. In Vonn Neumann architecture, system performance is affected as program memory and data memory cannot be accessed at the same time. However, the Vonn Neumann architecture requires less hardware in this respect.

In the Harvard architecture, there are separate data paths between the CPU and program memory and data memory. Therefore, the Harvard architecture requires more hardware. Since the processor can receive data at the same time, it works more efficiently. Harvard architecture was developed to overcome the bottleneck in Vonn Neumann architecture.

Figure 2.2 shows the basic architecture of embedded systems. Embedded systems consist of software and hardware parts. In the software section, there are various applications and embedded operating system. In the hardware section, there are embedded microprocessor, peripheral device, bus, memory and various hardware. The embedded system can be accessed with a keyboard and similar input hardware. Output can be in the form of electricity, sound and so on.

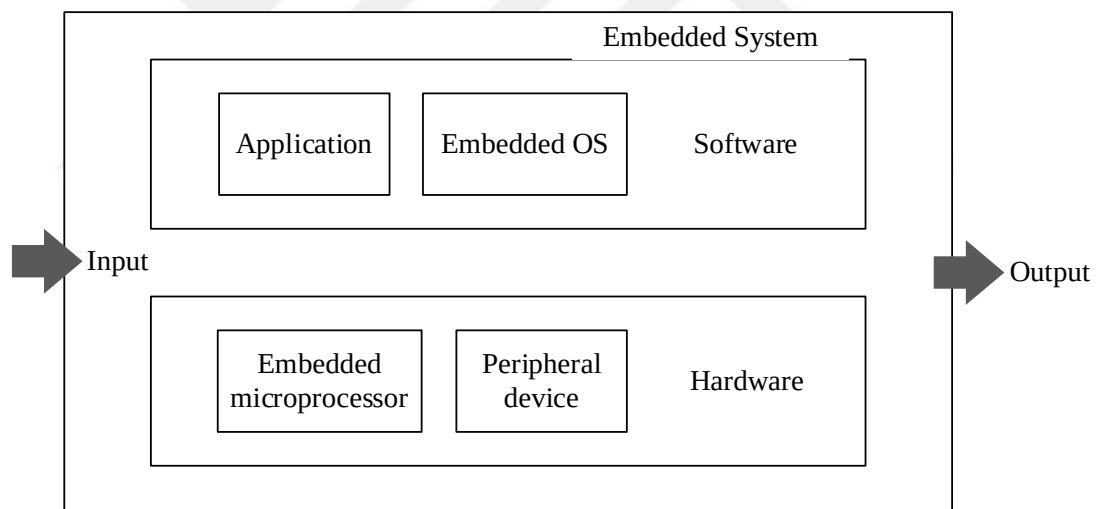


Figure 2.2 Basic architecture of embedded systems

IoT devices are a subset of embedded devices. Figure 2.3 shows that IoT, which is a subset of embedded systems, consists of 4 main components: sensors & actuators, local network, Internet and back-end services.

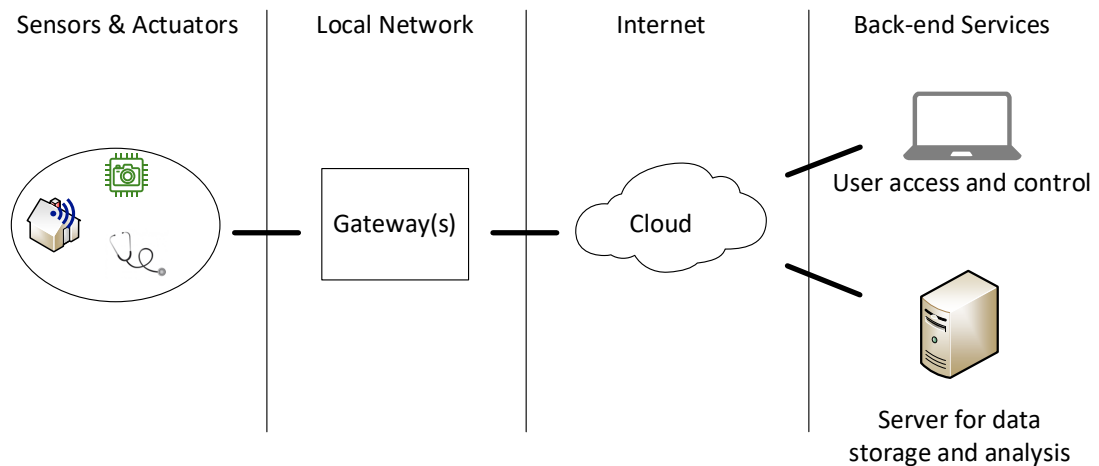


Figure 2.3 Embedded system subset (Hammoudi, Aliouat & Harous, 2018)

The data obtained by the sensors is first transferred to the gateway in the local network, and then transferred to the cloud system on the internet. These data, hosted and analyzed on the server, provide back-end services to users with user access and control. Likewise, when it is desired to control physical changes with the actuator, access to IoT devices is provided via cloud and gateway by using back-end services.

2.2 Architecture of IoT

IoT is an ecosystem where devices connect to the internet and allow the user to control these devices from anywhere via the internet. Figure 2.4 shows the IoT architecture.

The parts of the IoT architecture shown in Figure 2.4 are described below.

- **Sensor & actuators:** The sensor is a hardware that detects physical changes in its environment and converts it into electrical signals. For example, the motion sensor detects the motion around it and converts it into electrical signals. The actuator, on the other hand, works on the opposite principle. The actuator can physically make changes when it detects electrical signals.

For example, an electric valve enables the valve to open and close according to the electrical signals it receives. Thus, the water flow through the valve can be controlled electrically. Sensors and actuators only work at the physical level or in the IoT application domain.

- **Controller:** Controllers are used to control the actuators and sensors in the IoT system. Since actuators and sensors are analog devices, the task of the controller is analog to digital or digital to analog conversion.
- **Processor:** Processor collects, organizes and processes all data. It collects and processes the data coming from the sensors as digital signals through the controller.
- **Gateways:** It is used to transmit the data of the IoT system over the internet. It performs modulation and demodulation for data transmission. Protocols such as MQTT (Message Queuing Telemetry Transport) and Extensible Messaging and Presence Protocol (XMPP) are used in IoT systems. An example of a gateway device is a modem.
- **Cloud Server:** Cloud server is the area where IoT data is stored. The data received by the IoT system through sensors are collected in a central structure on the cloud server, so that the sensor data in different locations are on the same server. Likewise, users in different locations can access IoT data by connecting to the cloud server from anywhere.
- **User Devices:** They are devices used to control the status of devices in the IoT system and to manage devices. For example, IoT data on the cloud server can be accessed via a mobile device. Likewise, the sensor data on smart textiles can be accessed wirelessly with the help of a smart watch.

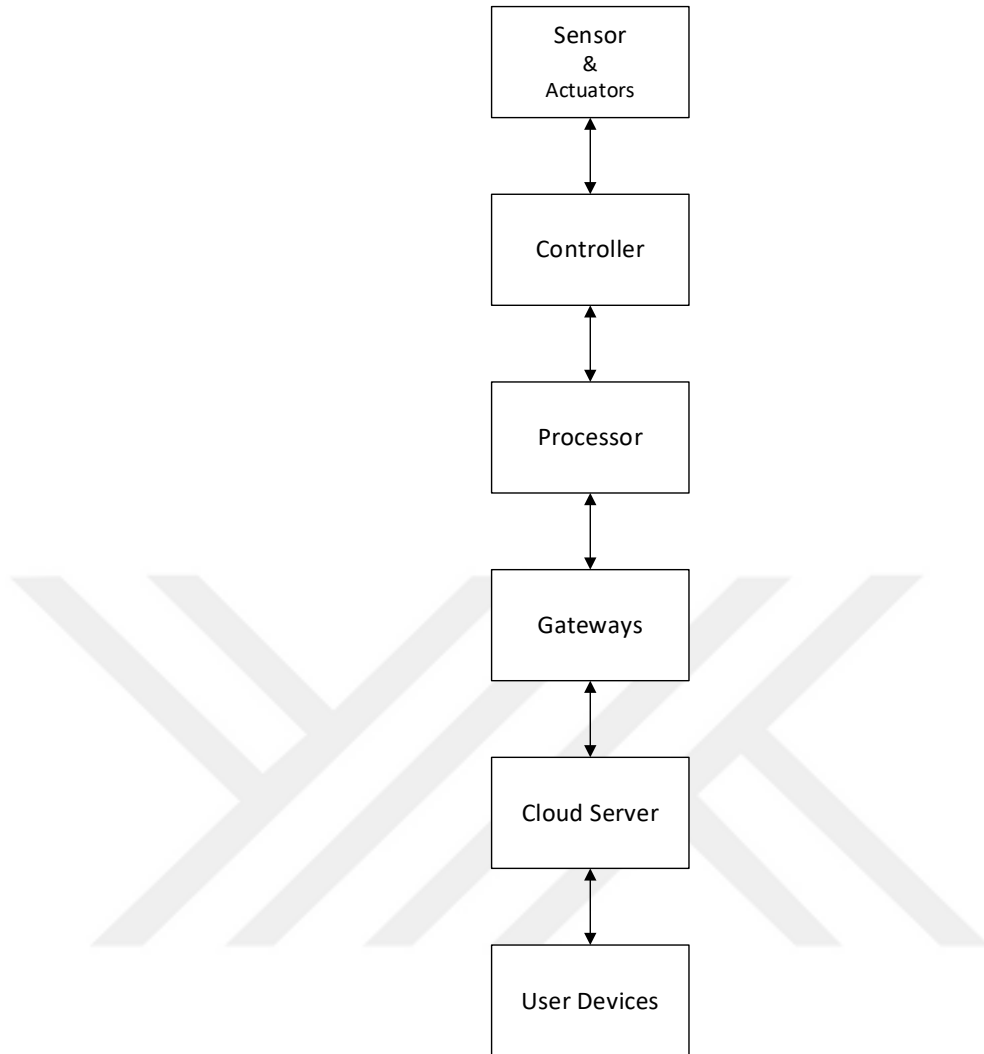


Figure 2.4 IoT architecture (ETechnoG, 2022)

2.3 Architecture of Wearable IoT Device

The wearable IoT device architecture, which is created by integrating sensors on smart textiles, is shown in Figure 2.5.

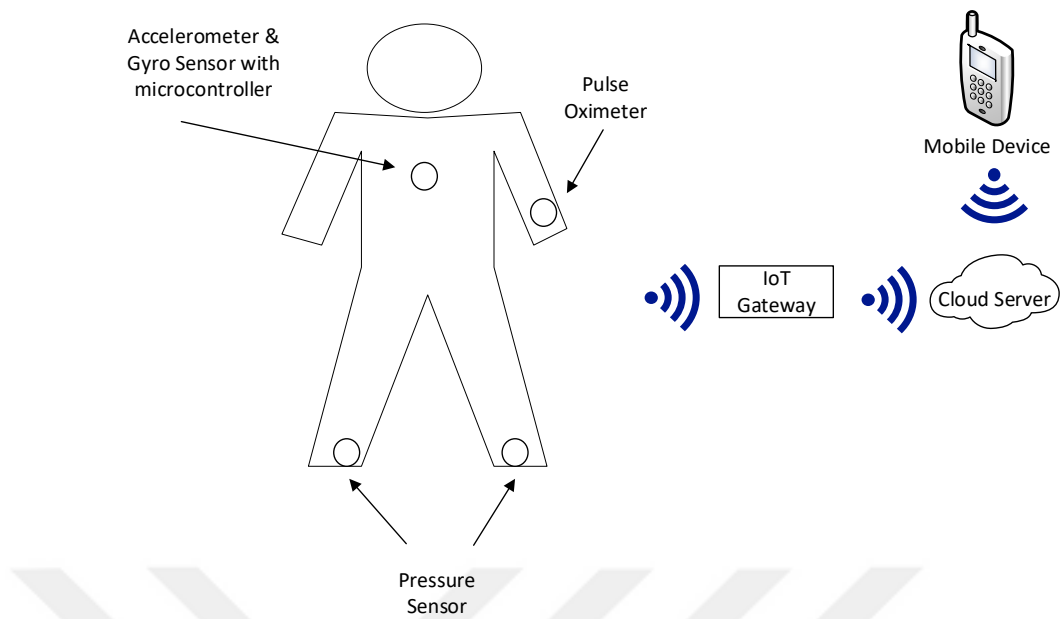


Figure 2.5 Wearable IoT device architecture

As shown in Figure 2.5, data is collected by the sensors (Pressure sensor, pulse oximeter, accelerometer & gyro) on the people and transferred to the IoT gateway wirelessly. These sensors can be used by integrating on smart textiles or through hardware such as smart watches. Microcontroller cards integrated on smart textiles can be used for wireless transmission of data in sensors.

The data transferred to the IoT gateway is transferred to the cloud server wirelessly and stored there. Users can access the data on the cloud server via the internet from anywhere using their mobile devices.

2.4 IoT Devices Functionality Layers

In the literature research, there are different layer architecture models proposed for IoT. These differences are due to different IoT technologies (Taş & Kiani, 2020). The model, which is accepted as general-purpose among the layer architecture models of

IoT devices, consists of 3 layers (Efe, Aksöz, Hanecioğlu & Yalman, 2018). Another type of architecture consists of 5 layers.

The network structure model, which is known as the OSI (open systems interconnection) model and consists of 7 layers, and the layers in IoT devices are similar, but they are not the same. Three-layer architecture and five-layer architecture are described below.

2.4.1 Three-Layer Architecture

Figure 2.6 shows the three-layer architectural model used for IoT devices. The three-layer architecture is one of the first architectures that emerged with the concept of IoT.

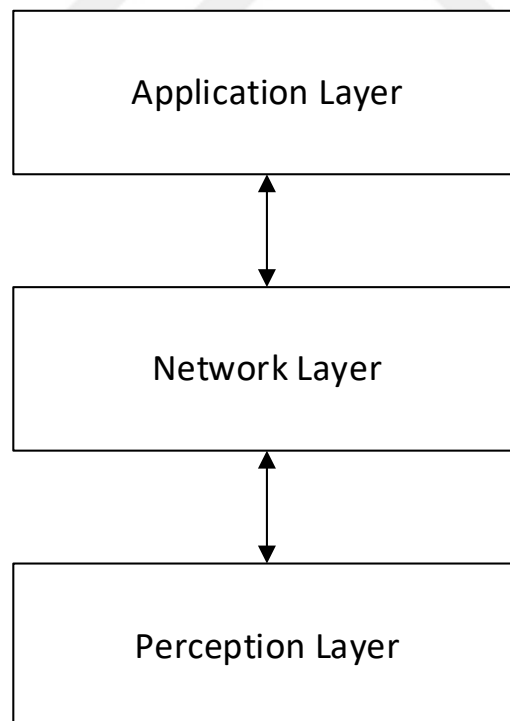


Figure 2.6 Three-layer architecture

Each layer in the three-layer architecture model is described below.

2.4.1.1 Perception Layer

The task of this layer, also known as the sensor layer, is to convert the information in the package it collects into digital signals and to recognize objects (Park et al., 2019). Protocols such as RFID, NFC, and Zigbee are used in this layer (Yang et al., 2011). There are various types of sensors used to collect data, such as temperature, humidity, motion, and vibration. There may be attackers who want to replace a sensor in IoT networks with their own, which can pose a threat to IoT networks. Tampering, jamming, and replay attacks are made to this layer.

2.4.1.2 Network Layer

The layer where data processing and data transmission takes place is the network layer (Çavdar & Öztürk, 2018). It processes the data from the Perception layer and transmits it to the upper layer. Protocols such as IPv6 over low-power wireless personal area networks (6LoWPAN), long-range wide area network (LoRaWAN), IPv6, and ICMP are used in this layer. Also, this layer uses RFID, 3G, WIFI, and other network technologies.

6LoWPAN technology is used for wireless sensors to access the internet network. 6LoWPAN includes IEEE 802.15.4 technology and IPv6 technology. 6LoWPANs encounter the same security attacks as traditional networks, including attacks such as DoS, user impersonation, man-in-the-middle, replay. Due to the limited resources of 6LoWPANs, it becomes difficult to ensure that they are in a secure structure.

LoRaWAN is a communication protocol designed for low-power and low-cost devices. LoRaWAN architecture consists of end nodes, gateways, network server, and

application server. End nodes, which are usually IoT devices, transmit the information they collect from the physical environment to the gateways via the LoRa physical layer. This information is transmitted by the gateways to the network server. This communication can be carried out using different technologies such as Ethernet, Wi-Fi, and satellite. The data is then transmitted to the application server.

LoRaWAN uses a 128-bit encryption key for encryption. In LoRaWAN, thousands of end devices can be connected to internet servers through a single gateway. LoRaWAN allows long-range communication.

Because this layer connects devices and other network components, it is vulnerable to attacks. DDoS/DoS and man-in-the-middle attacks are made to this layer.

2.4.1.3 *Application Layer*

The application layer is a layer that interacts directly with the end-user. In this layer, the user receives and uses the service from the service provider (Ferdows, Mehedi, Hossain, Shamim & Rasiq, 2020). The application layer is the last layer. MQTT and CoAP are protocols used in this layer.

MQTT is a machine-to-machine (M2M) message-based communication protocol. MQTT is a widely used lightweight protocol for resource-constrained IoT devices. MQTT works over TCP (transmission control protocol). In communication using the MQTT protocol, there is a manager called a broker that controls the traffic. The message broadcast is called publish, and those who subscribe to this message broadcast are called subscribe. Figure 2.7 shows the operation of the MQTT protocol.

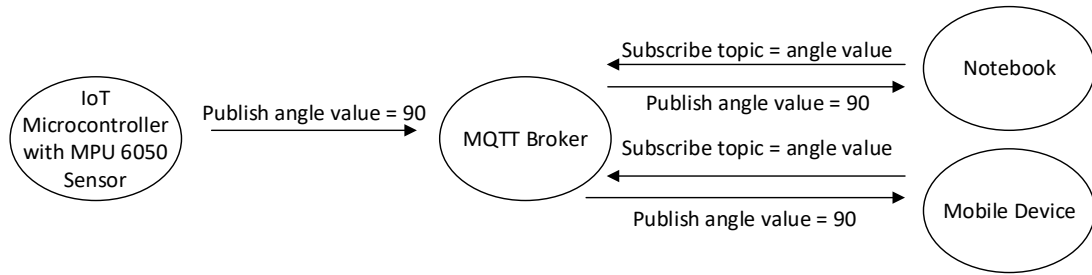


Figure 2.7 MQTT protocol

The data between the publisher and the subscriber of the message is transmitted asynchronously. The MQTT protocol does not require the publisher and subscriber to be connected at the same time. In the example communication shown in Figure 2.7, the data sent by the IoT sensor (publish) is transmitted to the MQTT broker. MQTT broker transmits this data to subscribers. The broker uses the topic to send the message to the subscribers. It is possible to transmit the same message to more than one subscriber with high performance. Because the MQTT protocol requires less resource consumption, it is highly preferred in IoT devices.

The CoAP (constrained application protocol) protocol is designed for use in M2M communication. CoAP was developed to work with HTTP (Naik, 2017). CoAP works over UDP (user datagram protocol). Therefore, SSL/TSL security cannot be provided. The operation of the CoAP protocol is based on the client/server model as seen in Figure 2.8. The client is the source that produces the data and transmits the data it produces to the server.

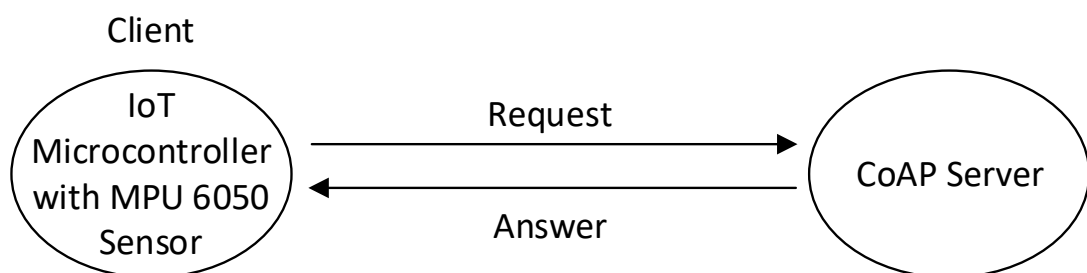


Figure 2.8 CoAP protocol

There are security threats that may arise in this layer as well. Malicious code injection attack is made to this layer.

Security attacks against IoT devices are described in detail in chapter five.

2.4.2 Five-Layer Architecture

Figure 2.9 shows the five-layer architectural model used for the IoT.

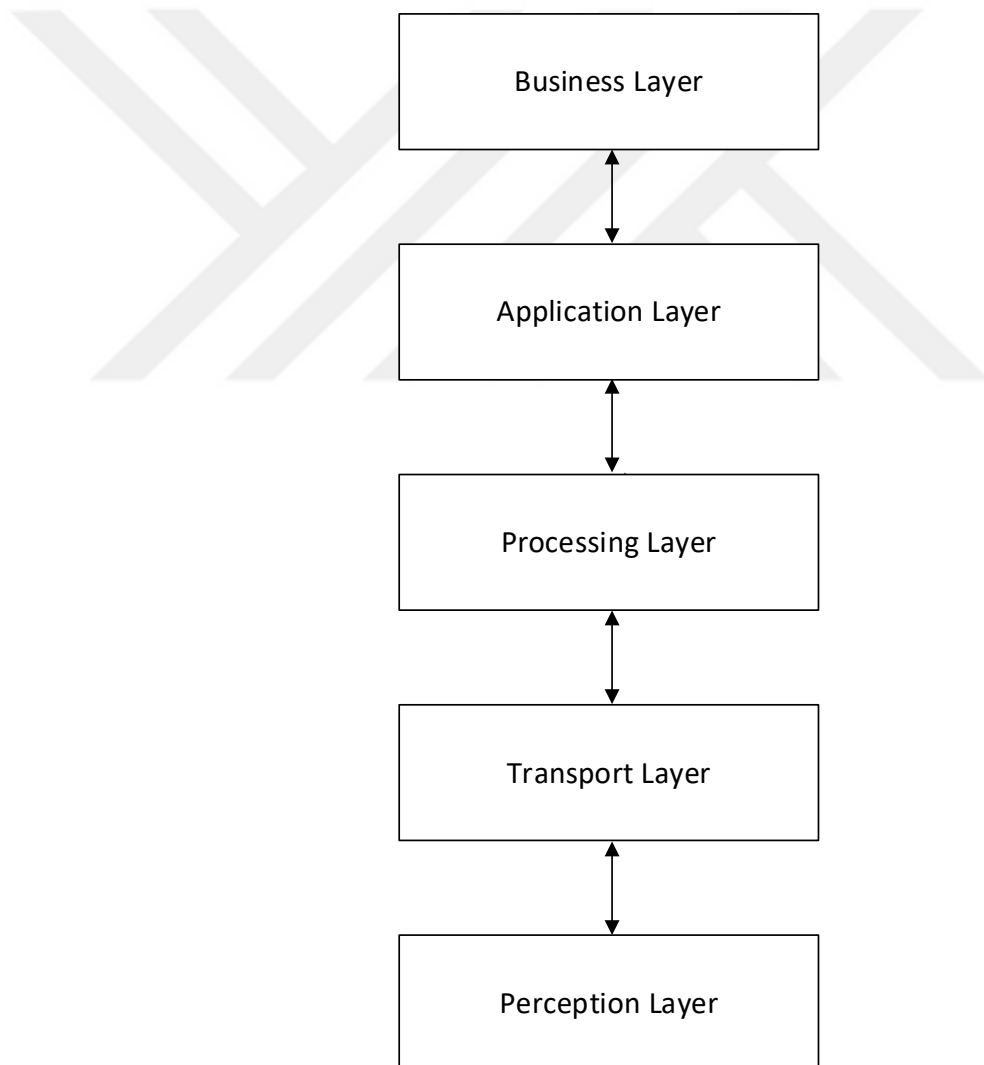


Figure 2.9 Five-layer architecture

The application, transport (network layer) and perception layers in the five-layer architecture perform the same function as in the three-layer architecture. Other layers are described below.

2.4.2.1 Processing Layer

The processing layer is also called the middle-ware layer. This layer is responsible for processing, analyzing, and storing data according to the calculation results. The processing layer is used to remove useless information from the transport layer.

2.4.2.2 Business Layer

The business layer is the manager of the IoT. It manages applications and business models. Makes decisions about road maps.

2.5 IoT Devices Security Requirements

In IoT environments, the security requirements described below must be met for IoT devices to provide secure services by reducing the effects of cyber attacks (Islam, Kwak, Kabir, Hossain & Kwak, 2015).

2.5.1 Confidentiality

The concept of security, which aims to prevent unauthorized access to the data transmitted over the network, is called confidentiality. There are different ways to ensure that data is confidential. Confidentiality of data can be ensured by taking some precautions. One of the ways to ensure the confidentiality condition is to transmit the

data encrypted. Data confidentiality is ensured since the content of the data cannot be accessed in case of unauthorized access to the encrypted data.

2.5.2 Integrity

Transmission of data to the other party without interruption and corruption is defined as integrity (Ceyhan & Sağıroğlu, 2013). Integrity ensures the integrity of the data. Integrity cannot be provided in cases where data is changed by unauthorized persons and data is lost. When data is created, transmitted, used, it should be preserved and in integrity at every moment of its life cycle.

2.5.3 Availability

It is the provision of accessibility without delay and rejection when data needs to be accessed (Nivetha & Shyla, 2021). IoT devices need to be able to reach the IoT gateway when they need it. Availability is necessary for the IoT gateway to serve other IoT devices efficiently and with performance. To provide availability, the network traffic density that may occur should be prevented by minimizing the attacks on the IoT gateway.

The CIA triad showing confidentiality, integrity, and availability is indicated in Figure 2.10.

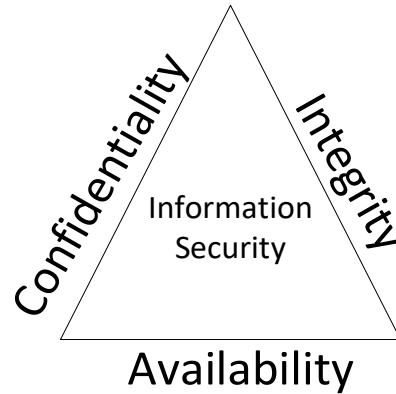


Figure 2.10 CIA triad

2.5.4 Authentication

Authentication is considered a basic requirement for IoT. Trusting the devices participating in the IoT network is important for the operation of the network. When IoT devices want to communicate with each other, they must be introducing themselves to the other device. Authentication should be the first step for IoT devices to securely communicate with each other. If any IoT device in the IoT network is accessed by malicious users, it can cause the entire network to crash. Traditional authentication schemes are not suitable for resource-constrained IoT devices. Therefore, with the emergence of lightweight authentication schemes, suitable solutions are provided for IoT.

2.5.5 Authorization

Authorization is the process of granting, limiting, or denying access to resources, applications, or data in the IoT environment. Only authorized persons and devices can access relevant resources, applications, and data. Each user and device should be given access authorization separately, and which resources, applications, and data they can access should be defined in advance. In this way, transactions are made according to

the level of authority. Unauthorized users and devices are prevented from entering the IoT environment.

2.5.6 Data Freshness

Data freshness means the freshness of the data and the freshness of the key. The data received from the sensors in the IoT network can be updated continuously, for example, many updated data are transferred to mobile devices via smart textiles and personal clothes are evaluated. Therefore, the freshness of the data becomes important. It is necessary to know that the transferred data is up-to-date. Data freshness indicates that each dataset is new, meaning that an unauthorized user cannot replay old messages.

2.5.7 Non-Repudiation

Data communication between IoT devices may not be completed due to errors that may occur. Non-repudiation means that IoT devices cannot refuse to send the previously sent message.

2.6 Encryption Methods

The proposed algorithm methods for IoT networks are of two types, symmetric and asymmetric.

The text (plaintext) to be sent by the sender to the receiver is encrypted with the secret key and ciphertext is obtained. The ciphertext in the sender is transmitted to the receiver side in encrypted form. The receiver, on the other hand, accesses the contents of the ciphertext using the secret key. Symmetric algorithms use a single key (secret key) for decryption and encryption. Symmetric algorithms are more powerful when

secret key exchange between devices (sender and receiver) is done securely. Figure 2.11 shows how the symmetric encryption algorithm works.

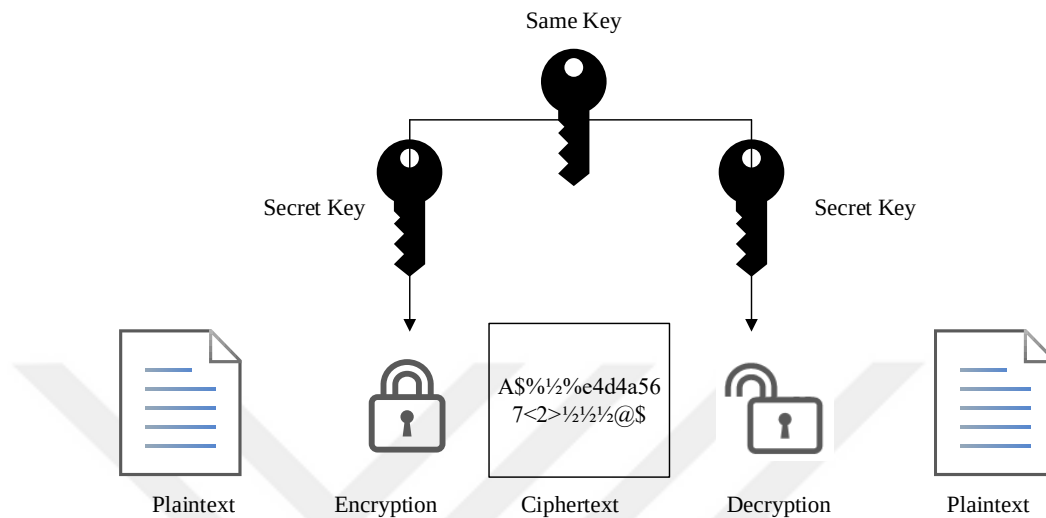


Figure 2.11 Illustration of symmetric encryption

Asymmetric algorithms have two keys, public and private. The public key is sent to the receiver over the network. The private key is secure as it is not transmitted over the network. During encryption, the plaintext is encrypted by the sender with the receiver's public key, and the encrypted text is sent to the receiver over the network. If the public key is known by someone else, the ciphertext cannot be read because there is no secret key. During decryption, the receiver will decrypt the ciphertext using the private key. Asymmetric algorithms consume more resources because they are more complex than symmetric algorithms. Figure 2.12 shows how the asymmetric encryption algorithm works.

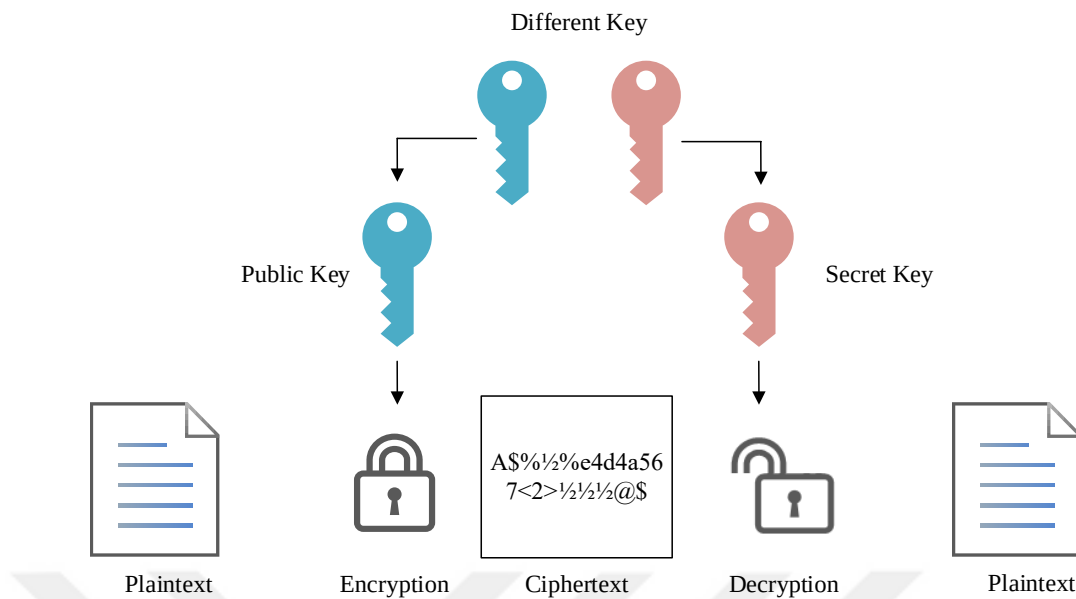


Figure 2.12 Illustration of asymmetric encryption

Since asymmetric algorithms consume more resources, Symmetric algorithms are used in IoT device applications with limited resources. Symmetric algorithms are secure as long as key confidentiality is maintained.

2.7 AES (Advanced Encryption Standard) Encryption Algorithm

AES is an encryption algorithm based on Substitution-Permutation. It was developed by Vincent Rijmen and Joan Daemen. It was created on the basis of the Rijndael algorithm. It provides secure data transmission by improving the deficiencies and weaknesses of the DES algorithm, which was used for electronic data encryption before it. The AES encryption algorithm is an algorithm in which keys are symmetrically related for both encryption and decryption. The encryption and decryption key used in the algorithm is the same. Both hardware and software performance of the AES algorithm is high.

The data to be encrypted by AES has a length of 128 bits. There are 3 different options for the key length to be used to encrypt this data. These are 128-bit, 192-bit, 256-bit. As the key length increases, the security level also increases, and at the same time, performance and speed decrease. There are 10 rounds of encryption for 128-bit encryption, 12 rounds for 192-bit encryption, and 14 rounds of encryption for 256-bit encryption. After the data to be transmitted is made an unreadable data set by undergoing the necessary encryption processes, it is transmitted to the receiver. After receiving the data, the receiving side needs the key to access its content. This key is the same as the key with which the data is encrypted. When the steps applied in the encryption process are reversed, the receiver can access the content of the data.

AES encrypts on a 128-bit long 4x4 column-priority byte matrix structure. The encryption process is done by following certain steps and repeating them in a certain cycle, as shown in Figure 2.10. First, the key required to encrypt the data is generated. Afterward, the data to be encrypted is converted into a 4x4 state matrix. The data converted to the matrix is XORed with the generated key. The processes up to this stage are done once. Afterward, the following operations are performed as specified in the encryption round section in Figure 2.13.

- Sub Bytes: The matrix obtained in the first step is replaced according to a predefined table named “Substitution Box”.
- Shift Rows: Circularly shifted rows in a certain number.
- Mix Columns: The 4-byte values in the column are mixed with each other.
- Add Round Key: The previously obtained matrix and key matrix elements are XORed.

The matrix obtained as a result of the operations done so far is used as an input in the next step. At the same time, a new key is produced at each new stage. The operations described above are repeated 10, 12, or 14 times depending on the AES bit encryption length. Afterward, transactions in the last round section specified for

encryption in Figure 2.13 are performed. The difference between the transactions made in the last round and the encryption round is that there are no “mix columns”. With the last round of the Encryption section, the ciphertext is obtained, and encrypted data is transmitted to the receiver. The receiver obtains the plaintext by doing the opposite of the above-described operations with the key he has.

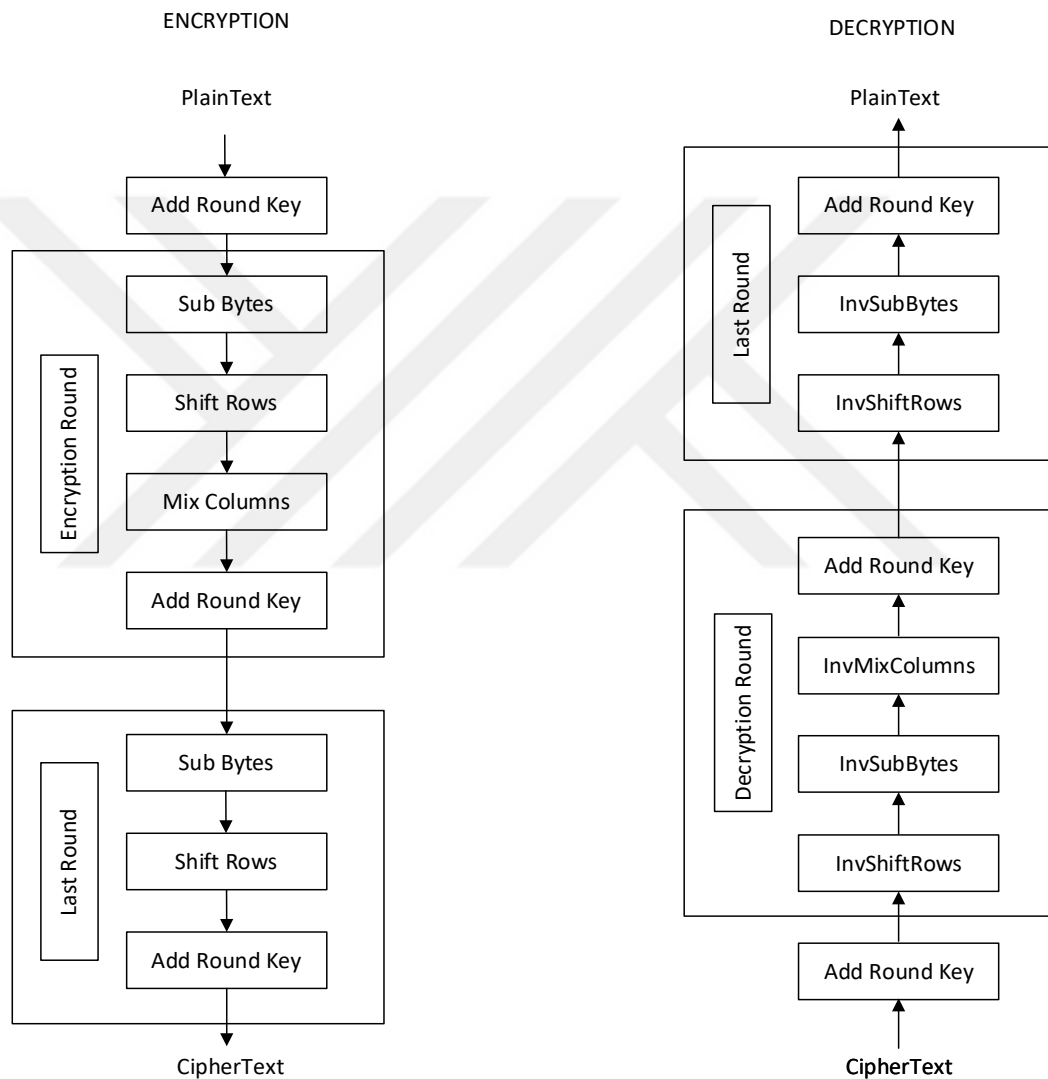


Figure 2.13 AES encryption and decryption process

The key matrix is important in the AES encryption algorithm. The key is known to the receiver and sender. An attacker who wants to obtain the data encrypted with the AES encryption algorithm must have the key. Since trying to find the AES key by trial

requires a very long time with today's technologies, the AES encryption algorithm is highly secure.

2.8 Types of Attacks in IoT

Attacks are generally divided into two passive and active attacks. In passive attacks, the information of the system can be learned and used without changing the system resources. In active attacks, data changes or incorrect data entry can be made into the system. Attacks on IoT networks are described below.

2.8.1 *Replay Attack*

Wireless transmission of data between IoT devices can cause it to be captured and used later by the attacker (Dalkılıç, 2014). Figure 2.14 shows that the data transmitted by the IoT sensor to the IoT gateway is sniffed by the attacker, and then this data is transmitted to the IoT gateway. To prevent this attack, it is necessary to have distinctive features that distinguish each session from others during authentication. Thus, since the information previously captured by the attacker is no longer valid, a replay attack will not be possible.

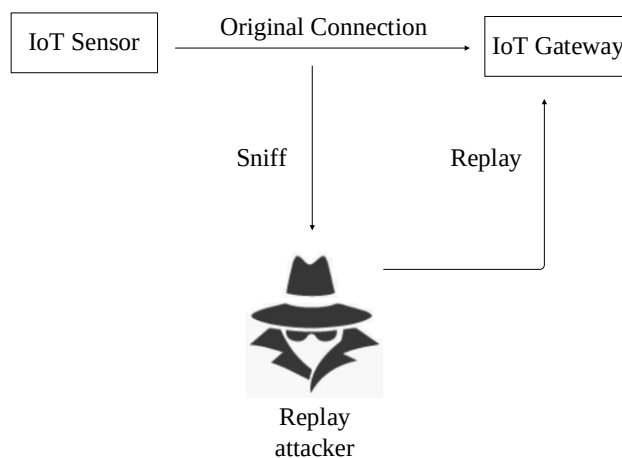


Figure 2.14 Replay attack demonstration

2.8.2 Denial of Service (DoS) Attack

Denial of service (DoS) is a type of attack that engages IoT devices with large volumes of network packets to render them inoperable (Baig, Sanguanpong, Firdous, Nguyen, & So-In, 2020). In the case of continuous communication request to IoT devices, system operation becomes inhibited. As seen in Figure 2.15, the attacker keeps the victim computer busy all the time. One of the ways to prevent this attack would be to limit an attacker's willingness to communicate, who wants to maintain a constant connection to a particular IoT device.

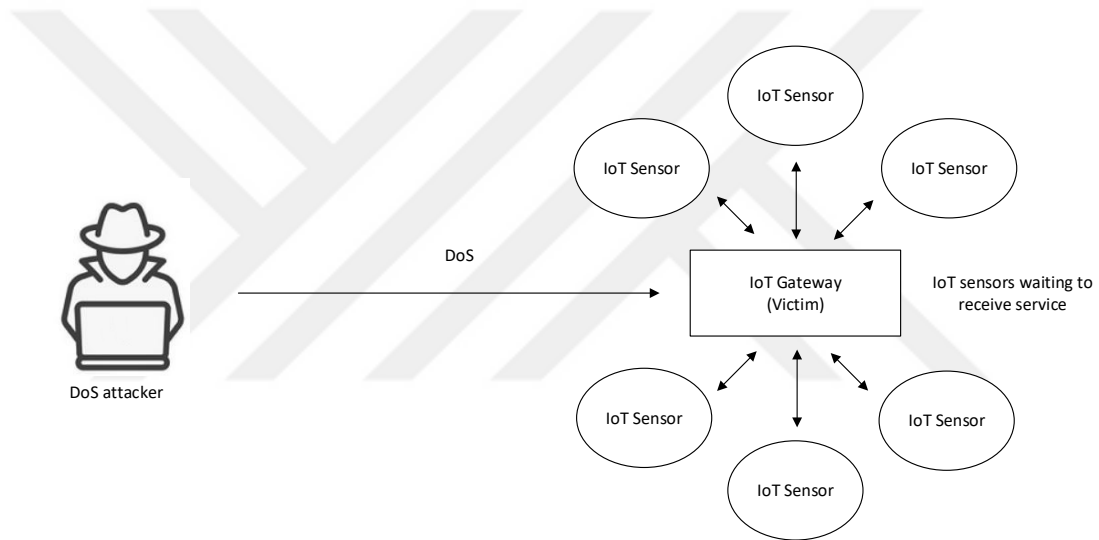


Figure 2.15 DoS attack demonstration

2.8.3 Man-in-the-Middle Attack

The man-in-the-middle attack is defined as changing the messages transmitted between IoT devices by listening and transmitting them to the receiver side as if they were sent from the real sender. As shown in Figure 2.16, the attacker cuts off the network communication between the two devices and communicates with both devices itself, and both devices can be attacked. Encryption is required to prevent an attacker from obtaining and modifying the message content.

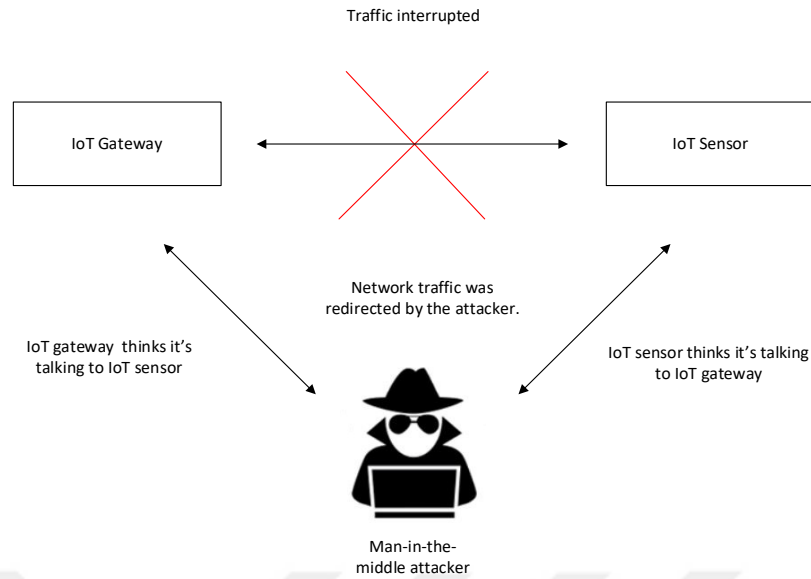


Figure 2.16 Man-in-the-middle attack demonstration

2.8.4 Full-disclosure Attack

It is a type of attack defined as the disclosure of the secrets of IoT devices (Özcanhan, Dalkılıç & Utku, 2014). It creates a security vulnerability as the content of the messages can be accessed because of the disclosure of secrets. Considering that people living in smart cities use IoT devices in many areas such as health, education, transportation, wearable devices, and so on, it is also known that the density of personal data stored in these devices is increasing. Accordingly, the security of the data hosted in IoT devices gains importance. To prevent this attack, the messages to be sent must be transmitted securely.

2.8.5 ID-theft Attack

An attacker's capture of the IoT device's ID information is defined as an ID-theft attack. The attacker may want to access the places where the IoT device has access by using the ID information he/she has obtained. With this attack, unauthorized people may interfere with the IoT network. This attack can be prevented by renewing the ID information every round.

2.8.6 De-synchronization Attack

Sequence numbers are used during communication to ensure that IoT devices match each other. By breaking the active connection between IoT devices, the attacker can send packets with fake sequence numbers or change the sequence number (Raymond & Midkiff, 2008). Thus, synchronization between IoT devices is broken. This attack can be prevented by ensuring that each step in the communication process relates to each other.

2.8.7 Identity Tracing Attack

Tracing IoT device identity leads to breach of device privacy. The attacker may want to use it later by tracking device IDs. This can cause security issues. Changing the device ID every round prevents this attack.

2.8.8 Impersonation Attack

It is a type of attack defined as an attacker impersonating any IoT device. Thus, it can access the authorized places of the IoT object it is impersonating. Using pseudo-ID and changing it in each round is one of the methods to prevent this attack.

2.8.9 Clone Attack

One of the methods that an attacker will use to gain access to the IoT gateway may be to want to appear as an existing sensor. For this, he may want to clone existing sensors. To clone a sensor, the attacker would need to obtain the sensor ID. Figure 2.17 shows the attacker getting the sensor ID by sniffing the IoT network. Changing the Sensor ID every round prevents this attack.

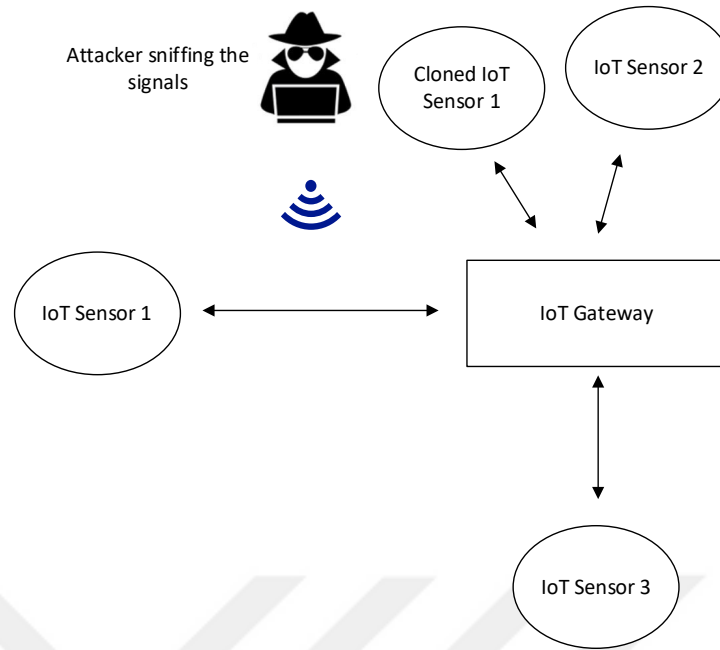


Figure 2.17 Clone attack demonstration

2.8.10 Node Tampering Attack

In cases where the attacker has physical access to IoT devices, it can change all or part of the hardware, as well as make a direct connection to change the information inside the device. Considering that personal data can also be found in IoT devices (such as wearable IoT devices, IoT healthcare applications) used in smart cities, this type of attack poses a serious risk. Encryption must be made so that the data stored in the IoT device cannot be read by the attackers.

2.8.11 Malicious Code Injection Attack

It is a type of attack that an attacker uses to gain unauthorized access to IoT devices by uploading malicious code. Usually, the firmware or software of IoT devices are updated wirelessly. This can allow attackers to inject malicious code into IoT devices. Attackers can perform unwanted functions by injecting malicious code into IoT devices. In some cases, it may be possible for them to access the entire IoT system. Figure 2.18 shows the injection of malware codes into IoT sensor devices. In the first case, the Attacker injects malicious codes into any device in the IoT network.

reduces the success rate of this method. In wireless communication systems, jamming attacks can be prevented by changing the communication mode. It is an important attack type for IoT devices that use wireless networks heavily. Figure 2.19 shows the jamming attack on IoT devices.

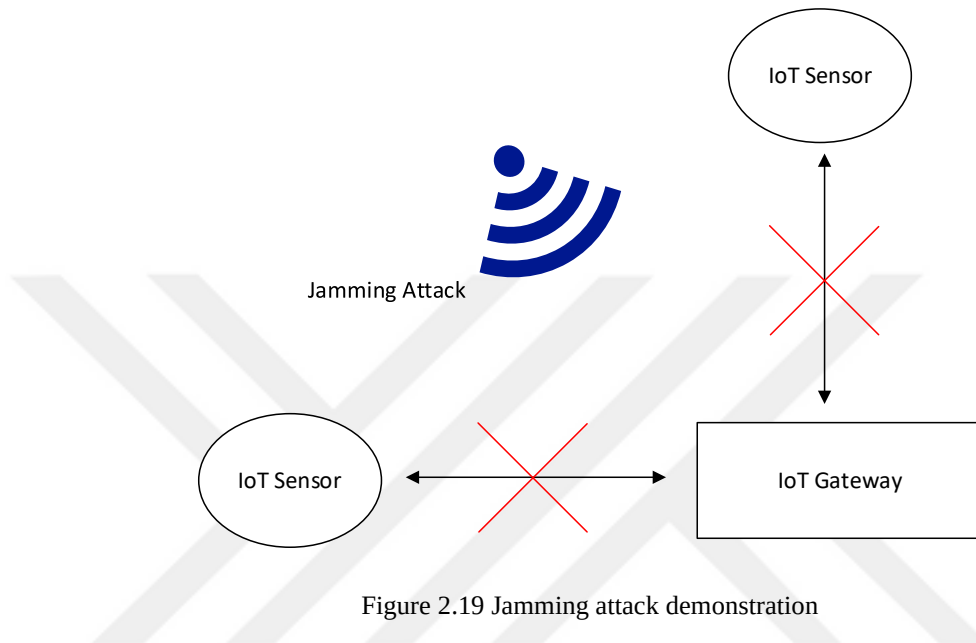


Figure 2.19 Jamming attack demonstration

2.9 Literature Review

There are many studies in the literature on the security of IoT devices.

Barbareschi, De Benedictis, La Montagna, Mazzeo & Mazzocca (2019) have proposed a Physically Unclonable Functions (PUFs) based mutual authentication scheme. The proposed protocol was claimed to be robust against tampering and man-in-the-middle attacks. However, it has been stated that the proposed scheme is vulnerable to impersonation, desynchronization, and traceability attacks (Adeli & Bagheri, 2019).

Sey's (2018) survey focused on lightweight and mutual authentication methods for the IoT. In the survey, it was stated that there are many different authentication methods used in the IT industry, but not all of them are adaptable to IoT. The survey found that mutual authentication is important for IoT. The author stated that due to the limitations of IoT devices, lightweight authentication is very useful when dealing with areas such as low bandwidth.

Arasteh et al. (2016) examined the relationship between the IoT and smart city. They also described practical experiences and key challenges in the world in their work. They stated that in cases where all data is collected and analyzed on a common IoT platform, the system may be exposed to some attacks such as side-channel attacks.

Alzahrani & Mahmood (2021) proposed an authentication protocol for IoT. They analyzed their authentication protocol in informal security analysis. Rajab & Cinkelr (2018) have reviewed the concepts of IoT and smart cities and their applications. They also explained the main challenges and weaknesses of the implementation of IoT technologies in smart cities. Ziegeldorf, Morchon & Wehrle (2014) analyzed the privacy issues in IoT in detail. They discussed the evolving features in IoT and classified the Privacy threats.

Ding, Wang, Xie & Li (2021) presented an authentication protocol for IoT devices based on elliptic curve and signcryption techniques. When they compared the proposed protocol with other protocols, they stated that while providing the security requirements, it further reduced the communication and computation costs of the client devices.

Özcanhan & Utku (2018) analyzed a proposed mutual authentication protocol for IoT devices. They made a recommendation to provide stronger security by pointing out the weaknesses of the protocol they were examining.

An authentication protocol was proposed by Bae & Kwak (2017). The proposed protocol was verified by simulation using the AVISPA tool. They stated that the proposed protocol is safe against session key leakage attacks, user impersonations, and various other attacks.

Lee, Yu, K. Park, Y. Park & Y. Park (2019) reviewed the smartcard-based two-factor authentication protocol for multi-gateway IoT proposed by Bae & Kwak (2017). They pointed out that the protocol they examined was vulnerable to session key disclosure, gateway spoofing attacks, user impersonation attacks, and did not provide mutual authentication. They proposed a three-factor mutual authentication protocol to address security vulnerabilities for multi-gateway IoT environments.

Shah & Venkatesan (2018) proposed a multi-key-based mutual authentication mechanism between IoT device and server. To prove their algorithm, they implemented this authentication scheme on an Arduino device. In addition, the power and security analysis of the proposed algorithm was performed. They stated that the algorithm they presented is safe against side-channel attacks.

Li, Yan, Zuo, Zeng & Luo (2019) suggested a virtual sensing technique for IoT devices without requiring any hardware modification in their study. The proposed method provides authentication by touching the wristband of the user for about two seconds. With this technique they proposed, they created an intuitive authentication method. They suggested using timestamp during authentication.

In the study by Lin & Bergmann (2016), they examined security and privacy challenges in different IoT application domains. As a result of their research, they discussed key future requirements for trusted smart home systems. They stated that system auto-configuration support will increase system security to provide a safe

environment for smart homes. They also stated that for the continuity of system security, automatic updating of the system software and firmware is required.

Ali, Sabir, & Ullah (2019) conducted a comprehensive study of security problems in IoT. They have classified possible cyber-attacks at each layer of the IoT architecture. In addition, they examined the difficulties of traditional security solutions such as authentication mechanisms and key management in IoT.

In the thesis study by Ismat (2015), a comparative analysis of AES encryption algorithms on Arduino was carried out. In the study, the memory usages of 5 different AES finalist algorithms were calculated and compared. In addition, the AES algorithm has been tested in encryption and decryption operations on Arduino with different key sizes.

In another study, a secure authentication scheme that provides mutual authentication between IoT devices and fog gateway has been proposed (Amanlou, Hasan & Bakar, 2021). The proposed schema uses the Elliptic Curve Diffie– Hellman Ephemeral (ECDHE) key exchange algorithm. To implement the proposed protocol, they prepared a test environment with infrastructure including cloud, fog gateway, and IoT end devices. After the tests, they made a performance evaluation. They stated that the proposed schema is secure against man-in-the-middle attacks, replay attacks, and DoS attacks.

Shariq et al. (2021) have proposed a secure RFID authentication protocol using the Digital Schnorr Cryptosystem for IoT-health in COVID-19 patient care. Using AVISPA and Scyther tools, they performed a security analysis of the proposed protocol. The proposed protocol resists various known passive and active attacks, including replay and man-in-the-middle attacks.

A lightweight smartcard-based secure authentication (LS-BSA) approach was presented in the study by Deebak & AL-Turjman (2021). Fuzzy verifier, elliptic-curve cryptosystems, and Bilinear-pairing/mapping are used in the presented scheme to increase efficiency in terms of communication and computation costs. They showed that data privacy is ensured for a reliable connection by performing formal analysis with real or random (RoR) and BAN-logic.

In a study by Lara, Aguilar, Sanchez & García (2020), an authentication protocol called LAKD has been proposed for the industrial Internet of Things (IIoT) devices. The security analysis of the proposed authentication method was performed with the Burrows–Abadi–Needham (BAN) logic and AVISPA tool. With the informal analysis they carried out, it was stated that there was resistance against known attacks such as man-in-the-middle, impersonation, replay, and DoS attacks.

In the study by Joshitta & Arockiam (2017), the authentication mechanism for IoT devices is presented. It is stated that the proposed schema is resistant to replay attack, impersonation, man-in-the-middle attack, eavesdropping and DoS attack. In addition, a formal security analysis of the proposed mechanism was carried out.

A cloud-based biometric system architecture was proposed in the study by Shah, Bharadi, Kaul & Amrutia (2015). Raspberry Pi is used as a remote authentication node. In the proposed system, biometric features such as fingerprints and face are captured and sent to the cloud service with encryption process. AES-256 algorithm is used for encryption of data.

Aman, Chua & Sikdar (2017) presented PUFs based lightweight mutual authentication protocols for IoT systems. One of the presented protocol is for the communication between the server and the IoT device, and the other is for the

communication of two IoT devices with each other. The authors presented the performance and security analysis of the proposed protocols.

Khemissa & Tandjaoui (2016) propose a new lightweight authentication scheme. The proposed scheme ensures that both the remote user and the sensor authenticate with each other for secure communication. In the proposed scheme, keyed-hash message authentication and exclusive-or transactions are used to check the integrity of different exchanges. The authors stated that the proposed schema is resistant to different types of attacks by analyzing performance and security.

In another study, they first examined the need for lightweight encryption and their design differences with normal block ciphers (Surendran, Nassef & Beheshti, 2018). They compared the performance of encryption algorithms in terms of energy, changing key size, changing packet size, and power consumption. They also examined the attacks against some lightweight crypto algorithms. In addition, performance analysis of lightweight ciphers was performed on Windows and Embedded platforms. Afterward, they compared the execution of some cipher algorithms on Windows and Embedded platforms and stated that some lightweight ciphers performed better on the Windows platform compared to embedded platforms.

In the study by Grammatikis, Sarigiannidis & Moscholios (2019), a comprehensive security analysis was conducted by examining potential threats and countermeasures in the IoT environment. By examining IoT security threats based on layers, they determined risk levels for each threat as high, low, and medium. They offered appropriate solutions against the risks they identified.

Shah & Bharadi (2016) proposed a biometric system using biometric technology. In the system they developed, they ensured the security of the data sent from the Raspberry Pi device to the cloud using RSA and AES-256. The authentication process

is provided by the biometric service hosted on the Azure cloud. Encrypted data is stored in the cloud. They measured memory usage, CPU usage, and encryption time on Raspberry Pi using the AES-256 encryption algorithm.

Al-Turjman, Zahmatkesh, & Shahroze (2019) examined the privacy and security problems of IoT-based applications in smart cities. They briefly described possible cyber-attacks in smart cities. They also introduced security and privacy solutions such as biometrics, cryptography, and machine learning techniques used in smart environments. They emphasized that security threats should be considered in the design of new smart systems.

Nandy et al. (2019) conducted research on authentication mechanisms in IoT devices to generate ideas for researchers working on IoT authentication security. They emphasized the attacks and technical methods against IoT authentication mechanisms. They demonstrated the available protocols to avoid potential threats in IoT authentication. They discussed IoT authentication evaluation plans.

Yahuza et al. (2021), in their study on Internet of Drones (IoD) security, emphasized the need for secure Internet of Drones architecture. They stated that the drones on the Internet of Drones network are connected to each other by IoT. They proposed a secure Internet of Drones architecture. The central server in the IoD architecture they recommend helps authenticate the communicating devices before the communication process. They investigated privacy and security threats from different categories of drones.

Aghili et al. (2019) presented a secure and lightweight mutual RFID authentication protocol called secLAP. The proposed protocol provides secure communication in medical Internet of Things (MIoT) systems and protects privacy. They used Burrows–Abadi–Needham (BAN) logic to validate the security features of their proposed

protocol. With their security analysis, they stated that the proposed protocol is robust against traceability, reader/tag impersonation, replay, and de-synchronization attacks and provides backward and forward data communication security.

Dhanda, Singh, & Jindal (2020) compared 54 Lightweight Cryptography primitives in their own classes in their study. Of these, 19 are lightweight stream ciphers, 21 are lightweight block ciphers, 5 are variants of elliptic curve cryptography (ECC) and 9 are lightweight hash functions. The comparison was on latency, throughput, software and hardware efficiency, power and energy, and chip area. As a result of the study, they stated that elliptic curve cryptography is an important option that provides authentication in asymmetric cryptography and among the block ciphers, most competitive cipher is AES.

ECC-based mutual authentication schema for IoT and cloud servers was proposed by Kalra & Sood (2015). The proposed schema is shown in Figure 2.20. Formal verification of the proposed protocol was performed using the AVISPA tool. The operation of the protocol is as follows.

- **Step1:** First, the embedded device sends its identity (X_a) to the server to register with the server Srv.
- **Step2:** When Server Srv receives the identity of the embedded device (X_a), it generates a unique password (Pwd_a) for the device and chooses a unique random number (Rnd_a). Server Srv then calculates $CI = H(Rnd_a || PK || EXP_T || X_a)$, $CI' = CI \times Gen$, $T_a = Rnd_a \oplus H(PK)$, $A_a = H(Rnd_a \oplus H(PK) \oplus Pwd_a \oplus CI')$ and $A_a' = A_a \times Gen$. The server stores A_a' , T_a , X_a , EXP_T in its database.
- **Step3:** Server Srv sends CI' to the embedded device.
- **Step4:** Embedded device chooses a random number (B_1). It calculates $Pwd_1 = B_1 \times Gen$ and $Pwd_2 = H(B_1 \times CI')$ using B_1 and CI' .
- **Step5:** Embedded device sends X_a , Pwd_1 , Pwd_2 to server Srv.

- **Step6:** The server calculates $Rnd_a = T_a \oplus H(PK)$, $CI = H(Rnd_a || PK || EXP_T || X_a)$ and $Pwd_2' = H(Pwd_1 \times CI)$ when it receives X_a , Pwd_1 , Pwd_2 . Then the server checks $Pwd_2' = Pwd_2$. If not equal, X_a is not a legal user and the session is terminated. Otherwise, the server Srv picks a random number (B_2) and calculates $Pwd_3 = B_2 \times Gen$ and $Pwd_4 = B_2 \times A_a'$.
- **Step7:** Server Srv sends Pwd_3 , Pwd_4 , T_a to the embedded device.
- **Step8:** Embedded device receives Pwd_3 , Pwd_4 , T_a from server Srv and calculates $A_a = H(T_a \oplus Pwd_a \oplus CI')$ and $Pwd_4' = Pwd_3 \times A_a$. Embedded device checks $Pwd_4' = Pwd_4$. If equal, Srv is a legal server. It continues with the calculation $V_a = H((B_a \times CI') || Pwd_4')$.
- **Step9:** Embedded device sends V_a to Srv.
- **Step10:** Srv receives V_a from the embedded device and calculates $V_a' = H(Pwd_1 \times CI) || Pwd_4$. Then, Srv checks $V_a' = V_a$.

After mutual authentication, both devices agree on a session key. Session Key is equal to $H(PK || X_a || B_1 || B_2)$

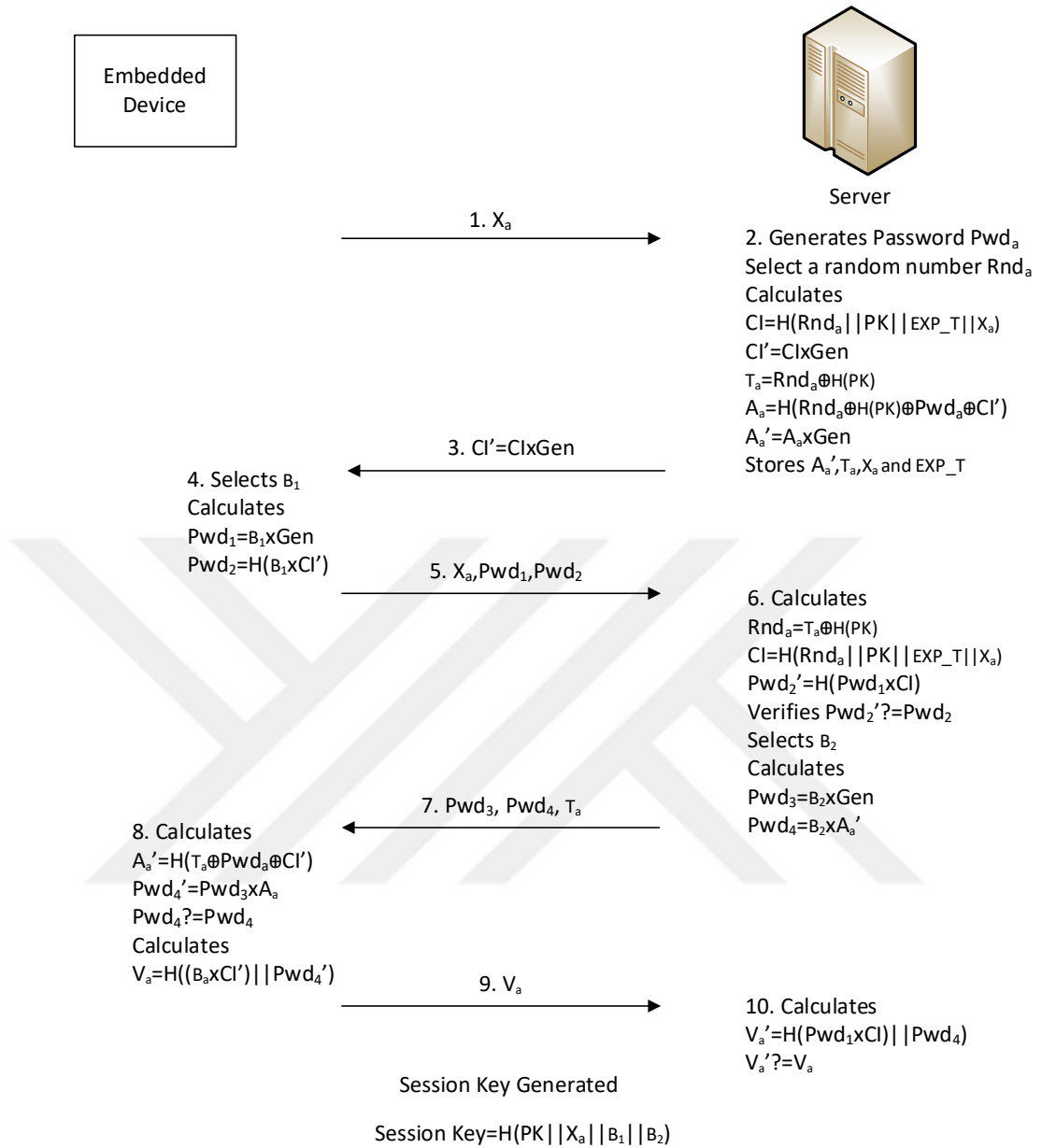
Chang, Wu & Sun (2017) found the mutual authentication failure of the protocol proposed by Kalra & Sood (2015). They also found the session key uncertainty of the proposed protocol. Chang, Wu & Sun (2017) first reviewed and analyzed the schema proposed by Kalra & Sood (2015). In their analysis, they stated that the embedded device cannot calculate A_a . T_a and CI' in the $A_a = H(T_a \oplus Pwd_a \oplus CI')$ calculation are delivered by the server in the authentication and registration phase. They stated that since the Pwd_a value created by the server is not sent to the embedded device, the embedded device cannot calculate the Pwd_4' value. Thus, they stated that the scheme proposed by Kalra & Sood (2015) could not perform mutual authentication.

Chang, Wu & Sun (2017) also stated that the server cannot obtain the B_1 value, the embedded device cannot obtain the B_2 value, and the embedded device cannot obtain the server's private key. Therefore, they explained that the server and the embedded

device cannot calculate the session key. Afterward, they suggested corrections as follows.

- They updated the Pwd_a value in step 2 of the schema to $H(Pwd_a)$.
- In step 3, the server will send the $H(Pwd_a)$ value to the user.
- In Step 8, they updated the Pwd_a value to $H(Pwd_a)$ in A_a .

After the embedded device and the server authenticate with each other, the session key value will be $H(X_a || B_1 x B_2 x Gen)$.



Notations used in the protocol

X _a	Identity of the device DVC _a
Pwd _a	Password of the device DVC _a
Rnd _a	Random number generated by the server
CI	Cookie information
H()	One-way hash function
PK	Private key of the server based on ECC
EXP_T	Expiration time of the cookie
Gen	Generator point of a large order n
	Concatenation
⊕	XOR Operation
B ₁ , B ₂	Random numbers generated for ECC parameters

Figure 2.20 Proposed schema by Kalra & Sood (2015)

In the study by Tewari & Gupta (2017), an authentication protocol is presented for IoT devices. The proposed schema is shown in Figure 2.21. The operation of the proposed protocol is described below.

Key and IDS are updated when authentication is completed successfully. The update may be interrupted. Therefore, these values are backed up as Key_{old} and IDS_{old} before the update.

- **Step 1:** The reader starts the protocol execution by sending the "hello" message to the tag.
- **Step 2:** After the tag receives the "hello" message from the reader, it sends $\{IDS_{new}, IDS_{old}\}$ to the reader.
- **Step 3:** The reader, which receives the IDS information from the tag, matches it with the values in the database and returns the Key information of this tag. If it does not match, the protocol is terminated. Then the reader generates 2 random numbers (a, b). Then it calculates $X=IDS \oplus a \oplus b$, $Y=Key \oplus b$, and $Z=Rot(Rot(Key \oplus b, IDS), Key \oplus b)$ and sends it to Tag.
- **Step 4:** The tag, which receives X, Y, Z values from the reader, calculates $b=Y \oplus Key$, $a=X \oplus b \oplus IDS$ and $Z'=Rot(Rot(Key \oplus b, IDS), Key \oplus a)$. If $Z'=Z$, Tag calculates $M=Rot(Rot(IDS \oplus a, Key), Z' \oplus b)$. Otherwise, the protocol is terminated. The M value is sent to the reader.
- **Step 5:** The reader, which takes the M value from the tag, calculates $M'=Rot(Rot(IDS \oplus a, Key), Z \oplus b)$. If $M'=M$, then mutual authentication is successful. IDS_{old} and Key_{old} values are changed as $IDS_{old}=IDS_{new}$ and $Key_{old}=Key_{new}$.

After this process both side will update their IDS and Key as $IDS_{new}=Rot(IDS \oplus b, Key \oplus b, IDS \oplus a)$ and $Key_{new}=Rot(Z \oplus b, IDS \oplus a)$.

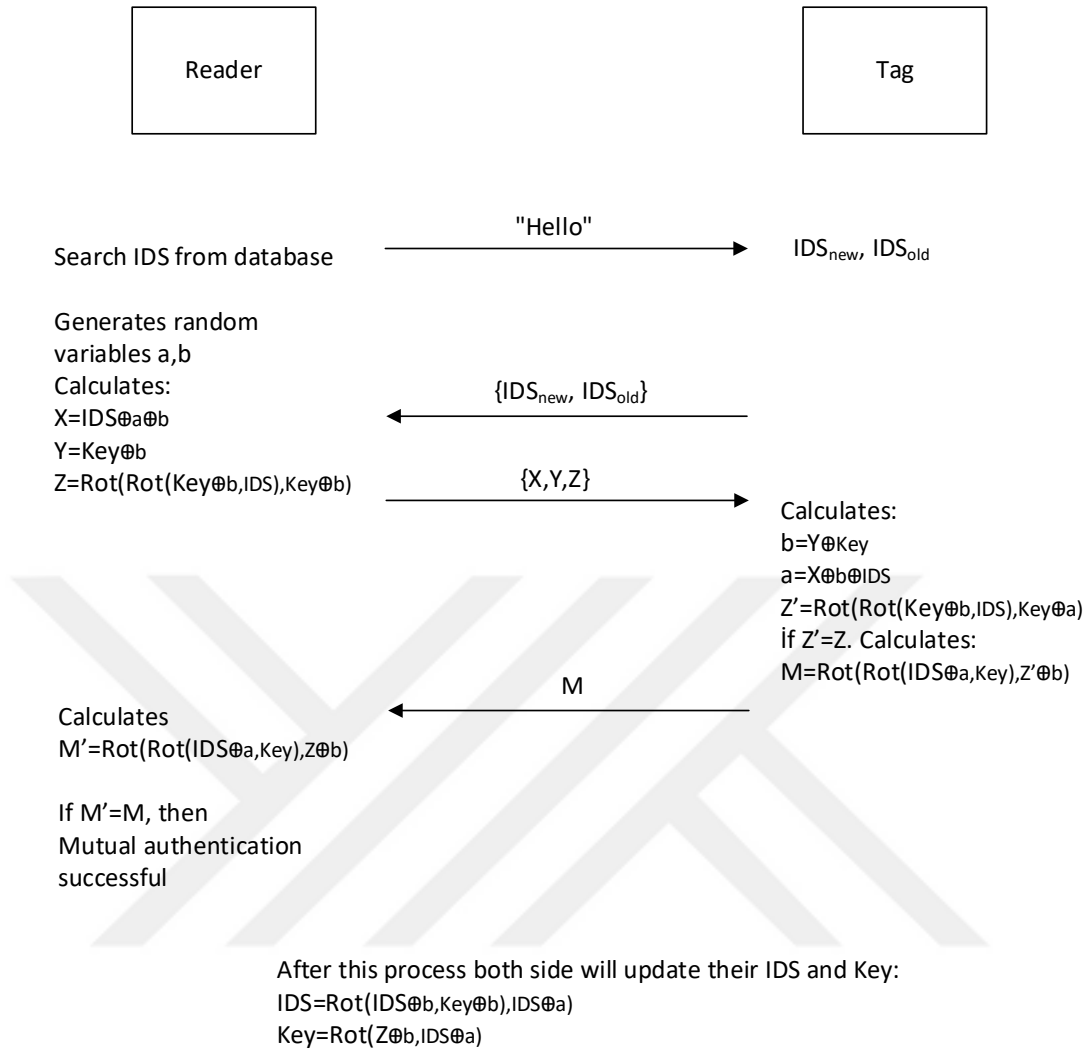


Figure 2.21 Proposed schema by Tewari & Gupta (2017)

Wang, Chen, Fang, & Wu (2018) stated in their study that in the protocol proposed by Tewari & Gupta (2017), an attacker can obtain the shared key between the tag and the server. They explained that by acting in passive mode, the attacker could perform a key disclosure attack with the algorithm they suggested, and a list of possible Keys could be drawn up. They stated that in subsequent sessions, the attacker could verify every possible Key by checking the updated IDS_{new} value.

In addition, Wang, Chen, Fang, & Wu (2018) suggested some modifications to the presented protocol. It changes like $Z = \text{Rot}(\text{Rot}(\text{Key}, \text{IDS} \oplus b) \oplus a, \text{Key})$ and $M = \text{Rot}(\text{Rot}(\text{Key}, \text{IDS} \oplus a) \oplus b, \text{Key} \oplus Z)$ with the changes they suggest.

In the study by Fu, Wu, Chen, Fan & Ping (2010), mutual authentication protocol was proposed for RFID systems. The operation of the protocol shown in Figure 2.22 is as follows. The notations used in the protocol are also indicated in Figure 2.22.

- **Step 1:** First, the reader generates a random value (rnd1). Then it sends it to tag with Msg1.
- **Step 2:** Tag generates a random value (rnd2). It then calculates $H(\text{Key} \parallel \text{rnd1})$. Then it sends the Msg2 packet to the reader with rnd2, $H(\text{Key} \parallel \text{rnd1})$, and IDSA.
- **Step 3:** Then the reader sends Msg3 to the server with Msg2 and rnd1.
- **Step 4:** The server obtains the IDS by decrypting the IDSA with the key known only to itself. Then it brings the Key. The server checks $H(\text{Key} \parallel \text{rnd1})$. If it is equal to the received value, the tag will authenticate successfully. Otherwise, authentication failed. If the tag is legal, a new random value (rnd0') is generated and the server calculates $\text{IDSA}' = E_{\text{Keys}}(\text{IDS} \parallel \text{rnd0}')$, $X = \text{IDSA}' \text{ xor } H(\text{Key} \parallel \text{rnd1} \parallel \text{rnd2})$ and $Y = \text{IDSA}' \text{ xor } H(\text{Key} \parallel \text{rnd2} \parallel \text{rnd1})$. Then, the server calculates $H(\text{Key} \parallel \text{rnd2})$ and sends it to the reader with Msg4.
- **Step 5:** The reader, who receives Msg4 from the server, transmits it to the tag as Msg5.
- **Step 6:** The tag verifies $H(\text{Key} \parallel \text{rnd2})$. If unsuccessful, authentication fails. If authentication successful, Tag calculates $H(\text{Key} \parallel \text{rnd1} \parallel \text{rnd2})$ and $H(\text{Key} \parallel \text{rnd2} \parallel \text{rnd1})$. Then, Tag calculates $\text{IDSA1} = X \oplus H(\text{Key} \parallel \text{rnd1} \parallel \text{rnd2})$ and $\text{IDSA2} = Y \oplus H(\text{Key} \parallel \text{rnd2} \parallel \text{rnd1})$. If $\text{IDSA1} = \text{IDSA2}$, the Tag stores IDSA1 as the new IDSA. Finally, it sends Msg6 as OK to the reader.

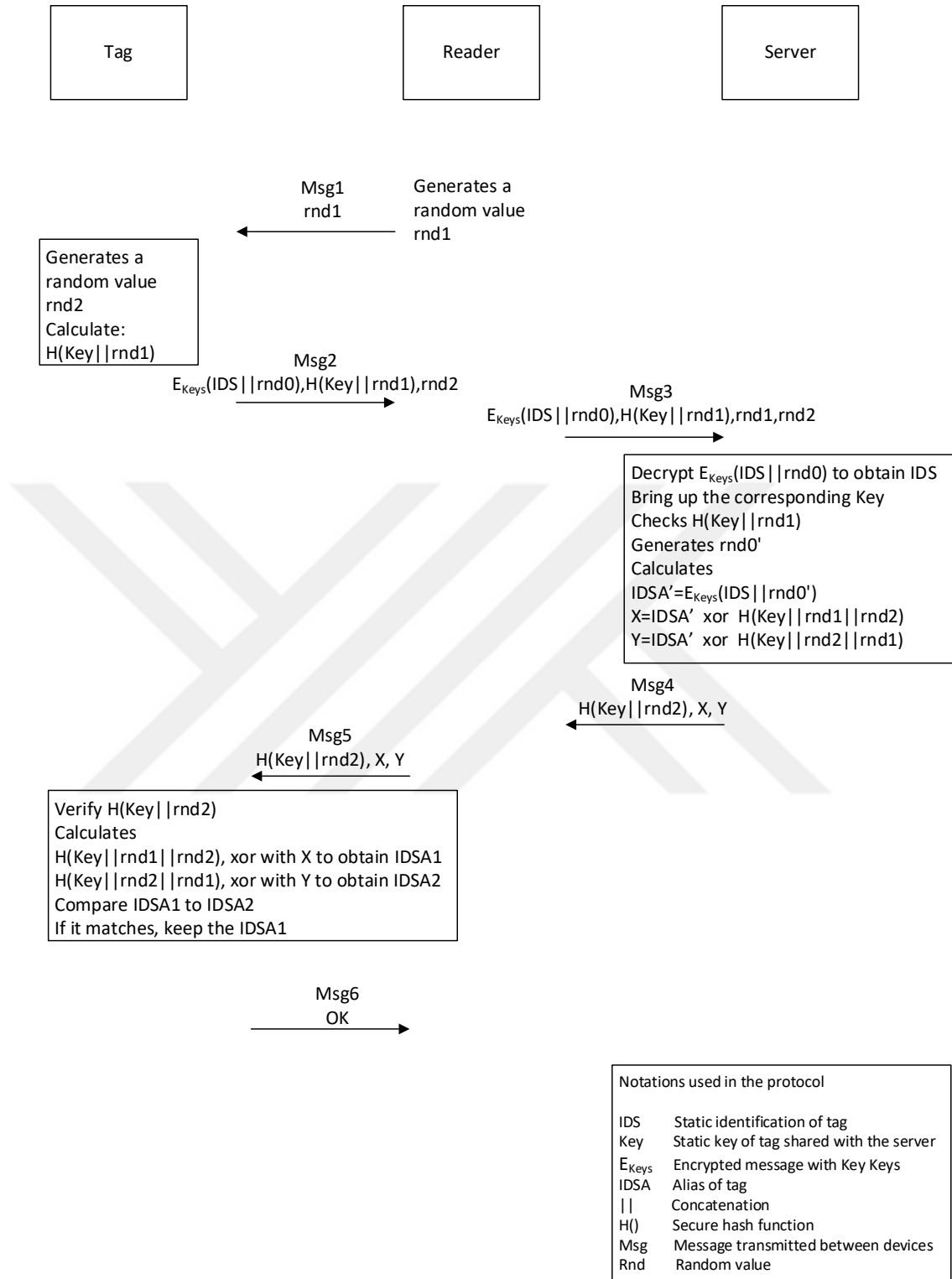


Figure 2.22 Proposed schema by Fu et al. (2010)

In the study by Deng & Zhu (2013), it was stated that the protocol proposed by Fu et al. (2010) is vulnerable to desynchronization attack. The vulnerability identified by Deng & Zhu (2013) is shown in Figure 2.23.

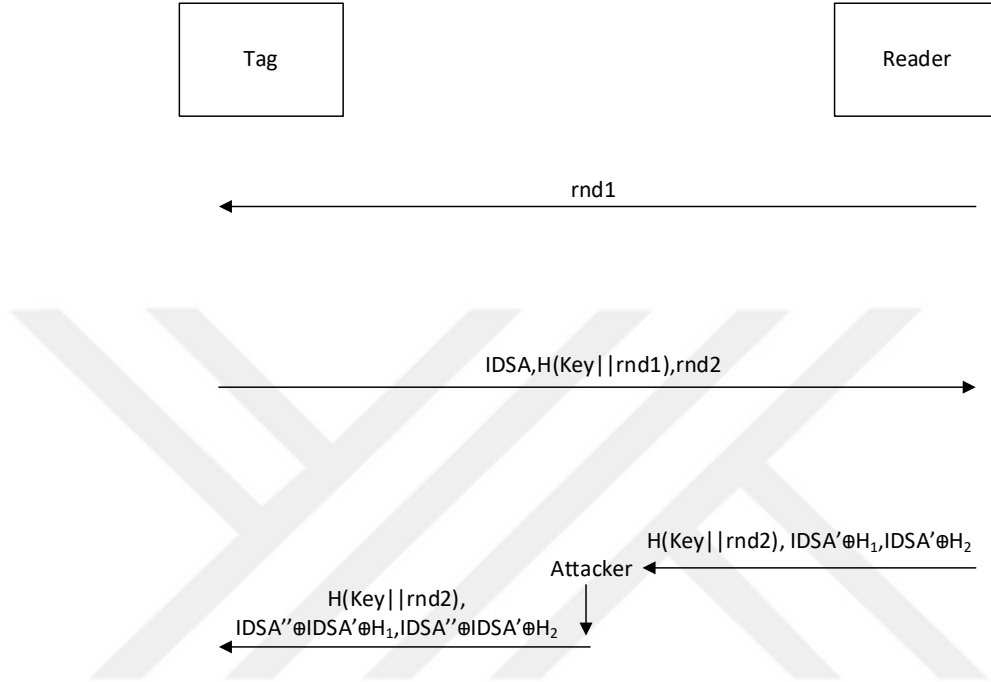


Figure 2.23 Desynchronization attack on the proposed schema by Fu et al. (2010)

As shown in Figure 2.23, it is stated that the attacker can listen to the messaging between the Reader and the Tag. It has also been stated that the attacker will allow the first two messages between the tag and the reader to be sent, but will change the two values in the third message to $IDS A'' \oplus IDS A' \oplus H(\text{Key} || \text{rnd1} || \text{rnd2})$ and $IDS A'' \oplus IDS A' \oplus H(\text{Key} || \text{rnd2} || \text{rnd1})$. The $IDS A''$ value here is a random number. Tag is not authorized by the reader in subsequent sessions, as the Attacker allows the reader and Tag to share different Alias.

Deng & Zhu (2013) suggested an improvement to the protocol proposed by Fu et al (2010) for the desynchronization attack described above, as follows.

In the third step, the reader calculates $H(\text{Key}||\text{IDSA}')$ and sends $H(\text{Key}||\text{rnd2})$, $\text{IDSA}' \oplus H(\text{Key}||\text{rnd1}||\text{rnd2})$, $\text{IDSA}' \oplus H(\text{Key}||\text{rnd2}||\text{rnd1})$ and $H(\text{Key}||\text{IDSA}')$ to the tag. In step four, after $\text{IDSA1} = \text{IDSA2}$ verification, Tag calculates $H(\text{Key}||\text{IDSA1})$ and checks its equality with $H(\text{Key}||\text{IDSA}')$. In case of match, Tag stores IDSA1 as new alias and sends OK message to reader.

Unlike the studies examined in the literature research, the mutual authentication protocol proposed in this thesis study is safe against replay, full-disclosure, ID-theft, man-in-the-middle, de-synchronization, identity tracing, impersonation, DoS, clone, and malicious code injection attacks. In addition, the mutual authentication protocol proposed in this thesis has been validated using Scyther and AVISPA tools.

CHAPTER 3

DESIGN AND IMPLEMENTATION OF A TYPICAL IOT DEVICE

3.1 Overview

Authentication protocol has been proposed to ensure data security while transmitting the sensor data collected by IoT devices used in smart cities to the IoT gateway. A prototype was created to test the proposed FSMAP among IoT devices. With the created smart textile device prototype, the measurement of the inner angle of the human arm is carried out. While creating the prototype, sensors and microcontroller card were integrated on the textile (woven fabric) containing conductive wires. In this section, the smart textile device, which consists of a combination of woven fabric containing conductive wires and IoT devices, and the hardware that makes up this device are explained. In addition, the Windows application developed to show the sensor data to the user and the security vulnerabilities in IoT are mentioned.

3.2 Materials

A prototype has been built to measure the human arm interior angle through sensors that will send data to the IoT gateway. Two MPU 6050 accelerometers and gyro sensors were used to measure the arm interior angle. To properly position the sensors on the textile, two perforated pertinax were used as shown in Figure 3.1. The perforated pertinaxes assist in fixing the sensors on the woven fabric. Conductive wires are used to connect the sensors with the woven fabric.

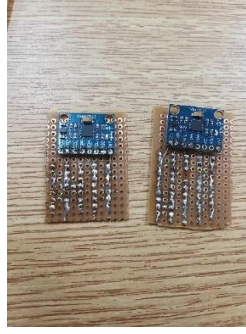


Figure 3.1 MPU 6050 sensors (Kişisel arşiv, 2021)

The connection of the sensors and the microcontroller card with the woven fabric is shown in Figure 3.2. Conductive wires are also used between the microcontroller card and the woven fabric. The hardware connection between the sensors and the microcontroller card is made through the conductive wires in the woven fabric. The Bluetooth module is connected to the Arduino UNO with a cable.

The wires in the woven fabric are in a fine structure to provide the flexibility of the textile. Therefore, it should be ensured that the connections made to the wires in the textile are not broken. As shown in Figure 3.2, it is fixed on the textile using silicone so that the wires soldered on the textile do not move.



Figure 3.2 Smart textile (Kişisel arşiv, 2021)

The connection diagram of the created smart textile is as seen in Figure 3.3.

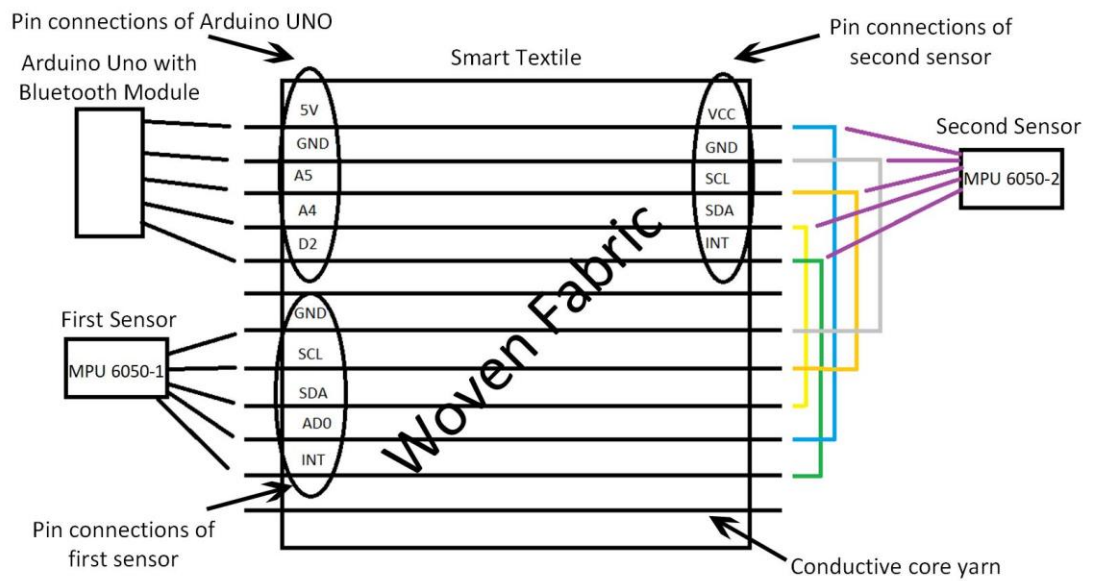


Figure 3.3 Smart textile circuit diagram

Arduino UNO with Bluetooth module and 2 MPU 6050 sensors are fixed on textile. Afterward, the created smart textile was wrapped on polyethylene material as in Figure 3.4 to perform the simulation by testing its use on the human arm.



Figure 3.4 Smart textile prototype (Kişisel arşiv, 2021)

With the prototype created, the measurement of the inner angle of the human arm is carried out. By measuring the inner angle of the human arm, the arm mobility of the elderly and sick people can be determined.

Arduino UNO microcontroller board with Bluetooth module is used as IoT gateway. The Arduino UNO, to which the sensors are connected, wirelessly transmits the data to the IoT gateway via Bluetooth.

To display the data received from the sensors on the computer screen, coding was done with C# language using Visual Studio. With the Windows application shown in Figure 3.5, the angle data from both sensors is printed on the screen. Both sensors perform angle measurement with reference to the ground. The Windows application calculates the difference of the angles (Θ) and the measured minimum-maximum difference values and prints them on the screen.

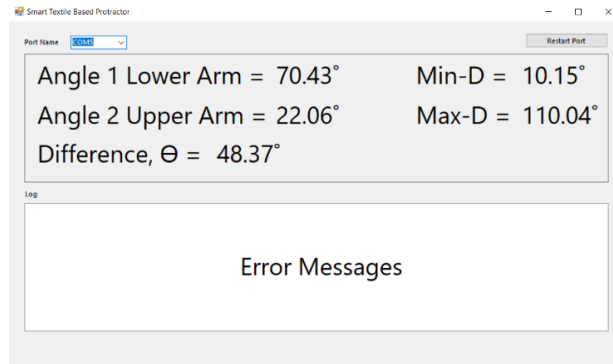


Figure 3.5 Windows application

3.3 Vulnerabilities in IoT

As it is known, IoT devices are devices with limited resources because they are intended to do certain tasks. It is often not possible to apply the security measures used in computer systems on IoT devices because they do not have high memory capacities and high processing power. Therefore, it is possible to say that their defense power against cyber-attacks is low. In a study conducted by HP in 2014, it was stated that 70% of the most widely used IoT devices contain serious security vulnerabilities. (Zhao, Shi, Cai & Chen, 2018).

While there are risks such as physically intervening in IoT devices and deleting and changing the information inside, they also have low defense mechanisms against unauthorized access by remote access method, since they are connected to the internet. Especially with the increase in the use of wireless technologies, IoT devices without security measures can be reached wirelessly from certain distances.

Considering that today IoT devices are used in wearable devices, smart homes, health, transportation, and similar areas, many IoT devices that also contain personal data need an accurate, robust, lightweight, and convenient authentication mechanism (Hutchins et al., 2018). In general, it is not possible to apply traditional authentication methods because hardware such as keyboard and mouse are not used in IoT devices.

When applying security measures on IoT devices, care should be taken to use security applications that consume fewer resources. The fact that most of the available resources of the IoT device are devoted to security applications will seriously reduce the working performance and efficiency.

In our proposed design, the FSMAF is offered, which requires low power consumption. With this protocol, which we recommend to use in wireless communication of IoT devices, data communication security between IoT devices increases.

3.4 The Hardware Used in the Designed IoT Device

The hardware used while creating the design that ensures the safe transmission of the data collected from the sensors is described below.

3.4.1 *Arduino UNO*

Arduino is a microcontroller board with input/output units used to develop stand-alone interactive objects. It is a programmable platform with Arduino IDE software by making a wired connection to the computer. Arduino IDE software runs on Linux, Macintosh OSX, and Windows operating systems.

UNO series, one of the most widely used Arduino boards, uses the ATmega328 microcontroller produced by Atmel. Arduino UNO is an application development board used in real-time system designs.

Arduino UNO was used to process the data collected from the sensors and send them to the IoT gateway via Bluetooth. In addition, Arduino UNO was used as an IoT gateway. Figure 3.6 shows the top view of the Arduino UNO board.

There are 14 digital input/output pins on the Arduino UNO board. Each pin can receive or provide 20mA. To avoid permanent damage to the microcontroller, the maximum value of 40mA should not be exceeded. 6 out of 14 digital pins can be used as Pulse Width Modulation (PWM) outputs. Arduino UNO also has 6 analog input pins. The ATmega328 also supports I2C (Inter-Integrated Circuit) and SPI (serial peripheral interface) communication.

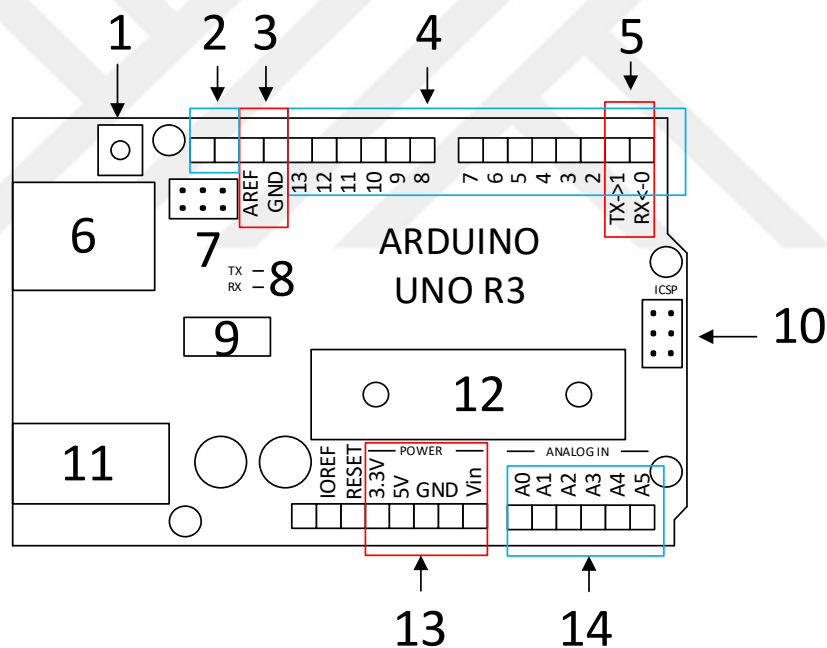


Figure 3.6 Arduino UNO rev3 microcontroller (Arduino UNO Rev3, 2021)

The recommended input voltage for Arduino UNO is in the range of 7-12 Volts. When powered with less than 7 volts the board may be unstable and the 5V pin may supply less than 5 volts. The board may be damaged if a supply voltage of more than 12 volts is applied. It can also be operated via the USB port on it. Serial communication can be provided through the TX-RX ports on the board.

The explanation of the hardware parts marked with numbers on the Arduino UNO is given in Table 3.1.

Table 3.1 Arduino UNO hardware part description

No	Description
1	Reset button
2	SCL/SDA (I ² C) communication pins
3	Aref & Gnd Pins
4	Digital I/O Pins
5	TX-RX Pins
6	Power connection
7	ICSP pins for USB
8	TX-RX Leds
9	16 Mhz Crystal
10	ICSP (In-Circuit Serial Programming) pins
11	USB connection
12	ATmega328P Microcontroller
13	Power Pins (3.3V, 5V, Gnd, V _{in})
14	Analog input pins

3.4.2 MPU 6050 Accelerometer & Gyro Sensor

MPU 6050 is a 6-axis motion tracking device with 3-axis gyroscope and 3-axis accelerometer. The MPU 6050 includes three 16-bit analog-to-digital converters (ADCs) to digitize the gyroscope outputs and 3 16-bit ADCs to digitize the accelerometer outputs. Its accelerometer and gyroscope can be programmed by the user to precisely track both fast and slow movements. MPU 6050 is a sensor that supports the I2C protocol. I2C is the serial communication protocol. When connecting with Arduino UNO, Arduino UNO pins that support I2C protocol should be used. The x and y axes are shown on the sensor.

Figure 3.7 shows the MPU 6050 accelerometer & gyro sensor. MPU 6050 accelerometer & gyro sensor was used to perform the angle measurement.

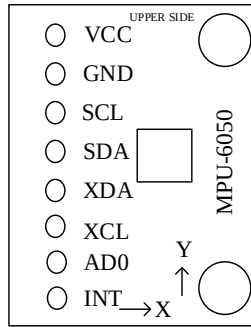


Figure 3.7 MPU 6050 accelerometer & gyro sensor

3.4.3 Bluetooth Module

HC-05 is a Bluetooth SPP (serial port protocol) module. It supports Bluetooth 2.0. It allows communication at 2.4 GHz frequency. The HC-05 uses UART (Universal Asynchronous Receiver Transmitter) serial communication. By using AT commands, the features of the module such as baud rate, password, and name can be changed.

HC-05 Bluetooth module is used to communicate the Arduino UNO board with other IoT devices wirelessly. The VCC and GND pins of the Bluetooth module shown in Figure 3.8 are connected to the Arduino UNO to provide the supply voltage. Data communication is provided with TX and RX pins.

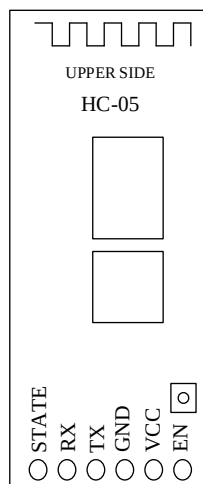


Figure 3.8 HC-05 bluetooth module

3.4.4 Woven Fabric

The woven fabric shown in Figure 3.9 was used to create a wearable smart textile. Table 3.2 shows the properties of the yarns used in the fabric content. Polyester yarns are used to obtain sufficient space between the conductive yarns on the fabric. By using polyester yarns between the conductive weft yarns, contact with each other is prevented. In Figure 3.10, polyester yarns and conductive yarns in the fabric are shown.

Table 3.2 The specifications of yarns (Dalkılıç, Özcanhan & Özdemir, 2021)

Material	Yarn count (dtex)	Diameter of wire (mm)	Conductor resistance ($\Omega\text{mm}^2/\text{m}$)
Polyester yarn	600	-	-
SS/Co core yarn	455	0.05	0.62



Figure 3.9 Woven fabric (Kişisel arşiv, 2021)

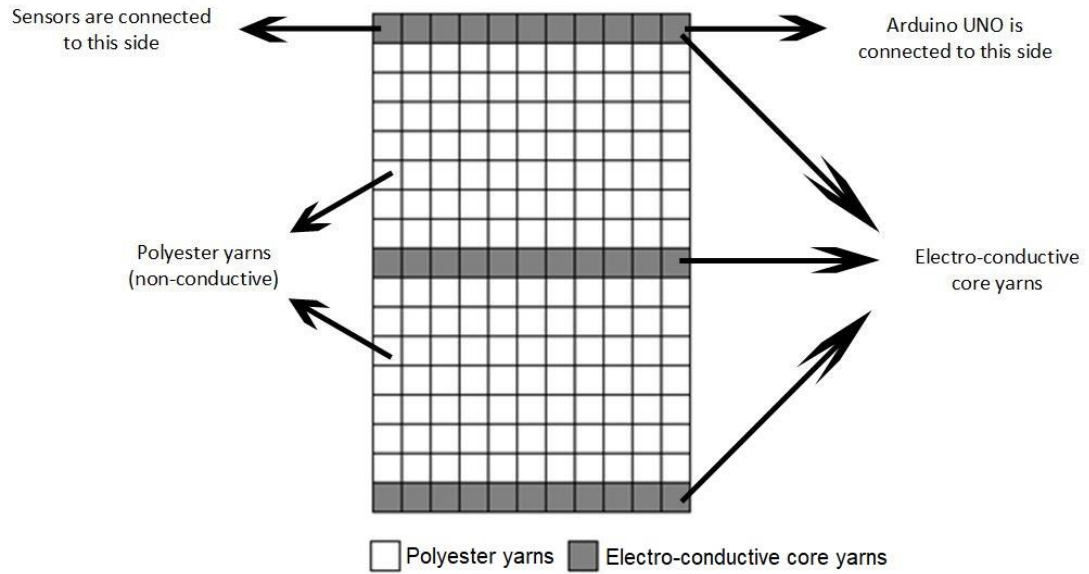


Figure 3.10 Schematic diagram of the woven fabrics (Dalkılıç et al., 2021)

3.5 Programming of Our Designed Microcontroller Hardware

In this section, the programming software used to code the microcontroller used in our design is explained. In this study, Arduino UNO board is used as Microcontroller.

Arduino IDE software is used to program the Arduino UNO microcontroller board. In the thesis study, Arduino IDE version 1.8.13, shown in Figure 3.11, was used. Arduino UNO microcontroller must be connected to the computer via a USB interface. Then, the necessary coding is made and uploaded to the Arduino UNO microcontroller.

After the codes are written in the editor, the appropriate Arduino hardware to which the codes will be sent should be selected from the "tools" menu. Then, the COM port to which the Arduino hardware is connected to the computer should also be selected from the "tools" menu. Thus, the connection between the Arduino hardware and the IDE software is established. To verify the written codes, the codes are compiled by selecting the "verify/compile" option from the "sketch" menu. Finally, to send the codes to the Arduino hardware, the codes written by selecting "Upload" from the

"sketch" menu are sent to the Arduino microcontroller. By using the "serial monitor" in the application, the data received from the sensors connected to the Arduino UNO can be displayed on the computer screen.



Figure 3.11 Arduino IDE software

In addition, the relevant libraries can be added to the Arduino IDE to facilitate the coding of the sensors connected to the microcontroller. Some libraries come with the Arduino UNO installation. Adding libraries can be done from within the Arduino IDE program. The "Library Manager" window opens by selecting "Manage Libraries" from the "Tools" menu. Then, the relevant library is selected from the listed libraries and the installation process is performed. The version of the library to be loaded can also be selected on this screen.

CHAPTER 4

PROPOSED SECURITY DESIGN AND ITS IMPLEMENTATION

4.1 Overview

This section describes the recommended encryption methods to secure data communication between IoT devices. AES implementation performance on the IoT device has been evaluated. In addition, it is shown that secure data communication is provided by simulating the proposed methods with the created smart textile hardware and the developed Windows application.

4.2 Methods

With the prototype created data transfer between IoT devices is carried out as shown in Figure 4.1. In Figure 4.1, IoT technologies used in smart cities are simulated. Arduino UNO microcontroller board, with 2 sensors connected, wirelessly transfers the data collected from the sensors to another Arduino UNO that acts as a gateway. The proposed mutual authentication protocol is used between these two IoT devices. Then, all the data collected by the gateway is transferred to the smart city cloud. The smart city application server is used to present the data on the smart city cloud to the users. Users who connect to the smart city cloud with mobile devices, notebooks, and similar devices have access to the data collected from the sensors. It is assumed that the data communication between IoT Gateway and smart city cloud is done securely.

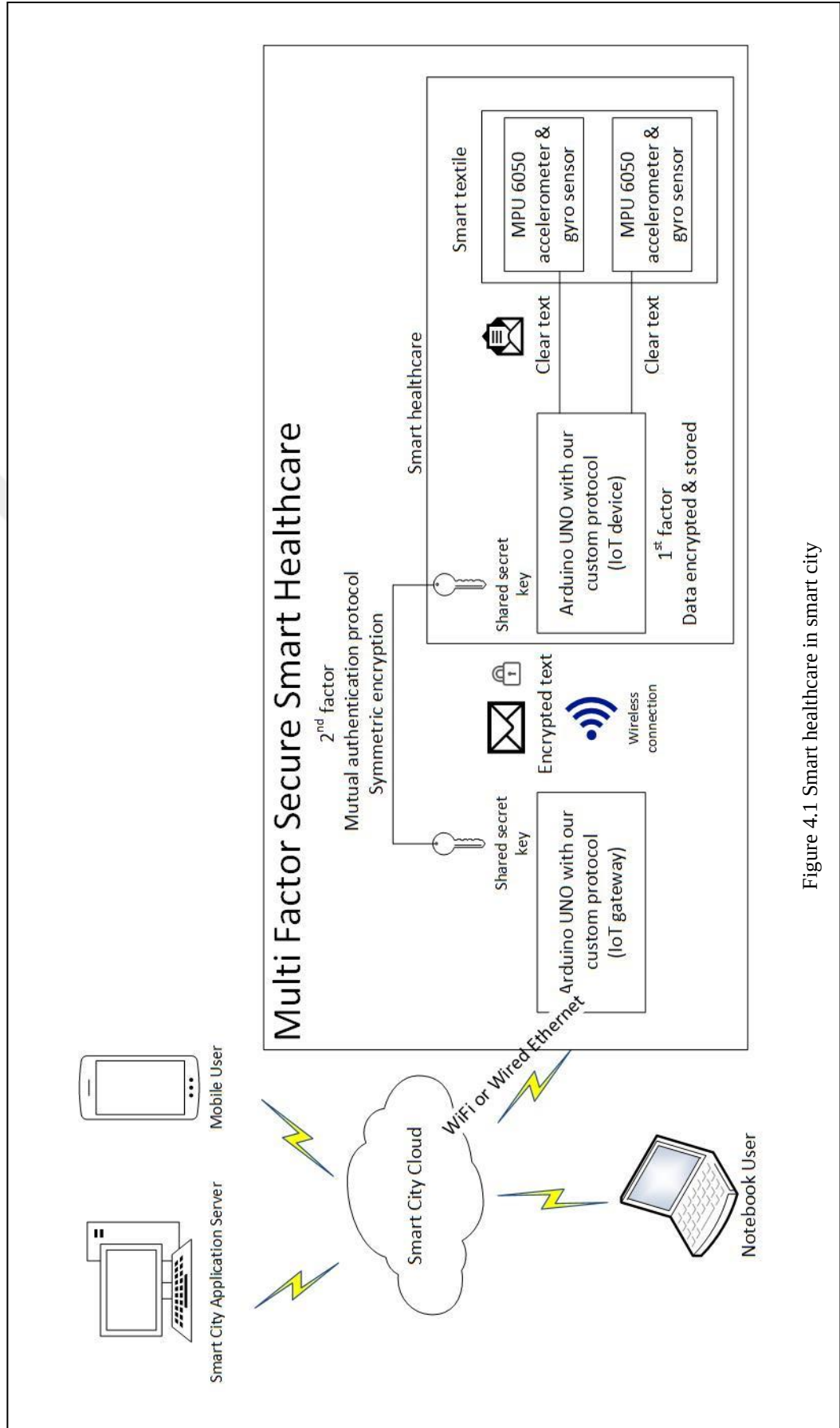


Figure 4.1 Smart healthcare in smart city

Our work is shown in Figure 4.1 in the multi-factor secure smart healthcare section. Data is taken from two MPU6050 accelerometers and gyro sensors connected to the Arduino UNO microcontroller board and encrypted with the AES encryption algorithm. AES encryption key is predefined in Arduino UNO microcontrollers, which are specified as IoT gateway and IoT device. Data stored encrypted on Arduino UNO is transferred to the Arduino UNO gateway device wirelessly. With the work done, the sensor data is transmitted securely up to the IoT gateway. The two-factor encryption method is used in the study.

The encryption methods applied to ensure mutually secure data communication between IoT devices are as follows.

- Data encryption
- Five-step mutual authentication protocol (FSMAP)

The block diagram of the proposed design is shown in Figure 4.2. As stated in Figure 4.2, firstly, data is collected from the sensors. Then, the data collected as the first encryption method is encrypted with the AES encryption algorithm on the Arduino UNO and stored encrypted on the Arduino UNO. Then, as the second encryption method, the IoT device will want to communicate with the IoT gateway using the FSMAP. If the IoT device sends data to the IoT gateway other than the recommended authentication protocol steps, the connection request will be refused by the IoT gateway. If the IoT device that wants to communicate with the IoT gateway communicates using the FSMAP, a secure connection will be established, and data communication will be realized.

The proposed design is capable of operating with low resource consumption. Therefore, it is considered to be suitable for use in many IoT devices with limited resources.

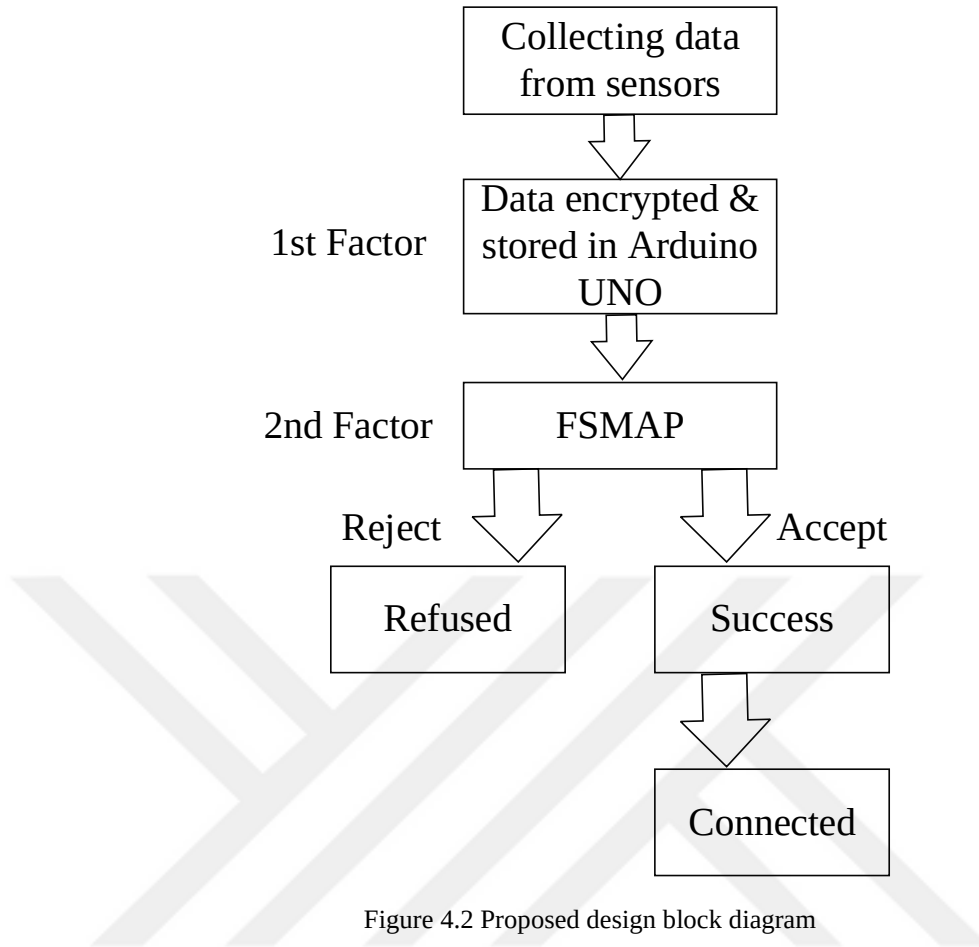


Figure 4.2 Proposed design block diagram

The encryption methods shown on the block diagram are explained in detail below.

4.2.1 Data Encryption

The data collected from the sensors are transferred to the IoT device (Arduino UNO) by wire. If an unauthorized person interferes with the IoT device, the confidentiality of the data in the device is important. Therefore, the transferred data must be protected by a security mechanism against the risk of unauthorized access. In the study, the data transferred by the sensors to the Arduino UNO is encrypted with the AES encryption algorithm and stored encrypted. Thus, in case of interference with the IoT device, it is not possible to access the content of the data in the device.

Since AES encryption keys are defined in all IoT devices beforehand, keys are not sent as plaintext between devices. As will be explained in the Mutual authentication section, renewed AES encryption keys are transmitted between IoT devices by providing an encrypted connection. The AES encryption algorithm used to encrypt data is known as a reliable encryption algorithm. In the future, it is possible to provide data communication security between IoT devices by using different encryption algorithms instead of AES encryption algorithm when using recommended encryption methods.

4.2.1.1 AES Implementation

Arduino UNO microcontrollers were coded with Arduino IDE software to simulate secure data communication between IoT devices. In this coding, the AESLib library was used to encrypt the data on the Arduino UNO with the AES-128 algorithm (Landman, 2021). Tests were carried out to measure the performance of the FSMAP coded with the AES-128 algorithm on Arduino UNO. With these tests, the working performance and memory load of the FSMAP on Arduino UNO were calculated.

4.2.1.2 AES Implementation Performance

After performing the AES Implementation on Arduino, the first test was to find the average encryption time by encrypting the data on the Arduino UNO 1000 times with AES. Likewise, the average decryption time was determined by decrypting the 1000 times encrypted data 1000 times. As shown in Table 4.1, the average encryption time of 1000 times encrypted data is 282 μ s. The average time resulting from 1000 times decryption of 1000 times encrypted data is 322 μ s. If the FSMAP is run on the Arduino UNO to which the sensors are connected, as shown in Figure 4.5, the time spent as a result of 3 encryption and 2 decryption has been determined as 1492 μ s.

Figure 4.3 graphically shows the encryption and decryption time spent when the AES encryption algorithm is applied on the Arduino UNO, as well as the total time spent for 3 encryption and 2 decryptions during five-step authentication.

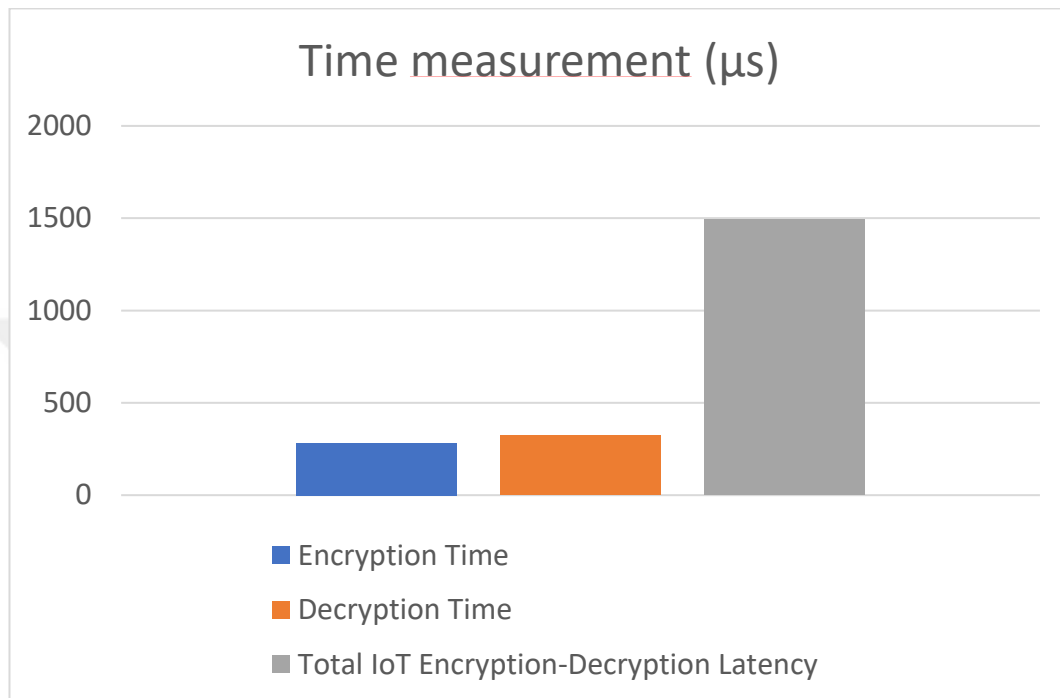


Figure 4.3 Graphical illustration of Arduino UNO time measurement

It is possible to say that the performance of the AES encryption algorithm is high as a result of working in a short time that is negligible on an IoT device with limited resources. The low time calculated for the encryption process shows that the proposed protocol can be used in many IoT devices used in smart cities.

Table 4.1 Arduino UNO time measurement

Type Of Times	Duration
Encryption Time	282 μ s
Decryption Time	322 μ s
Total IoT Encryption-Decryption Latency	1492 μ s

The memory usage of the FSMAP on Arduino UNO is also examined. With the coding done, the space occupied by the protocol on the Arduino is 4.67 KB. It can be said that the authentication protocol takes up little space for the Arduino UNO microcontroller with a total memory of 32 KB. In addition, it was determined that during the execution of the coding, it occupies 0.21 KB of space on the SRAM. Arduino UNO has a total of 2 KB of SRAM. Therefore, it can be said that the memory load on the Arduino of the FSMAP is negligible. Increasing data communication security for IoT devices with limited memory capacities is possible with FSMAP.

Table 4.2 Arduino UNO memory consumption

Type Of Memories	Total Space Available	Space Used	Space remaining
Flash	32 KB	4.67 KB	27.33 KB
SRAM	2 KB	0.21 KB	1.79 KB

4.2.1.3 Power Performance of Our AES Implementation

A mechanism as in Figure 4.4 was established to calculate how much power the FSMAP coded on Arduino UNO consumes. In the established setup, only one line of the supply voltage connected to the Arduino UNO was cut. Then the multimeter was connected in series between the two cut ends. The measurement was made while the multimeter was in mA position. Thus, the current drawn by the Arduino UNO device connected to the computer via the USB port was measured. Arduino UNO drew a current of 47.4 mA during operation as shown in Figure 4.4.

The power consumed in the Arduino UNO device was calculated during the AES encryption with Formula 4.1. Since the Arduino UNO is connected to the computer via the USB port, the supply voltage is 5 Volts. The current drawn by Arduino UNO is 47.4 mA.

$$P=V \times I \quad (4.1)$$

As a result of the calculation, the power consumed by Arduino UNO for the FSMAP was 237 mW. It has been observed that there is no power consumption that will affect the performance of the hardware while the AES algorithm is running on the Arduino UNO.

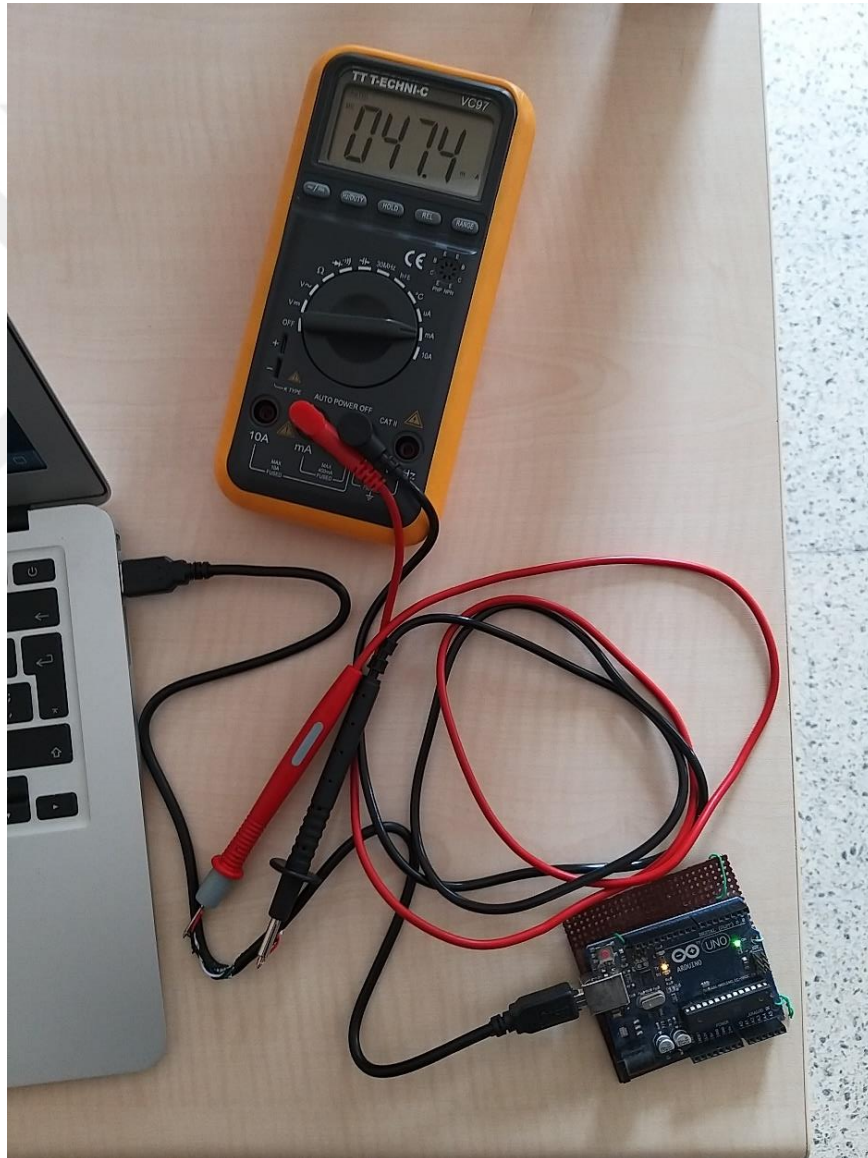


Figure 4.4 Current measurement on Arduino UNO (Kişisel arşiv, 2021)

4.2.2 Five-Step Mutual Authentication Protocol (FSMAP)

The second encryption method applied for the design shown in Figure 4.1 is mutual authentication. Mutual authentication is performed to securely transfer the sensor data in the IoT device to the IoT gateway. The mutual authentication protocol to be applied between the IoT device to which the sensors are connected and the IoT gateway is shown in Figure 4.5. The designed mutual authentication protocol consists of five steps. The abbreviations and notations used in the authentication protocol are also indicated in Figure 4.5



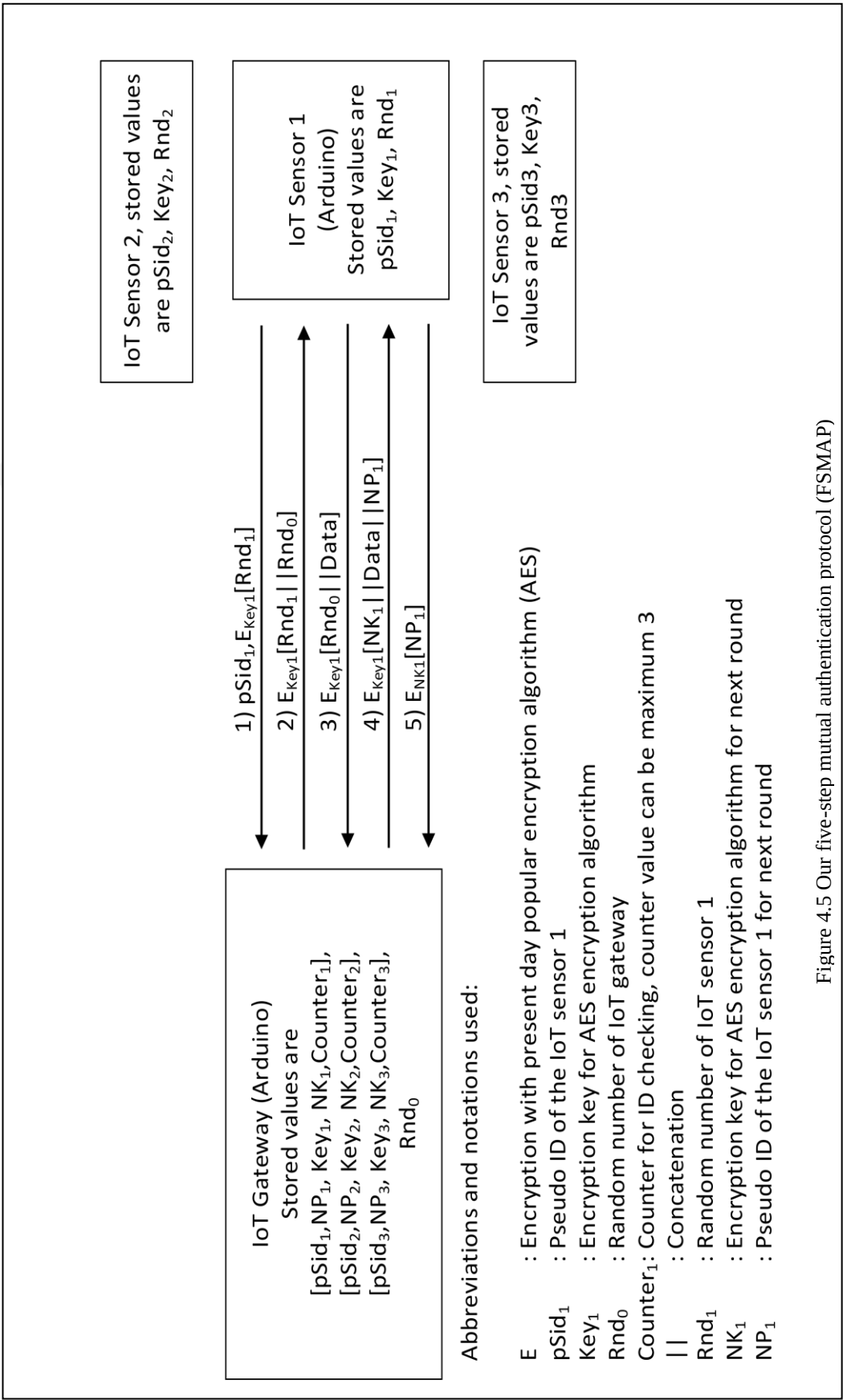


Figure 4.5 Our five-step mutual authentication protocol (FSMAP)

All IoT devices contain predefined AES keys to be used for the encryption algorithm. Likewise, pseudo-ID values, which are the identifying information of all sensors, are predefined on the sensors themselves and the IoT gateway.

The mathematical calculations of the FSMAP are given in Figure 4.6.

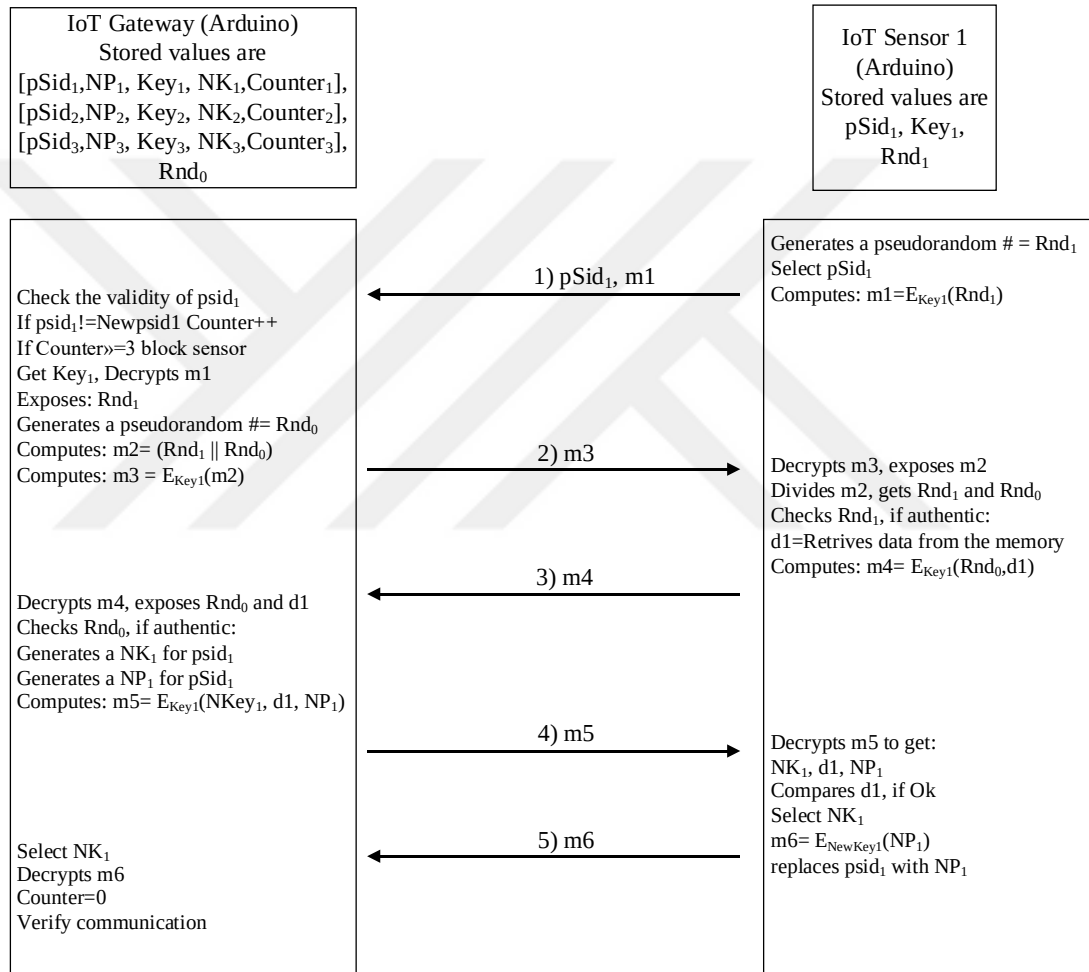


Figure 4.6 Mathematical calculations in our protocol design

The steps of the FSMAP are explained in detail below.

- *Step 1:* The sensor, which wants to transfer its data to the IoT gateway, sends the pseudo-ID (pSid₁) value as cleartext and the pseudo-random number value

(Rnd_1) that it produces as encrypted to the gateway to start the communication. With the $Psid_1$ value it has acquired, the gateway learns which sensor is applying to it and starts communication. It opens the encrypted Rnd_1 value using the Key_1 key and accesses its content.

- *Step 2:* The gateway sends encrypted Rnd_1 (received from the sensor) and Rnd_0 (Pseudo-random number generated by the gateway) information to the sensor. The sensor decrypts the Rnd_1 and Rnd_0 values using the Key_1 key. Trusts the gateway after confirming the Rnd_1 value is correct.
- *Step 3:* In this step, the sensor sends the pseudo-random number (Rnd_0) of the gateway and the sensor data it collects to the gateway by encrypting it with Key_1 . Afterward, the gateway decrypts its pseudo-random number (Rnd_0) and the sensor data using the Key_1 key. After confirming that the Rnd_0 information is correct, the gateway trusts the sensor. It also saves the sensor data in the same package to the database.
- *Step 4:* A new key (NK_1) and a new pseudo-ID (NP_1) were generated by the gateway to ensure that the encryption key used in the communication of the gateway and the sensor, and the pseudo-ID value of the sensor are changed in each round. Using the existing key (Key_1), the gateway sends the new encryption key (NK_1), the sensor data, and the new ID information of the sensor (NP_1) to the sensor to confirm the accuracy of the data received from the sensor. The sensor reads the new encryption key (NK_1) and the new pseudo-ID value (NP_1) in the package content it receives using the Key_1 key. It saves this information in the database. At the same time, it checks the accuracy of the data information and makes sure of the data stored in the gateway.
- *Step 5:* Finally, the sensor encrypts the NP_1 value with the NK_1 key and sends it to the gateway. The gateway reads the new ID information (NP_1) received from the sensor using the NK_1 key and confirms its accuracy. It makes sure that the NP_1 and NK_1 values are updated in the sensor.

If the communication is interrupted for any reason in the fourth step, the sensor's new pseudo-ID (NP_1) and the new key (NK_1) to be used for encryption will not reach

the sensor. If there is an interruption in the fifth step of the communication, the sensor will not be able to notify the gateway that it has received this information correctly. If the same sensor does not receive this information more than 3 times, the communication is terminated, which may mean that there is a problem. A counter is kept on the gateway to prevent another unauthorized device from trying to communicate with the gateway many times by capturing any sensor ID and to prevent unnecessary busyness as a result. When the gateway detects that it wants to connect with the old ID of the sensor in the first step of the communication, it will allow up to 3 attempts thanks to the counter. Afterward, it will block the relevant sensor ID for a certain period, preventing unnecessary occupation.

The random number generation of IoT devices often leads to the generation of numbers that can be repeated. In the designed system, the pseudo-random number is used to make the use of random number more efficient. Likewise, pseudo-ID is used to renew sensor IDs. Thus, it is prevented that the generated random numbers can be estimated by unauthorized persons.

AES is used as the encryption algorithm in the designed system. In today's conditions, the AES encryption algorithm is strong and difficult to break the algorithm. The encryption algorithm used in the mutual authentication protocol can be updated with any up-to-date encryption algorithm in the future to ensure its safe operation.

4.3 Testing Platform

The design, which provides secure data transmission between IoT devices, has been tested using the created smart textile prototype. Figure 4.7 shows the use of the smart textile device on the human arm. The smart textile device, which is worn on the human arm, provides the measurement of the arm's inner angle through sensors. By measuring the inner angle of the arm, the arm mobility of the elderly and sick people is determined and this data can be transferred wirelessly to other IoT devices.

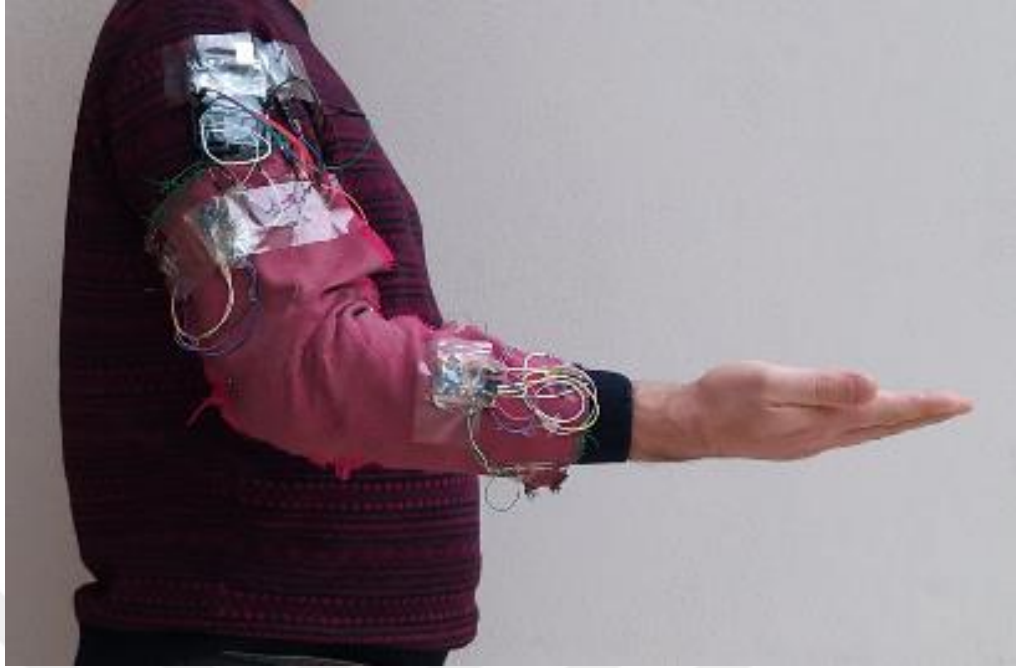


Figure 4.7 Designed smart textile prototype (Kişisel arşiv, 2021)

The sensor data transferred wirelessly to the IoT gateway can then be transferred to the cloud service in the smart city via the internet. Thus, the data can be collected in a central storage system and then used in decision support systems.

Figure 4.8 shows the architecture in which the data sent over the created prototype is sent to the IoT gateway and from there to the smart city cloud service. As seen in Figure 4.8, data communication takes place between two IoT devices, and the proposed design is used in this section. Data communication between IoT devices is done wirelessly.

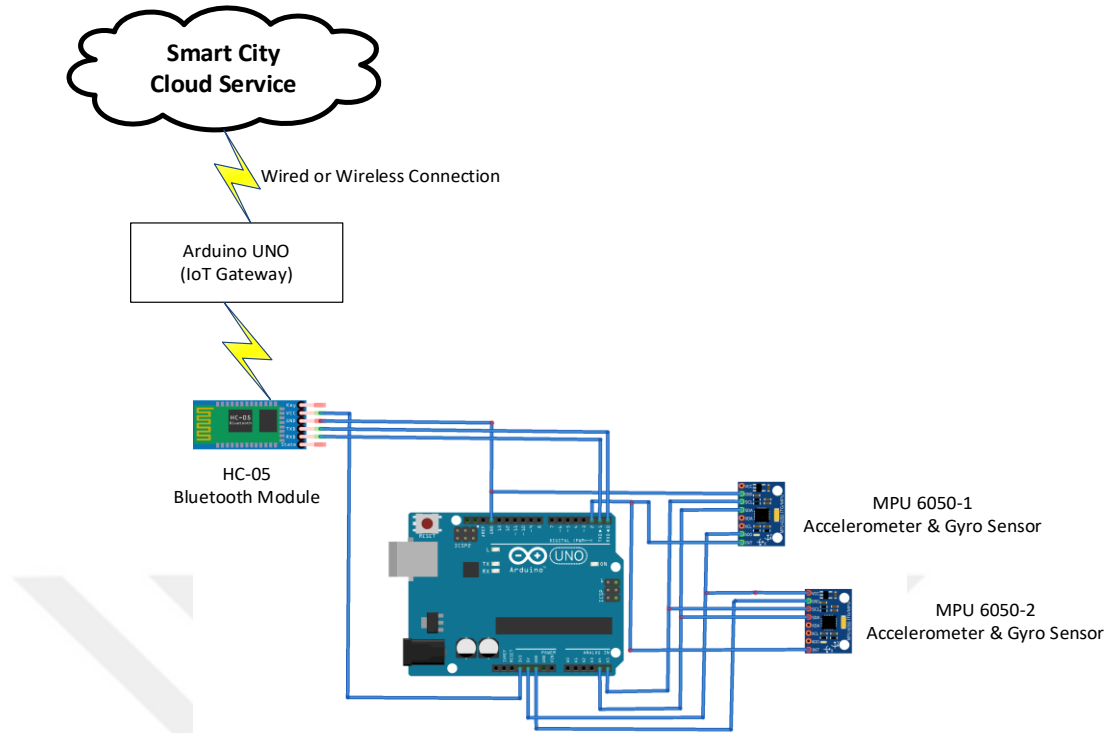


Figure 4.8 Proposed design hardware architecture

4.4 Testbed

A Windows application has been developed to send the data received from the sensors to the IoT gateway with the created smart textile device. The developed application was implemented in C# language and using Visual Studio. The application interface is shown in Figure 4.9.

With the developed Windows application, each step of the FSMAP is simulated one by one, and the operations performed on the screen are indicated. In addition, the direction of data flow is shown to the user in an animated way with the icon on the screen.

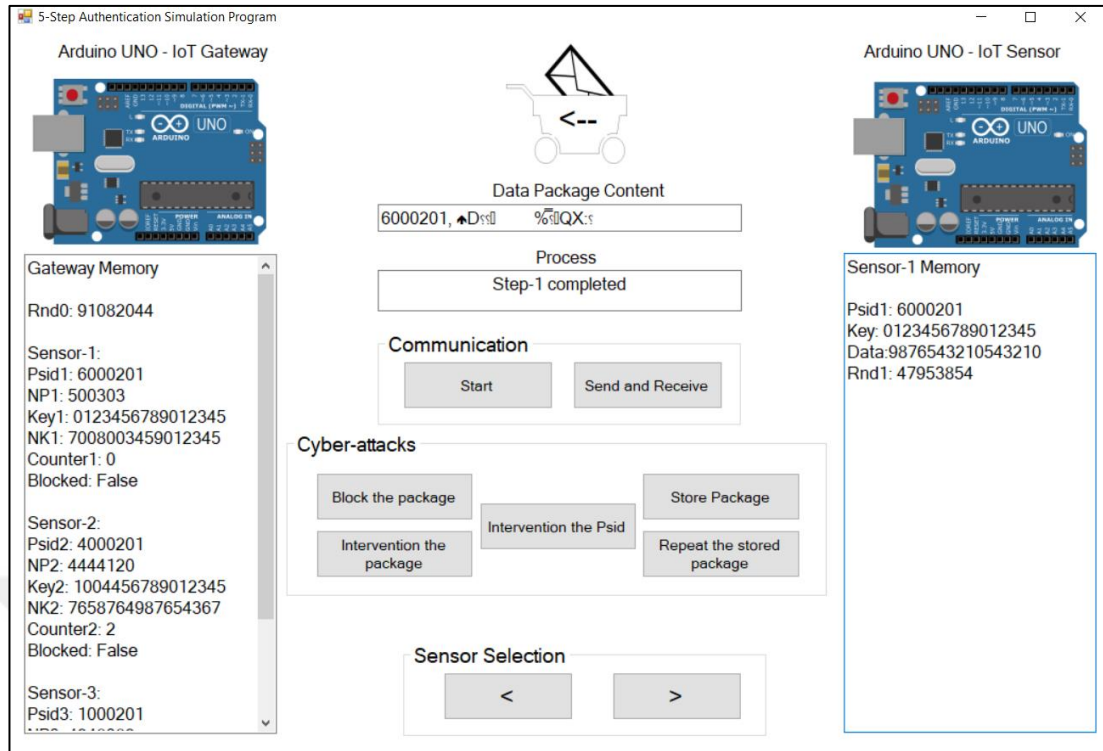


Figure 4.9 Windows application for IoT data communication

The IoT device on the smart textile wirelessly sends the sensor data to the IoT gateway using the FSMAP. During data communication, cyber-attack simulation is carried out by using the buttons in the area under the "cyber-attacks" title on the application. It is shown that when data communication is interfered with through these buttons, IoT devices terminate the protocol to protect themselves against attackers. Thus, there is no change/corruption in the data records in the IoT devices. Details about the application are given below.

On the left side of the application screen shown in Figure 4.9, there is the IoT gateway device picture and below it, the information on the IoT gateway content. On the right side of the application screen, there is the picture of the IoT device to which the sensor is connected and the information on the IoT device under it. In the upper-middle part of the application window, there is an icon symbolizing the data flow between devices. Inside this icon, there is an arrow sign indicating the data flow direction. Since the first data packet is sent while the application window shown in

Figure 4.9 is running, there is an arrow pointing to the left in the icon. In the textbox titled “data package content” under the data flow icon, the content of the data package transmitted between the IoT devices at the current authentication protocol step is displayed. As seen in the package content, the psid1 value is transmitted as cleartext, and rnd1 is transmitted as encrypted. In the textbox under the "process" heading, the step of the data communication is shown.

The functions of the buttons on the application are explained below.

- Left arrow in sensor selection section: It allows to switch to the previous IoT device to select the desired IoT device to which the sensor is connected.
- Right arrow in sensor selection section: It allows to switch to the next IoT device to select the desired IoT device to which the sensor is connected.
- Start: It is used to initialize the authentication protocol. The first step of the protocol takes place.
- Send and receive: It is used to advance each step when the authentication protocol starts working.
- Block the Package: It provides blocking of data communication between IoT devices. It stops the Authentication protocol from working.
- Intervention the package: It manipulates the contents of data packets transmitted between IoT devices. Because the package content is changed, IoT devices cannot verify the transmitted data and the authentication protocol ends.
- Intervention the Psid: The sensor ID used during data communication of IoT devices is changed by the attacker. Therefore, since the ID information is not verified by the other device, the authentication protocol ends.
- Store package: The data packet transmitted between IoT devices is captured and recorded by the attacker for later use.
- Repeat the stored package: It enables the data packet previously recorded by the attacker to be resent to the IoT devices. Since the Sensor ID and AES key used in each round of the authentication protocol are different, previously captured packets will not be valid. Therefore, since the package has expired, it cannot be used by the attacker.

The developed application has been tested by running as follows:

When the program is run, information about 3 different IoT sensors are defined in the IoT gateway as seen on the left side of the application screen. The content of the IoT sensor displayed on the right side of the screen is also included in the Arduino UNO.

Arduino UNO device is connected to the computer where the simulation program is installed with a USB cable. This device will act as an IoT gateway. To provide secure data communication between the smart textile device, the construction of which is described above, and the IoT gateway, the simulation was carried out by applying the following steps.

FSMAP was started by pressing the "Start" button under the Communication title within the Windows application shown in Figure 4.9. At the same time, the first packet was sent from the sensor to the gateway, which is the first step of the protocol. The operating steps of the authentication protocol are displayed in the section under the Process heading. Then, the "Send and Receive" button was pressed at each step to advance the other protocol steps in sequence. After the five steps were completed, the sensor data was transferred to the IoT gateway.

In addition, data communication was interfered with by using the buttons under the Cyber-attacks title during data communication. Devices are not affected by attacks because IoT device communications are not maintained when packets are blocked, manipulated, or the sensor ID is manipulated. In addition, when the packages are tried to be obtained and reused, it is seen that the attack is ineffective because the package content expires.

In the next chapter, formal and informal analysis has been carried out and it has been explained that the proposed protocol is reliable against cyber attacks.



CHAPTER 5

SECURITY ANALYSIS OF DESIGNED AUTHENTICATION METHOD

5.1 Overview

This section contains the analysis of FSMAF and the results of the analysis. Informal and formal analysis of the safety of the proposed protocol was carried out. AVISPA and Scyther tools were used to perform formal analysis. Afterward, performance and security comparisons were made with previous studies.

5.2 Informal Analysis

There are many cyber-attacks against IoT devices. The authentication protocol is used to prevent these attacks and the security of IoT devices is increased. Figure 5.1 shows the summary of cyber-attacks on IoT devices.

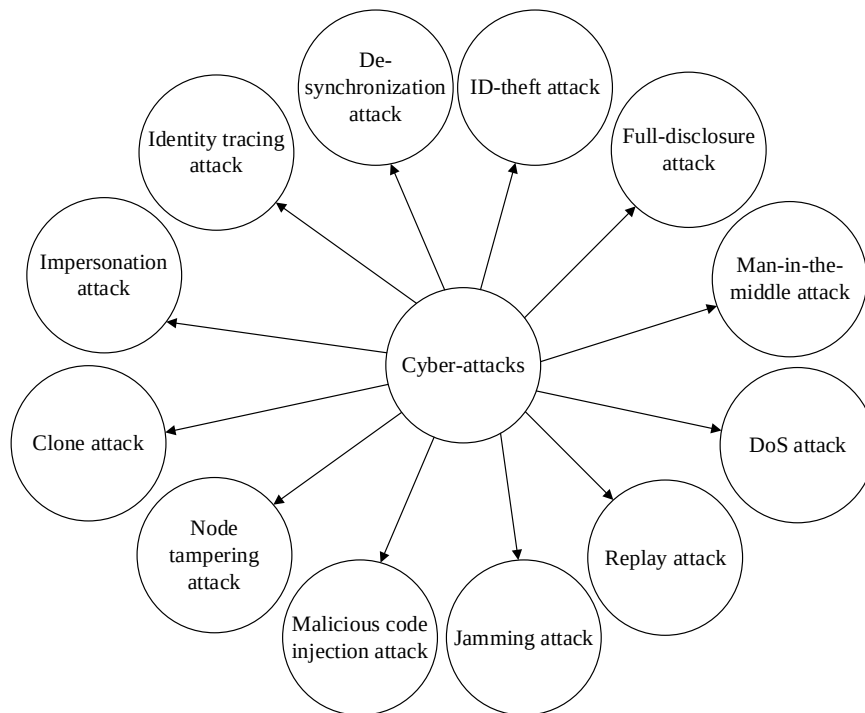


Figure 5.1 Summary of cyber-attacks in IoT

Cyber-attacks against IoT devices, methods of protection and how the work done is reliable against cyber-attacks are explained below.

5.2.1 IoT Attacks

5.2.1.1 Replay Attack

Capturing and reusing messaging between IoT devices is defined as a replay attack. In the proposed authentication protocol, a different encryption key is used in each session. In addition, the pseudo-ID of each sensor is renewed at the end of each session and updated on both IoT devices (IoT gateway-IoT Sensor). Therefore, since this information is renewed for each session, the proposed design is safe against replay attacks.

5.2.1.2 Denial of Service (DoS) Attack

DoS attack is defined as the fact that any device in the IoT system is constantly occupied and its resources are consumed quickly and cannot serve other IoT devices. Our proposed design includes measures to prevent the IoT gateway from serving other IoT devices by keeping it busy.

In the proposed design, the counter is kept on the IoT gateway. In cases where the fourth or fifth step of the authentication protocol cannot be completed, the IoT gateway will accept the same sensor ID and related encryption key up to 3 times. Therefore, an attacker who wants to keep the IoT gateway constantly busy using the same sensor ID will not be able to make communication requests more than 3 times. The recommended authentication protocol is secure against DoS attacks.

5.2.1.3 *Man-in-the-Middle Attack*

Man in the middle attack means that the messaging between IoT devices is eavesdropped by the attacker and sent to the receiving device as if they were transmitted from the real sender.

All packets in the proposed authentication protocol are transmitted encrypted between IoT devices. Therefore, since the contents of the transmitted packets cannot be read, it is not possible to change them. FS MAP is safe against man-in-the-middle attacks.

5.2.1.4 *Full-disclosure Attack*

Revealing information and secrets on IoT devices is defined as a full-disclosure attack. In the study, the AES encryption algorithm is used as the encryption method. AES is known to be a very powerful encryption algorithm. Therefore, the design is safe.

5.2.1.5 *ID-theft Attack*

The ID of the IoT device can be captured by the attacker. With this ID, the attacker can access the authorized areas of the IoT network.

In the study, sensor IDs were defined as pseudo-ID to prevent this attack. In addition, sensor IDs are renewed in each round of the authentication protocol. Thus, the design is safe against ID-theft attacks.

5.2.1.6 *De-synchronization Attack*

An attacker can interrupt communication between IoT devices. Interruption of data communication between IoT sensor and IoT gateway can cause synchronization issue. In the fifth step of the FS MAP, if the packet sent to the gateway does not arrive, the sensor can accept that these packets have arrived. There is a counter for each sensor on the gateway to avoid the synchronization problem. If the authentication protocol is interrupted, the same sensor may try to initiate communication three times in total.

The gateway sends NP_1 to the sensor in the fourth step of the authentication protocol. If this packet cannot be transmitted, the next time the sensor wants to initiate communication, it will refer to the gateway with the old $Psid_1$ instead of NP_1 . In case the sensor requests to initiate communication to the gateway with the same $Psid_1$, the counter value is incremented by one. However, when the fifth step is completed, the counter on the network of the relevant sensor is reset. The sensor's request to connect to the gateway with the old $Psid_1$ can be up to three times. When it is desired to communicate with the old $Psid_1$ after these three attempts, the $Psid_1$ value of the relevant sensor is blocked on the gateway. During the operation of the authentication protocol, each step is interconnected. Thus, the gateway and the sensor remain in sync.

5.2.1.7 *Identity Tracing Attack*

Tracing the IDs of IoT devices can cause security problems. ID tracing and capture can be considered. The obtained ID information may be desired to be used later by the attacker.

In our design, the pseudo-ID of the sensor is renewed at the end of each communication with the IoT gateway. Since the pseudo-ID of the sensor is renewed at

the end of each round, it cannot be used later if captured. Therefore, the recommended protocol is secure against identity tracing attacks.

5.2.1.8 Impersonation Attack

Pseudo-IDs of sensors are sent as clear text during data communication. At the same time, pseudo-IDs are renewed every round. The attacker may want to guess the renewed sensor ID by collecting the sensor IDs. But since sensor IDs are pseudo-ID, it is not possible to impersonate a sensor. Turkanović, Brumen & Hölbl (2014) have proposed an authentication scheme. In their study, Amin & Biswas (2016) stated that the protocol presented by Turkanović et al. (2014) is vulnerable to impersonation attacks.

5.2.1.9 Clone Attack

The attacker may want to clone the sensors in the IoT network. Thus, the access authority of the cloned sensor will be available to the attacker. In the proposed design, since the sensor ID is renewed every round, the captured sensor ID will be invalid for later use. So the clone attack will be invalid.

5.2.1.10 Node Tampering Attack

In cases where the attacker can physically access the devices in the IoT network, they can read or change the information inside the device. The proposed design is not a precaution against physical attacks. However, in the proposed design, since the data will be stored encrypted on the IoT device, the content of the data will not be accessible if the data is accessed.

5.2.1.11 Malicious Code Injection Attack

An attacker can access the IoT device and inject malicious code. Therefore, the IoT device can become manageable by the attacker. In our proposed design, mutual authentication is made between IoT devices. Therefore, devices must complete the authentication protocol in order to communicate with each other. Our proposed design is safe from malicious code injection attacks.

5.2.1.12 Jamming Attack

Wireless communications of IoT devices can be intercepted by the attacker. This can cause security issues. The proposed mutual authentication protocol has no function against jamming attacks. However, in the case of a jamming attack in the proposed protocol, after the communication between IoT devices is interrupted, an unauthorized user may constantly communicate with the IoT gateway to prevent the IoT gateway from serving (DoS attack). The FSMAP keeps counters for unauthorized access that will prevent the IoT gateway from serving. This counter blocks the access of the relevant device to the IoT gateway after three attempts.

5.3 Formal Analysis

Formal analysis of the proposed authentication protocol was performed using AVISPA and Scyther tools. Analysis and analysis results with both tools are explained below.

5.3.1 AVISPA as a Formal Analysis Tool

AVISPA is one of the commonly used tools for validating authentication protocols. There are many studies that provide verification of authentication protocols using AVISPA (Mohanta, Jena & Patnaik, 2021; Sadhukhan, Ray, Biswas, Khan & Dasgupta, 2021; Thakur et al., 2021; Zargar, Shahidinejad & Ghobaei-Arani, 2021). The protocol to be verified with AVISPA must be implemented by encoding with the High-Level Protocol Specification Language (HLPSL). There are four backends integrated via HLPSL. These are On- The-Fly-Model-Checker (OFMC), Constraint-Logic- based Attack Searcher (CL- AtSe), State-of-the-Art Solver (SAT), and Tree Automata based on Automate Approximation for Analysis of Security Protocol (TA4SP).

The FSMAF was coded using the HLPSL language and simulated with SPAN, the security protocol animator for AVISPA. The SPAN 1.6 protocol verification application used in Figure 5.2 is shown. The graphical interface shown in Figure 5.2, that is a modified version of the figure in (Genet, 2015), is explained with numbers and explained below.

- (1) It is used to open/save a CAS+/HLPSL specification file from within the File menu.
- (2) It is used to save the edited CAS+/HLPSL file.
- (3) Used to reload CAS+.
- (4) Used to reload HLPSL.
- (5) It is used for the protocol animator.
- (6) It is used for the protocol animator with the intruder.
- (7) It is used to perform automatic construction of the attack simulation.
- (8) This is the area where CAS+/HLPSL specification files can be edited.
- (9) This is the section where one of the verification tools is selected.
- (10) Used to execute the CAS+/HLPSL specification on the selected validation tool.

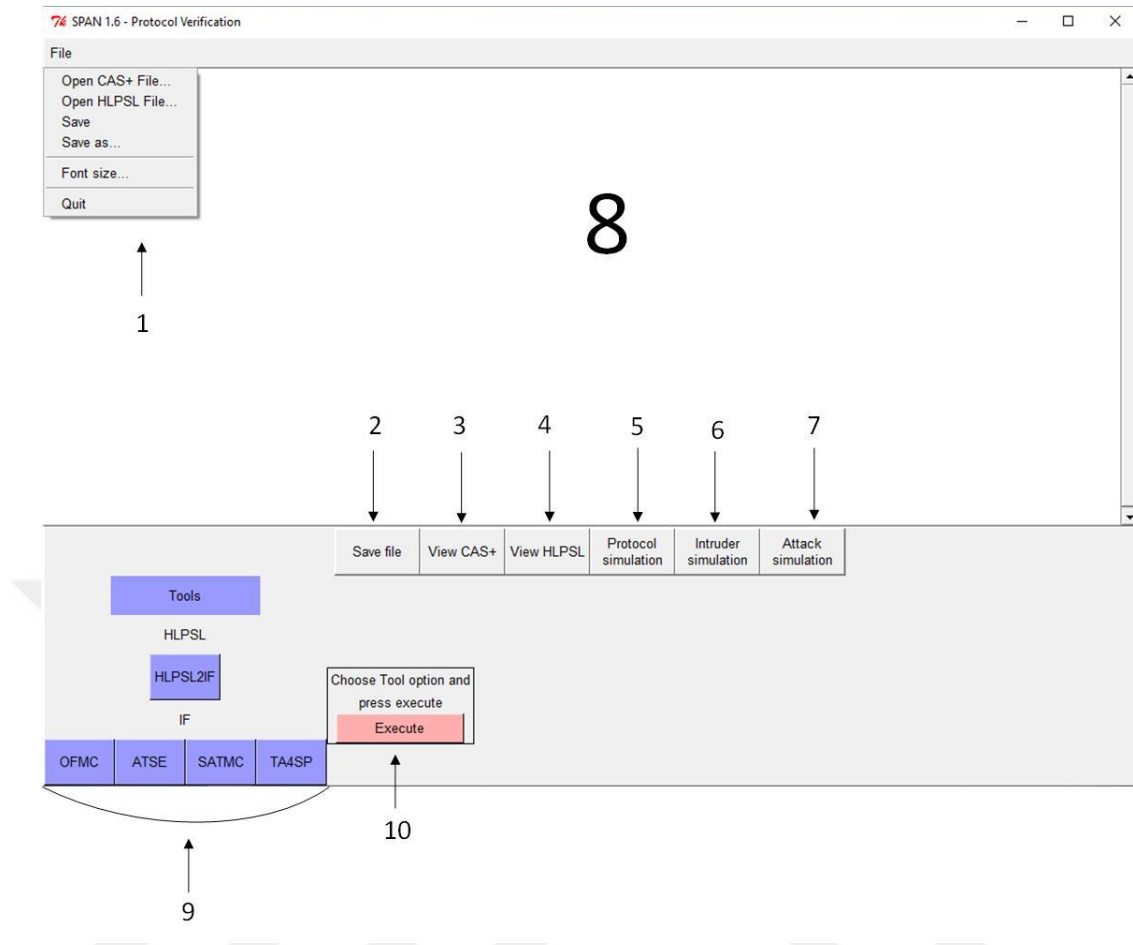


Figure 5.2 SPAN tool for AVISPA

AVISPA checks the submitted protocol and warns when it finds an attack against the protocol. Below are some commands used in the HLPSP language.

- `Rcv(message)` : used to receive messages.
- `Snd(message)` : used to send messages.
- `variable' := expression` : It is used to assign a value to the specified variable.
- `variable' :=new()` : used to generate fresh information.

Verifying our protocol using the AVISPA tool is described below.

5.3.1.1 Verification of Our Proposed Protocol Using AVISPA

Our protocol's HLPSL implementation was based on two roles: IoTsensor and IoTgateway. Figure 5.3 shows the IoT sensor role codes in the HLPSL language.

```

role sensor(
  IoTsensor, IoTgateway : agent,
  Key1 : symmetric_key,
  Hash : hash_func,
  SND, RCV : channel(dy))

played_by IoTsensor def=

  local
    State : nat,
    Rnd1, Rnd0 : nat,
    PSid1: text,
    Data: text,
    NP1: text,
    NK1 : message

  init
    State := 0

  transition

    1. State = 0 /\ RCV(start) =|>
       State' := 2 /\ Rnd1' := new()
       /\ SND(PSid1'.{Rnd1'}_Key1)

    2. State = 2 /\ RCV({Rnd1'.Rnd0'}_Key1) =|>
       State' := 4 /\ Data' := new()
       /\ SND(PSid1'.{Rnd0'.Data}_Key1')
       /\ NK1' := Hash(Rnd1'.Rnd0')
       /\ secret(NK1', nk1, {IoTgateway, IoTsensor})

    3. State = 4 /\ RCV({NK1'.Data'.NP1'}_Key1) =|>
       State' := 6 /\ SND({NP1'}_NK1')
       /\ request(IoTsensor, IoTgateway, gateway_sensor_Rnd0, Rnd0)

end role

```

Figure 5.3 IoT sensor role implementation

Figure 5.4 shows the IoT gateway role codes in the HLPSL language.

```

role gateway(
  IoTsensor,IoTgateway : agent,
  Key1 : symmetric_key,
  Hash : hash_func,
  SND,RCV : channel(dy))

played_by IoTgateway def=

  local
    State : nat,
    Rnd0,Rnd1 : nat,
    PSid1: text,
    Data: text,
    NP1:text,
    NK1 : message

  init
    State := 1

  transition
  1. State = 1 /\ RCV(PSid1'.{Rnd1'}_Key1) =|>
    State' := 3 /\ Rnd0' := new()
    /\ SND({Rnd1'.Rnd0'}_Key1)

  2. State = 3 /\ RCV(PSid1'.{Rnd0.Data'}_Key1) =|>
    State' := 5 /\ NP1' := new()
    /\ NK1' := Hash(Rnd1'.Rnd0')
    /\ SND({NK1'.Data'.NP1'}_Key1)
    /\ witness(IoTgateway,IoTsensor,gateway_sensor_Rnd0,Rnd0')

  3. State = 5 /\ RCV({NP1'}_NK1) =|>
    State' := 7

end role

```

Figure 5.4 IoT gateway role implementation

The codes written for the IoT sensor and IoT gateway are explained below in parts.

As shown in Figure 5.5, firstly, a role named sensor is defined. The protocol steps implemented on the sensor side take place in this role. The role definition has parameters. IoTsensor and IoTgateway parameters are defined in the agent type, IoTsensor represents the sensor and IoTgateway represents the gateway. The Key1 parameter defines the key to be used in symmetric encryption. The hash function is used to update the key used for the encryption algorithm. In this section, a hash function definition is made. SND and RCV parameters are channel type. These refer to the channels through which the agent acting as the sensor will communicate. The attribute of the channel type (dy) indicates the intruder model of the channel. All variables in HLPSL start with a capital letter.

```

role sensor(
    IoTsensor, IoTgateway : agent,
    Key1 : symmetric_key,
    Hash : hash_func,
    SND, RCV : channel(dy))

```

Figure 5.5 HLPSL code role definition for IoT sensor

As shown in Figure 5.6, the parameter IoTsensor is used in the played_by section. This means that IoTsensor indicates the name of the agent that plays the sensor role. In the Local section, local variables related to the sensor role are defined. State, Rnd1, and Rnd0 variables are of type nat (a natural number). Psid1 (Sensor ID), data (data to be transmitted by the sensor) and NP1 (new sensor ID) are of text type. The key to be updated, Key1 (NK1), is of the message type. The state variable, which specifies the protocol steps, is set to zero in the init section and its initial state is determined.

```

played_by IoTsensor def=
    local
        State : nat,
        Rnd1, Rnd0 : nat,
        PSid1: text,
        Data: text,
        NP1:text,
        NK1 : message
    init
        State := 0

```

Figure 5.6 HLPSL code local variables definition for IoT sensor

As seen in Figure 5.7, the transition section contains the set of transition. This set represents receiving and sending messages. Each transition has a name and this name separates the transitions from each other. The /\ operator means that the specified roles are executed together.

First of all, with the RCV command, the start message is received, indicating the beginning of the execution of the protocol. The start message is therefore only available in one role. Then, the Rnd1 variable is created. Then, to perform the first step of the FS MAP, the Psid1 value is sent to the receiving side as clear text with the SND command. In the SND command content, the Rnd1 value is sent encrypted with Key1.

```

transition
1. State = 0 /\ RCV(start) =|>
   State' := 2 /\ Rnd1' := new()
               /\ SND(Psid1'.{Rnd1'}_Key1)

2. State = 2 /\ RCV({Rnd1'.Rnd0'}_Key1) =|>
   State' := 4 /\ Data' := new()
               /\ SND(Psid1'.{Rnd0'.Data}_Key1')
               /\ NK1' := Hash(Rnd1'.Rnd0')
               /\ secret(NK1', nk1, {IoTgateway, IoTsensor})

3. State = 4 /\ RCV({NK1'.Data'.NP1'}_Key1) =|>
   State' := 6 /\ SND({NP1'}_NK1')
               /\ request(IoTsensord, IoTgateway, gateway_sensord_Rnd0, Rnd0)
end role

```

Figure 5.7 HPSL code transition section for IoT sensor

In case number 2, the packet coming from the gateway with the RCV command is decrypted with Key1, and Rnd1 and Rnd2 values are obtained. Then, the Data variable, which represents the data to be sent by the sensor, is created. Using the SND command, the Psid1 value is sent as cleartext. In the same packet, Rnd0 and data are encrypted with Key1 and sent to the gateway. The new encryption key is generated via the Hash function and equated to the NK1 value. The secret command is used to control the secrecy of the NK1 value to be transmitted between the gateway and the sensor.

In case number 3, the packet received from the gateway with the RCV command is decrypted with Key1, and NK1, data, and NP1 values are obtained. Then, via the SND command, the NP1 encrypted with the new encryption key (NK1) is sent to the gateway. The request event is used for authentication. The request event is used to make sure that the Rnd0 value sent by the gateway to the sensor is actually created to be sent to the sensor by the gateway. The sensor role is completed with the end role command.

Figure 5.8 shows the gateway role. The role name is specified as a gateway with its parameters. It is defined as IoTsensor and IoTgateway agent type. Key1 parameter is defined as symmetric key in gateway role. In addition, hash function, SND, and RCV parameters are also defined.

```
role gateway(
    IoTsensor, IoTgateway : agent,
    Key1 : symmetric_key,
    Hash : hash_func,
    SND, RCV : channel(dy))
```

Figure 5.8 HLPsL code role definition for IoT gateway

The IoTgateway parameter shown in Figure 5.9 is used in the played_by section, meaning that it displays the name of the agent that plays the gateway role. Local variables of the IoTgateway role are defined in the Local section. State, Rnd0, and Rnd1 variables are defined as type nat. Psid1 and data are defined as text type. NK1 is defined as message type. In the init section, the state value is specified as 1.

```
played_by IoTgateway def=

    local
        State : nat,
        Rnd0, Rnd1 : nat,
        PSid1: text,
        Data: text,
        NP1: text,
        NK1 : message

    init
        State := 1
```

Figure 5.9 HLPsL code local variables definition for IoT gateway

Figure 5.10 shows the transition section in the gateway role. First of all, the Psid1 value sent from the sensor with the RCV command is taken as cleartext and it is determined which sensor wants to communicate. The Rnd1 value sent in the same

package is also decrypted with Key1. Then the Rnd0 variable is created. Rnd1 and Rnd0 values encrypted with Key1 are sent to the sensor using the SND command.

In case number 2, the Psid1 value coming from the sensor as cleartext is taken. Rnd0 and data variables in the same package are decrypted with Key1. A variable named NP1 is created for the sensor. Then NK1 is created with the hash function. NK1 will be used as the new encryption key. NK1, data, and NP1 variables are encrypted with Key1 and sent to the sensor using the SND command. Witness event is used for authentication. The IoTsensor authenticates IoTgateway on the value of Rnd0.

```

transition
1. State = 1 /\ RCV(Psid1'.{Rnd1'}_Key1) =|>
   State' := 3 /\ Rnd0' := new()
               /\ SND({Rnd1'.Rnd0'}_Key1)

2. State = 3 /\ RCV(Psid1'.{Rnd0.Data'}_Key1) =|>
   State' := 5 /\ NP1' := new()
               /\ NK1' := Hash(Rnd1'.Rnd0')
               /\ SND({NK1'.Data'.NP1'}_Key1)
               /\ witness(IoTgateway,IoTsensor,gateway_sensor_Rnd0,Rnd0')

3. State = 5 /\ RCV({NP1'}_NK1) =|>
   State' := 7

end role

```

Figure 5.10 HLPSTL code transition section for IoT gateway

In case 3, NP1 received from the sensor with the RCV command is decrypted with the NK1 key.

Authentication protocol was analyzed using the OFMC backend. The analysis result of FSMAP with SPAN is shown in Figure 5.11.

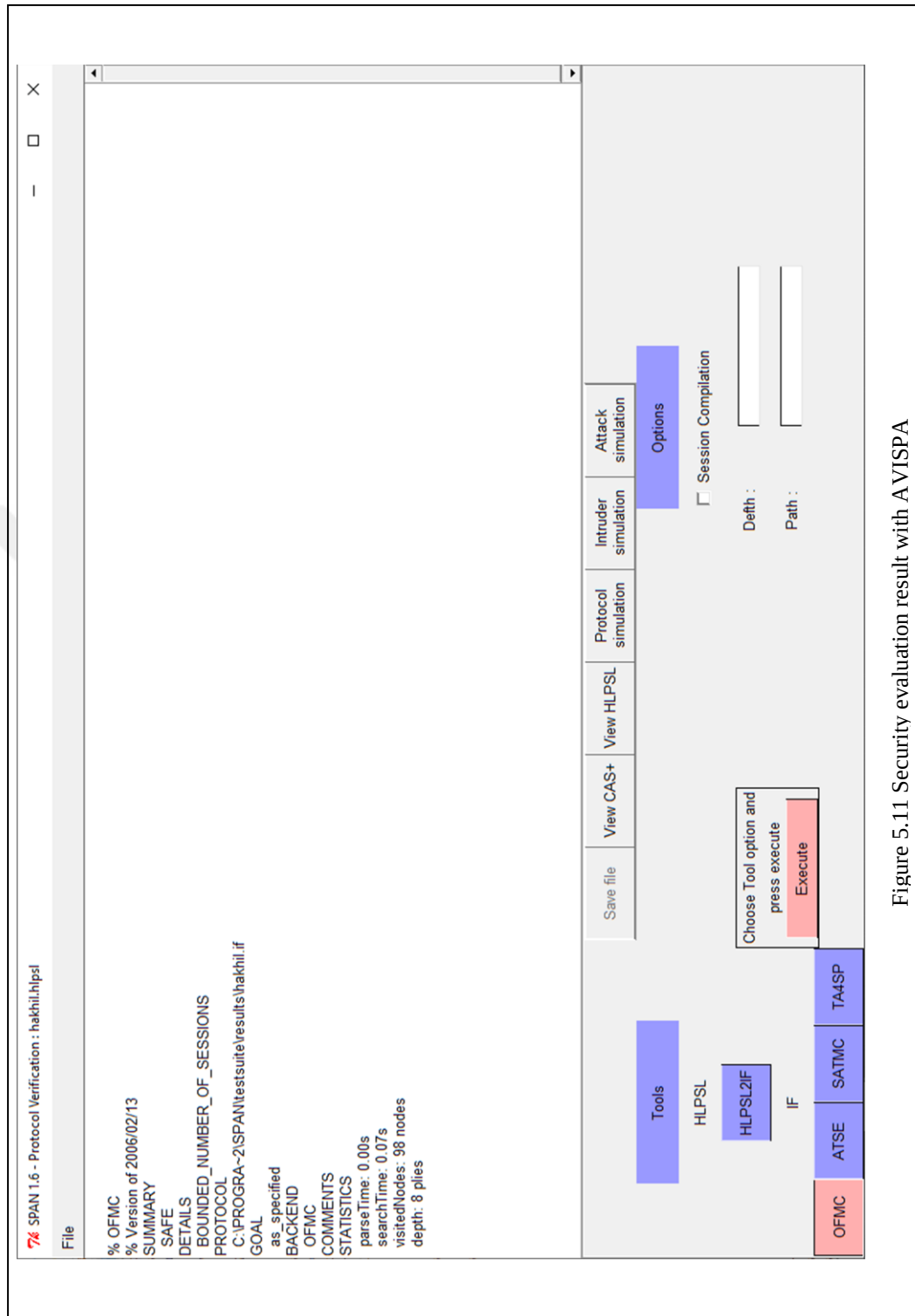


Figure 5.11 Security evaluation result with AVISPA

The result on line 4 shows that our proposed mutual authentication protocol is 'SAFE' against attacks tested by AVISPA. Details of the protocol analysis are summarized after the 'SAFE' declaration of the tool. The tool confirms that our protocol had a bounded number of checker sessions. The location of the protocol HPSL code file is also given in the results to ensure the user about the description analyzed.

5.3.2 Scyther as a Formal Analysis Tool

The Scyther tool has been used in recent years to verify authentication protocols. In the literature search carried out, there are many studies with Scyther (El Madhoun, Bertin, Badra & Pujolle, 2021; Navas & Toutain, 2018; Yıldırım, Dalkılıç & Duru, 2019). Scyther v1.1.3 Windows version, seen in Figure 5.12, was used in our study. For the Scyther program to work, the following programming language and plugins must be installed on the computer.

- The Graphviz library (Graph visualization Software)

A library used by the Scyther tool to draw graphs. Graphviz is open-source graph visualization software. Graphical visualization allows us to represent structural information as diagrams of abstract graphs. Graphviz 2.44.2 version was used in our Scyther tool installation.

- Python (Programming language)

Scyther's graphical user interface is written in Python. The Scyther tool used does not support Python 3. Python 2.7.18 version has been installed in order to use Scyther.

- wxPython libraries (the cross-platform GUI toolkit for the Python language)

Scyther's graphical user interface uses the wxpython library to draw widgets. WxPython enables software developers to create user interfaces for their

Python applications running on Windows and other similar systems. To use the Scyther tool, wxPython version 2.8 has been installed.

After the Python programming language and related plugins are installed, the files of the Scyther program are downloaded from the relevant website and run with the scyther-gui.py file from the folder where it is copied to the computer (Cremers, 2021).

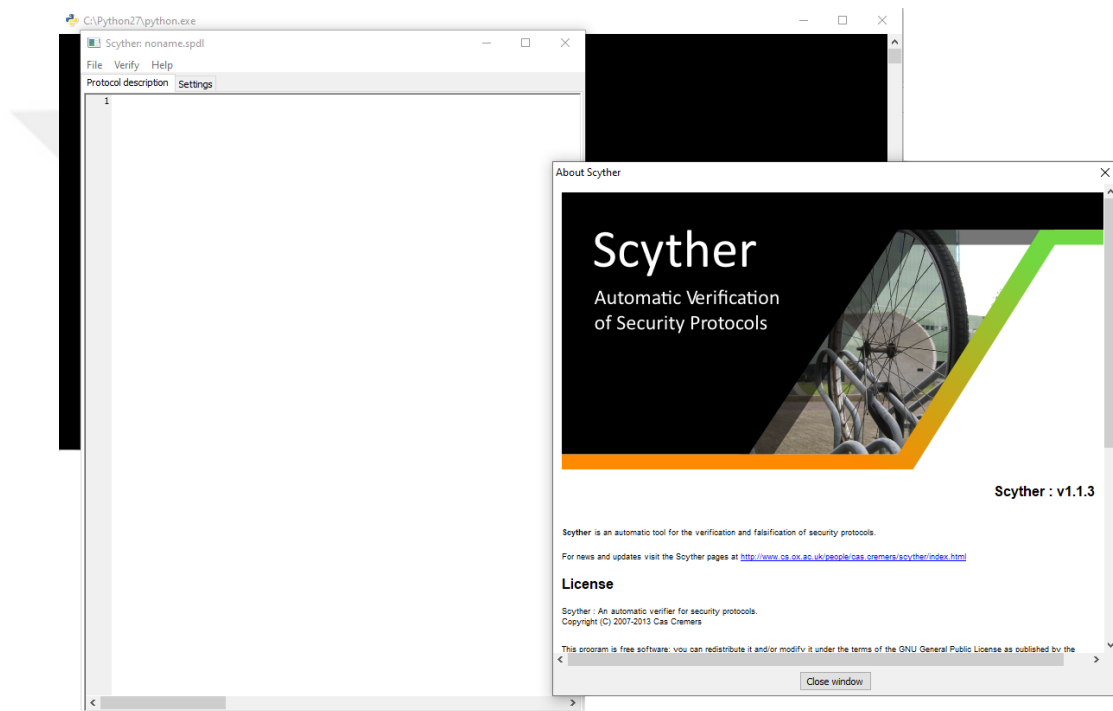


Figure 5.12 Scyther tool

The created authentication protocol was coded using the Security Protocol Description Language (SPDL) within the Scyther program. The Scyther definitions used in coding are explained below.

- Usertype: It is used to define a new type. The usage of the command is as follows.

```
usertype newmessage;
```

- Protocol: It is used to name the protocol to be modeled and to identify the parties of the protocol. Parties are specified as parameters. The usage example is as follows:

```
protocol ExampleProtocol(Sensor, Gateway) { };
```

- Role: It is the section where data sending and receiving events are made between the parties. Send and receive events are performed by creating a separate role for each party defined with the protocol command. Its usage is as follows:

```
role Sensor { };
```

```
role gateway { };
```

- Fresh: Random numbers are generated in security protocols. It can be done in role definition using fresh declarations. The usage example is as follows:

```
fresh X:Nonce;
```

- Var: Parties can use variables to store the information they receive. Variables can be defined using this command. Usage example:

```
var X:Nonce;
```

- Symmetric key: any term can be used as a key for symmetric encryption. Encrypting the Value with the key term is as follows:

```
{value}key
```

- Send: It is used for sending messages between parties.
- Recv: It is used to receive the message sent from the other party.
- Match: Used to specify pattern matching.
- Claim: It is used to indicate a role in modeling the intended security features. The claim event in the example below means that the value is hidden.

```
claim(Sensor, Secret, value);
```

- Running: Used to specify data agreement.

- Secret: It is used to control the secrecy of the variable specified by the parameter.
- Global declarations: There are global constants as they are used in many applications. Labels, string constants, and protocol definitions are shown in Figure 5.13.

```

usertype String;
const text1: String;

protocol Hakhil(Sensor, Gateway)
{
    role Sensor
    {
        send_1(Sensor, Gateway, text1);
    }

    role Gateway
    {
        recv_1(Sensor, Gateway, text1);
    }
}

```

Figure 5.13 Global declarations

- Aliveness (Alive): It is a form of authentication that aims to determine that communication partners are alive.
- Weak agreement (Weakagree): ensures that the protocol is safe against impersonation attacks (Nikooghadam & Amintoosi, 2020).
- Session-key reveal (SKR): Used to mark the parameter term as an additional session key. As a result, the use of the SKR rule will reveal the parameter term.
- Non-injective agreement (Niagree): It is used to ensure that the protocol work between the two parties is completed and that the content of the exchanged messages is the same. Thus, it is guaranteed that the data is transmitted without corruption and data integrity is ensured.

- Non-injective synchronization (Nisynch): It is used to guarantee that all received messages are sent by the sender, the communication partner, and received by the receiver.

5.3.2.1 Verification of Our Proposed Protocol Using Scyther

The code developed to simulate the FSMAP is shown in Figure 5.14. With the code written in Scyther, only 1 round of the proposed protocol is proven by running. Since the Scyther tool does not run the authentication protocol continuously, it tests it in a single round. Therefore, sensor pseudo-ID and new AES key values that change in each round in our proposed protocol cannot be proven in Scyther because they are used in the next round. In addition, counters are kept in the proposed protocol. This counter contains a variable used for more than one round. Since Scyther does not run the authentication protocol more than one round, no coding has been done for the counter.

```

usertype String;

protocol Hakhil(Sensor, Gateway)
{
  role Sensor
  {
    fresh Rnd1,psid1:Nonce;
    var Rnd0,Nk1,Np1:Nonce;
    fresh Sensordata: String;

    send_1(Sensor,Gateway, psid1,{Rnd1}k(Sensor,Gateway) );
    recv_2(Gateway,Sensor, {Rnd1,Rnd0}k(Sensor,Gateway) );
    match(Rnd1,{Rnd1}k(Sensor,Gateway));
    send_3(Sensor,Gateway, psid1,{Rnd0,Sensordata}k(Sensor,Gateway) );
    recv_4(Gateway,Sensor, {Nk1,Sensordata,Np1}k(Sensor,Gateway) );
    send_5(Sensor,Gateway, {Np1}Nk1(Sensor,Gateway) );

    claim(Sensor,Secret,Rnd1);
    claim(Sensor,Secret,Rnd0);
    claim(Sensor,Secret,Sensordata);
    claim(Sensor,Secret,Nk1);
    claim(Sensor,Secret,Np1);
    claim(Sensor,Alive);
    claim(Sensor,Weakagree);
    claim(Sensor,SKR,k);
    claim(Sensor,Niagree);
    claim(Sensor,Nisynch);
  }

  role Gateway
  {
    var Rnd1,psid1: Nonce;
    fresh Rnd0,Nk1,Np1: Nonce;
    var Sensordata: String;

    recv_1(Sensor,Gateway,psid1, {Rnd1}k(Sensor,Gateway) );
    send_2(Gateway,Sensor, {Rnd1,Rnd0}k(Sensor,Gateway) );
    recv_3(Sensor,Gateway, psid1,{Rnd0,Sensordata}k(Sensor,Gateway) );
    match(Rnd0,{Rnd0}k(Sensor,Gateway));
    send_4(Gateway,Sensor, {Nk1,Sensordata,Np1}k(Sensor,Gateway) );
    recv_5(Sensor,Gateway, {Np1}Nk1(Sensor,Gateway) );

    claim(Gateway,Secret,Rnd1);
    claim(Gateway,Secret,Rnd0);
    claim(Gateway,Secret,Sensordata);
    claim(Gateway,Secret,Nk1);
    claim(Gateway,Secret,Np1);
    claim(Gateway,Alive);
    claim(Gateway,Weakagree);
    claim(Gateway,SKR,k);
    claim(Gateway,Niagree);
    claim(Gateway,Nisynch);
  }
}

```

Figure 5.14 The Scyther code of the FSMAP

The code shown in Figure 5.14 is explained below in sections.

As shown in Figure 5.15, at the beginning of the coding, the "String" type is defined with the usertype command. Then, with the "protocol" parameter, the names of the devices that perform mutual authentication (Sensor, Gateway) and the name of the protocol (Hakhil) are defined. Afterward, a "role" definition was made for the sensor. Variable definitions are made inside the sensor role. The variable definitions of the sensor are made with the "fresh" and "var" commands. Rnd1 and psid1 variables are defined as "nonce" type using the fresh command. Then, the variables Rnd0, Nk1, and Np1 are defined as "nonce" using the var command. Afterward, the variable sensordata is defined as string type.

```
usertype String;

protocol Hakhil(Sensor, Gateway)
{
  role Sensor
  {
    fresh Rnd1,psid1:Nonce;
    var Rnd0,Nk1,Np1:Nonce;
    fresh Sensordata: String;
```

Figure 5.15 Scyther code for variable declarations

As shown in Figure 5.16, the messages sent and received between devices are defined during the five-step authentication with the "send" and "recv" parameters. The first data transmission is started with the send_1 command used in the sensor role. With the parameters used, it is stated from which role to which role the message is transferred and the content of the message. With the Recv_2 command, the message sent from the gateway is received. With the parameters, the roles in which the message is transmitted and received are specified. It is also defined as a parameter in the received message.

The authentication study is carried out using symmetric encryption. $\{Rnd1\}_k$ notation is used to encrypt the Rnd1 value using the symmetric key. With the match command, it is verified that the Rnd1 value received from the gateway matches the sent Rnd1 value.

```

send_1(Sensor, Gateway, psid1, {Rnd1}k(Sensor, Gateway) );
recv_2(Gateway, Sensor, {Rnd1, Rnd0}k(Sensor, Gateway) );
match(Rnd1, {Rnd1}k(Sensor, Gateway));
send_3(Sensor, Gateway, psid1, {Rnd0, Sensordata}k(Sensor, Gateway) );
recv_4(Gateway, Sensor, {Nk1, Sensordata, Np1}k(Sensor, Gateway) );
send_5(Sensor, Gateway, {Np1}Nk1(Sensor, Gateway) );

```

Figure 5.16 Scyther code for data transfer on the sensor side

The security features to be modeled on the sensor side are defined as shown in Figure 5.17 Claim command is used with the secret parameter to determine the secrecy of Rnd1, Rnd0, sensordata, nk1, and np1 variables. When the code is run, each claim result on the sensor side is displayed the same on the Scyther results screen.

```

claim(Sensor, Secret, Rnd1);
claim(Sensor, Secret, Rnd0);
claim(Sensor, Secret, Sensordata);
claim(Sensor, Secret, Nk1);
claim(Sensor, Secret, Np1);
claim(Sensor, Alive);
claim(Sensor, Weakagree);
claim(Sensor, SKR, k);
claim(Sensor, Niagree);
claim(Sensor, Nisynch);
}

```

Figure 5.17 Scyther code for security properties on the sensor side

The definition for the variables to be used by the Gateway is done inside the Gateway role as shown in Figure 5.18. With the “var” command, the “nonce” type Rnd1 and psid1 variables are defined. With the “fresh” command, the “nonce” type Rnd0, nk1, and np1 variables are defined. With the “var” command, a “string” type sensordata variable is defined.

```

role Gateway
{
    var Rnd1, psid1: Nonce;
    fresh Rnd0, Nk1, Np1: Nonce;
    var Sensordata: String;
}

```

Figure 5.18 Scyther code for data transfer on the gateway side

As shown in Figure 5.19, send and rcv commands are used to define the messages sent between the sensor and the gateway. Since the FSMap starts the communication with the sensor sending data to the gateway, the communication codes on the gateway side start with the rcv command. Thus, the gateway starts communication by receiving the message from the sensor. With the match command, the match of the Rnd0 value coming from the sensor with the Rnd0 value sent to the sensor is checked and confirmed.

```
rcv_1(Sensor, Gateway, psid1, {Rnd1}k(Sensor, Gateway) );
send_2(Gateway, Sensor, {Rnd1, Rnd0}k(Sensor, Gateway) );
rcv_3(Sensor, Gateway, psid1, {Rnd0, Sensordata}k(Sensor, Gateway) );
match(Rnd0, {Rnd0}k(Sensor, Gateway) );
send_4(Gateway, Sensor, {Nk1, Sensordata, Np1}k(Sensor, Gateway) );
rcv_5(Sensor, Gateway, {Np1}Nk1(Sensor, Gateway) );
```

Figure 5.19 Variable definitions Scyther codes used by the gateway

The security features to be modeled on the gateway side are defined as shown in Figure 5.20. Claim parameter is used to determine whether the secrecy of Rnd1, Rnd0, sensordata, nk1, and np1 variables is provided. When the code is run, each claim result on the gateway side is displayed the same on the Scyther results screen.

```
claim(Gateway, Secret, Rnd1);
claim(Gateway, Secret, Rnd0);
claim(Gateway, Secret, Sensordata);
claim(Gateway, Secret, Nk1);
claim(Gateway, Secret, Np1);
claim(Gateway, Alive);
claim(Gateway, Weakagree);
claim(Gateway, SKR, k);
claim(Gateway, Niagree);
claim(Gateway, Nisynch);
}
```

Figure 5.20 Scyther code for security properties on the gateway side

As a result of the coding, the result was obtained as in Figure 5.21. The status of each claim is shown separately with the coding written in the Scyther program.

Scyther results : verify

Claim				Status		Comments
Hakhil	Sensor	Hakhil,Sensor 1	Secret Rnd1	Ok	Verified	No attacks.
		Hakhil,Sensor2	Secret Rnd0	Ok	Verified	No attacks.
		Hakhil,Sensor3	Secret Sensordata	Ok	Verified	No attacks.
		Hakhil,Sensor4	Secret Nk1	Ok	Verified	No attacks.
		Hakhil,Sensor5	Secret Np1	Ok	Verified	No attacks.
		Hakhil,Sensor6	Alive	Ok	Verified	No attacks.
		Hakhil,Sensor7	Weakagree	Ok	Verified	No attacks.
		Hakhil,Sensor8	SKR k	Ok	Verified	No attacks.
		Hakhil,Sensor9	Niagree	Ok	Verified	No attacks.
		Hakhil,Sensor10	Nisynch	Ok	Verified	No attacks.
Gateway	Gateway	Hakhil,Gateway4	Secret Rnd1	Ok	Verified	No attacks.
		Hakhil,Gateway5	Secret Rnd0	Ok	Verified	No attacks.
		Hakhil,Gateway6	Secret Sensordata	Ok	Verified	No attacks.
		Hakhil,Gateway7	Secret Nk1	Ok	Verified	No attacks.
		Hakhil,Gateway8	Secret Np1	Ok	Verified	No attacks.
		Hakhil,Gateway9	Alive	Ok	Verified	No attacks.
		Hakhil,Gateway10	Weakagree	Ok	Verified	No attacks.
		Hakhil,Gateway11	SKR k	Ok	Verified	No attacks.
		Hakhil,Gateway12	Niagree	Ok	Verified	No attacks.
		Hakhil,Gateway13	Nisynch	Ok	Verified	No attacks.

Done.

Figure 5.21 Authentication protocol result by Scyther tool

The secret claim shows that Rnd1, Rnd0, sensordata, Nk1, and Np1 values are kept confidential during transmission, as seen in Figure 5.21. It is seen that confidentiality is ensured by ensuring the secrecy of the data transferred between the sensor and the gateway.

Alive status has been verified. Our protocol ensures aliveness. Thus, it is seen that Availability is provided. The Weakagree claim ensures that the protocol is safe against impersonation attacks (Nikooghadam & Amintoosi, 2020). The SKR claim defines that session key confidentiality is achieved.

The niagree claim shown by the Sensor and Gateway shows that the non-injective agreement on the message is provided. It is understood that the data is transmitted without corruption and data integrity is ensured. The nisynd claim shows that non-injective synchronization on the message is provided.

As seen in Figure 5.21, the Scyther tool did not find any attack against FSMAP.

5.4 Performance and Security Comparison with Previous Works

The FSMAP has been compared with different studies as seen in Table 5.1 as a result of the literature research. Information on the publications compared in the table were obtained from the contents announced by the authors of the relevant publications.

The protocols proposed in the studies by Nikooghadam & Amintoosi (2020) and Sangwan & Singh (2017) were proven with the Scyther formal verification tool. The proposed protocol in the study by Bera, Das, Balzano & Medaglia (2020) was proven with the AVISPA tool. FSMAP has been proven by testing separately with the Scyther

and AVISPA tools. Among the publications compared, only our study was analyzed using more than one formal verification tool.

The recommended protocols in all studies compared in the table are mutual authentication. While the authentication protocol we recommend is completed in 5 steps, the protocol in the study by Bera et al. (2020) is 18 steps, the protocol in the study by Nikooghadam & Amintoosi (2020) is in 4 steps, and the protocol in the study by Sangwan & Singh (2017) is completed in 6 steps. With the increase in the number of steps of the mutual authentication protocol, the calculations that need to be made increase, and the working time of the protocol increases. Considering the efficiency of IoT devices, the authentication time is also important because it needs to work with performance while ensuring security. It is important that the authentication steps are designed to work with performance, as in the protocol we recommend.

The protocol we propose and the one in the study by Bera et al. (2020) use symmetric encryption. In the Symmetric encryption method, encryption and decryption operations are carried out using a single key. The symmetric encryption method requires less computational power than the asymmetric encryption method and therefore works faster (Ahmad, Alam, Rahman & Tamura, 2015).

The protocol proposed by Bera et al. (2020) shown in the table is secure against man-in-the-middle attack, replay attack, and impersonation attack. The protocol proposed by Nikooghadam & Amintoosi (2020), is secure against replay attack and impersonation attack. The protocol proposed by Sangwan & Singh (2017) is secure against replay attack, man-in-the-middle attack, and DoS attack. The FSMAP is secure against replay attack, full-disclosure attack, ID-theft attack, man-in-the-middle-attack, de-synchronization attack, identity tracing attack, impersonation attack, DoS attack, clone attack, and malicious code injection attack. Considering that IoT devices contain personal data and that these devices provide data transfer frequently in recent years, the importance of protection from cyber-attacks increases once again. Therefore, the

more an IoT device defends against cyber-attacks, the more secure that device is. We can say that our work is secure against ten different cyber-attacks in the communication of IoT devices.

As seen in Table 5.1, FSMAP is superior to other studies compared. The protocol we recommend is offered with strong security features that other protocols do not have.



Table 5.1 Comparison of security features

	FSMAP	(Bera et al., 2020)	(Nikooghadam & Amintoosi, 2020)	(Sangwan & Singh, 2017)
Used formal verification Tool	Scyther & AVISPA	AVISPA	Scyther	Scyther
Mutual authentication	✓	✓	✓	✓
Number of scheme steps	5	18	4	6
Encryption method	Symmetric	Symmetric	Unspecified	Unspecified
Replay attack	✓	✓	✓	✓
Full-disclosure attack	✓	X	X	X
ID-theft attack	✓	X	X	X
Man-in-the-middle attack	✓	✓	X	✓
De-synchronization attack	✓	X	X	X
Identity tracing attack	✓	X	X	X
Impersonation attack	✓	✓	✓	X
DoS attack	✓	X	X	✓
Clone attack	✓	X	X	X
Malicious code injection attack	✓	X	X	X

CHAPTER 6

CONCLUSION

6.1 Conclusion

IoT devices, which are increasingly used in smart cities, also contain personal data. It is important to ensure the security of the information contained by IoT devices that are in constant communication with each other. The more resilient IoT devices with limited resources are to cyber-attacks, the more efficient they can work. Therefore, performance, efficiency, and security are indispensable for IoT devices to provide service.

In this thesis, the FSMAP is proposed to ensure data security during the communication of IoT devices with each other. The proposed authentication protocol is a security protocol that can easily work on IoT devices with limited resources. The security analysis of the proposed protocol is performed with Scyther and AVISPA tools, and it is shown that it is reliable against cyber attacks.

In this thesis, to apply the proposed design in the test environment, a smart textile device is obtained by connecting sensors and IoT-based microcontroller on the woven fabric containing conductive yarns. It is envisaged that the created wearable smart textile device will be used in the field of health in smart cities to determine the arm mobility of the elderly and sick people by measuring the human arm inner angle.

To ensure data security, when the data obtained from the sensors reaches the IoT device, it is encrypted and stored. Thus, the data that can be obtained in case of unauthorized access to the device is completely encrypted. Afterward, the encrypted stored data is wirelessly transferred to the IoT gateway using the FSMAP. When the communication used between the IoT gateway and the smart city cloud service is

considered to be secure, end-to-end security is provided by the use of the proposed protocol.

An application was developed on the Visual Studio platform to display the data sent by the created smart textile device to the IoT gateway. The Windows application developed with the C# programming language runs on the computer to which the IoT gateway is connected and displays the information sent from the IoT sensor to the IoT gateway on the screen. In addition, with a different Windows application developed, the FSMAP is simulated and displayed on the application screen.

The proposed mutual authentication protocol can be used in smart cities to provide more reliable service to IoT devices used in many areas such as health, education, transportation, and so on. It is known that IoT devices are designed with lower features such as memory, battery, and processor than computers. In this respect, it is important to keep IoT device performances at the highest level possible. Reducing cyber-attacks that may occur together with ensuring the security of IoT devices will also increase the performance of IoT devices.

The FSMAP was compared with different studies as a result of the literature review. The proposed protocol is safe against replay, full-disclosure, ID-theft, man-in-the-middle, de-synchronization, identity tracing, impersonation, DoS, clone, and malicious code injection attacks. The study by Bera et al. (2020) is secure against replay, man-in-the-middle, and impersonation attacks. The study by Nikooghadam & Amintoosi (2020) is secure against replay and impersonation attacks. The study by Sangwan & Singh (2017) is secure against man-in-the-middle and DoS attacks. The proposed protocol is more secure against cyber attacks than any other protocol compared.

6.2 Future Works

The protocol proposed in the thesis is applicable to many IoT devices that can be used in smart cities. In cases where the AES encryption algorithm used in the protocol becomes outdated in the future, it will be renewed with an updated encryption algorithm and its usability will be maintained.

In addition, with the new feature brought by Bluetooth 5.1 technology, the distances of Bluetooth devices to each other can be determined. In authentication processes of IoT devices with each other, it can be considered to create an additional security protocol by creating a pattern related to distance information.

REFERENCES

- Adeli, M., & Bagheri, N. (2019). Cryptanalysis of two recently proposed PUF based authentication protocols for IoT: PHEMAP and Salted PHEMAP. *Cryptology ePrint Archive*.
- Aghili, S. F., Mala, H., Kaliyar, P., & Conti, M. (2019). SecLAP: Secure and lightweight RFID authentication protocol for Medical IoT. *Future Generation Computer Systems*, 101, 621-634. <https://doi.org/10.1016/j.future.2019.07.004>
- Ahmad, S., Alam, K. M. R., Rahman, H., & Tamura, S. (2015). A comparison between symmetric and asymmetric key encryption algorithm based decryption mixnets. *In 2015 International Conference on Networking Systems and Security (NSysS), IEEE*, 1-5. <https://doi.org/10.1109/nsyss.2015.7043532>
- Ali, I., Sabir, S., & Ullah, Z. (2019). Internet of things security, device authentication and access control: a review. *International Journal of Computer Science and Information Security (IJCSIS)*, 14(8). <https://doi.org/10.48550/arXiv.1901.07309>
- Alladi, T., Chamola, V., Sikdar, B., & Choo, K. K. R. (2020). Consumer IoT: Security vulnerability case studies and solutions. *IEEE Consumer Electronics Magazine*, 9(2), 17-25. <https://doi.org/10.1109/mce.2019.2953740>
- AL-Turjman, F., Zahmatkesh, H., & Shahroze, R. (2019). An overview of security and privacy in smart cities' IoT communications. *Transactions on Emerging Telecommunications Technologies*. <https://doi.org/10.1002/ett.3677>
- Alzahrani, B. A., & Mahmood, K. (2021). Provable Privacy Preserving Authentication Solution for Internet of Things Environment. *IEEE Access*, 9, 82857-82865. <https://doi.org/10.1109/ACCESS.2021.3086735>

- Aman, M. N., Chua, K. C., & Sikdar, B. (2017). Mutual authentication in IoT systems using physical unclonable functions. *IEEE Internet of Things Journal*, 4(5), 1327-1340. <https://doi.org/10.1109/JIOT.2017.2703088>
- Amanlou, S., Hasan, M. K., & Bakar, K. A. A. (2021). Lightweight and secure authentication scheme for IoT network based on publish–subscribe fog computing model. *Computer Networks*, 199. <https://doi.org/10.1016/j.comnet.2021.108465>
- Amin, R., & Biswas, G.P. (2016). A secure light weight scheme for user authentication and key agreement in multi-gateway based wireless sensor networks. *Ad Hoc Networks*, 36, 58–80. <https://doi.org/10.1016/j.adhoc.2015.05.020>
- Arasteh, H., Hosseinneshad, V., Loia, V., Tommasetti, A., Troisi, O., Shafie-khah, M., & Siano, P. (2016). Iot-based smart cities: A survey. In *2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)*, IEEE, 1-6. <https://doi.org/10.1109/EEEIC.2016.7555867>
- Arduino Online Shop. (2021, April 24). *Arduino Uno Rev3*. <https://store.arduino.cc/usa/arduino-uno-rev3>
- Bae, W. & Kwak, J. (2017). Smart card-based secure authentication protocol in multi-server IoT environment. *Multimedia Tools and Applications*, 79, 15793-15811. <https://doi.org/10.1007/s11042-017-5548-2>
- Baig, Z. A., Sanguanpong, S., Firdous, S. N., Nguyen, T. G., & So-In, C. (2020). Averaged dependence estimators for DoS attack detection in IoT networks. *Future Generation Computer Systems*, 102, 198-209. <https://doi.org/10.1016/j.future.2019.08.007>
- Barbareschi, M., De Benedictis, A., La Montagna, E., Mazzeo, A., & Mazzocca, N. (2019). A PUF-based mutual authentication scheme for Cloud-Edges IoT systems.

- Future Generation Computer Systems*, 101, 246–261.
<https://doi.org/10.1016/j.future.2019.06.012>
- Bera, B., Das, A. K., Balzano, W., & Medaglia, C. M. (2020). On the design of biometric-based user authentication protocol in smart city environment. *Pattern Recognition Letters*, 138, 439-446. <https://doi.org/10.1016/j.patrec.2020.08.017>
- Brezulianu, A., Geman, O., Zbancioc, M. D., Hagan, M., Aghion, C., Hemanth, D. J., et al. (2019). IoT based heart activity monitoring using inductive sensors. *Sensors*, 19(15). <https://doi.org/10.3390/s19153284>
- Ceyhan, E. B., & Sağıroğlu, Ş. (2013). Kablosuz algılayıcı ağlarda güvenlik sorunları ve alınabilecek önlemler. *Politeknik Dergisi*, 16(4),155-163.
- Chakrabarty, S., & Engels, D. W. (2016). A secure IoT architecture for smart cities. In *2016 13th IEEE annual consumer communications & networking conference (CCNC)*, IEEE, 812-813. <https://doi.org/10.1109/ccnc.2016.7444889>
- Chang, C. C., Wu, H. L., & Sun, C. Y. (2017). Notes on “Secure authentication scheme for IoT and cloud servers”. *Pervasive and Mobile Computing*, 38, 275-278. <http://dx.doi.org/10.1016/j.pmcj.2015.12.003>
- Cremers, C. (2021, October 29). *The Scyther tool*. CISPA. <https://people.cispa.io/cas.cremers/scyther>
- Çavdar, T., & Öztürk, E. (2018). Nesnelerin interneti için yeni bir mimari tasarımı. *Sakarya Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, 22(1), 39-48. <https://doi.org/10.16984/saufenbilder.285444>
- Dalkılıç, F. (2015). *Real-time intelligent journey planner* [doctoral dissertation]. Dokuz Eylül University, İzmir.

- Dalkılıç, G. (2014). An enhanced lightweight authentication (ELA) protocol for WISP based RFID applications. *Life Science Journal*, 11(10), 878-885. <https://doi.org/10.7537/marslsj111014.138>
- Dalkılıç, H. (2016). *Applications of sensors in secure embedded systems* [Master's dissertation]. Dokuz Eylül University, İzmir.
- Dalkılıç, H., Özcanhan, M. H., & Özdemir, H. (2021). Wireless data transfer with the use of internet of things (IoT) technologies in smart textiles. *The Journal of The Textile Institute*, Published Online: 17 June 2021. <https://doi.org/10.1080/00405000.2021.1940035>
- Deebak, B. D., & AL-Turjman, F. (2021). Lightweight authentication for IoT/Cloud-based forensics in intelligent data computing. *Future Generation Computer Systems*, 116, 406-425. <https://doi.org/10.1016/j.future.2020.11.010>
- Deng, M., & Zhu, W. (2013). Desynchronization attacks on RFID security protocols. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, 11(2), 681-688.
- Dhanda, S. S., Singh, B., & Jindal, P. (2020). Lightweight cryptography: a solution to secure IoT. *Wireless Personal Communications*, 112(3), 1947-1980. <https://doi.org/10.1007/s11277-020-07134-3>
- Ding, X., Wang, X., Xie, Y., & Li, F. (2021). A Lightweight Anonymous Authentication Protocol for Resource-Constrained Devices in Internet of Things. *IEEE Internet of Things Journal*, 9 (3), 1818-1829. <https://doi.org/10.1109/JIOT.2021.3088641>
- Efe, A., Aksöz, E., Hanecioğlu, N., & Yalman, Ş. N. (2018). Smart Security of IoT Against to DDoS Attacks. *Uluslararası Yenilikçi Mühendislik Uygulamaları Dergisi*, 2(2), 35-43.

- El Madhoun, N., Bertin, E., Badra, M., & Pujolle, G. (2021). Towards more secure EMV purchase transactions. *Annals of Telecommunications*, 76(3), 203-222. <https://doi.org/10.1007/s12243-020-00784-1>
- ETechnoG. (2022, February 10). *Easily Understand IOT Block Diagram and Architecture*. <https://www.etechnog.com/2021/05/iot-block-diagram-architecture.html>
- Fadele, A. A., Othman, M., Hashem, I. A. T., Yaqoob, I., Imran, M., & Shoaib, M. (2019). A novel countermeasure technique for reactive jamming attack in internet of things. *Multimedia Tools and Applications*, 78(21), 29899-29920. <https://doi.org/10.1007/s11042-018-6684-z>
- Ferdows, J., Mehedi, S. T., Hossain, A. D., Shamim, A. A. M., & Rasiq, G. R. I. (2020). A Comprehensive Study of IoT Application Layer Security Management. In *2020 IEEE International Conference for Innovation in Technology (INOCON)*, IEEE, 1-7. <https://doi.org/10.1109/inocon50539.2020.9298245>
- Fu, J., Wu, C., Chen, X., Fan, R., & Ping, L. (2010). Scalable pseudo random RFID private mutual authentication. In *2010 2nd International Conference on Computer Engineering and Technology*, 7, IEEE, V7-497-V7-500. <https://doi.org/10.1109/ICCET.2010.5485559>
- Genet, T. (2015). *A Short SPAN+AVISPA Tutorial* [Research Report]. IRISA.
- Grammatikis, P. I. R., Sarigiannidis, P. G., & Moscholios, I. D. (2019). Securing the Internet of Things: Challenges, threats and solutions. *Internet of Things*, 5, 41-70. <https://doi.org/10.1016/j.iot.2018.11.003>
- Hammoudi, S., Aliouat, Z., & Harous, S. (2018). Challenges and research directions for Internet of Things. *Telecommunication Systems*, 67(2), 367-385. <https://doi.org/10.1007/s11235-017-0343-y>

- Hernández, J. L., García, R., Schonowski, J., Atlan, D., Chanson, G., & Ruohomäki, T. (2020). Interoperable open specifications framework for the implementation of standardized urban platforms. *Sensors*, 20(8), 2402. <https://doi.org/10.3390/s20082402>
- Hutchins, B., Reddy, A., Jin, W., Zhou, M., Li, M., & Yang, L. (2018). Beat-pin: A user authentication mechanism for wearable devices through secret beats. *In Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 101-115. <https://doi.org/10.1145/3196494.3196543>
- Islam, S. M. R., Kwak, D., Kabir, M. H., Hossain, M., & Kwak, K.-S. (2015). The internet of things for health care: a comprehensive survey. *IEEE access*, 3, 678-708. <https://doi.org/10.1109/access.2015.2437951>
- Ismat, T. (2015). *Comparative analysis of AES algorithms and implementation of AES in Arduino* [Doctoral dissertation]. BRAC University.
- Joshitta, R. S. M., & Arockiam, L. (2017). Device authentication mechanism for IoT enabled healthcare system. *In 2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)*, IEEE, 1-6. <https://doi.org/10.1109/ICAMMAET.2017.8186646>
- Kalra, S., & Sood, S. K. (2015). Secure authentication scheme for IoT and cloud servers. *Pervasive and Mobile Computing*, 24, 210-223. <http://dx.doi.org/10.1016/j.pmcj.2015.08.001>
- Khemissa, H., & Tandjaoui, D. (2016). A novel lightweight authentication scheme for heterogeneous wireless sensor networks in the context of Internet of Things. *In 2016 Wireless Telecommunications Symposium (WTS)*, IEEE, 1-6. <https://doi.org/10.1109/WTS.2016.7482039>

Köseoğlu, Ö., & Demirci, Y. (2018). Akıllı şehirler ve yerel sorunların çözümünde yenilikçi teknolojilerin kullanımı. *Uluslararası Politik Araştırmalar Dergisi*, 4(2), 40-57. <https://doi.org/10.25272/j.2149-8539.2018.4.2.03>

Landman, D. (2021 April 1). *DavyLandman/AESLib: Arduino Library for AES Encryption (source based on avr-crypto-lib)*. GitHub. <https://github.com/DavyLandman/AESLib>

Lara, E., Aguilar, L., Sanchez, M. A., & García, J. A. (2020). Lightweight authentication protocol for M2M communications of resource-constrained devices in industrial Internet of Things. *Sensors*, 20(2), 501. <https://doi.org/10.3390/s20020501>

Lee, J., Yu, S., Park, K., Park, Y., & Park, Y. (2019). Secure three-factor authentication protocol for multi-gateway IoT environments. *Sensors*, 19(10), 2358. <https://doi.org/10.3390/s19102358>

Li, X., Yan, F., Zuo, F., Zeng, Q., & Luo, L. (2019). Touch well before use: Intuitive and secure authentication for iot devices. *In The 25th annual international conference on mobile computing and networking*, 1-17. <https://doi.org/10.1145/3300061.3345434>

Lin, H., & Bergmann, N. (2016). IoT privacy and security challenges for smart home environments. *Information*, 7(3), 44. <https://doi.org/10.3390/info7030044>

Lin, X., & Seet, B. C. (2016). Battery-free smart sock for abnormal relative plantar pressure monitoring. *IEEE transactions on biomedical circuits and systems*, 11(2), 464-473. <https://doi.org/10.1109/TBCAS.2016.2615603>

Malche, T., Maheshwary, P., & Kumar, R. (2019). Environmental monitoring system for smart city based on secure Internet of Things (IoT) architecture. *Wireless*

Personal Communications, 107(4), 2143-2172. <https://doi.org/10.1007/s11277-019-06376-0>

Mohanta, B. K., Jena, D., & Patnaik, S. (2021). An ECC-based authentication protocol for fog-IoT enabled smart home environment. *International Journal of Intelligent Systems Technologies and Applications*, 20(1), 25-46. <https://doi.org/10.1504/ijista.2021.114638>

Naik, N. (2017). Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In *2017 IEEE international systems engineering symposium (ISSE)*, IEEE, 1-7. <https://doi.org/10.1109/SysEng.2017.8088251>

Nandy, T., Idris, M. Y. I. B., Noor, R. M., Kiah, L. M., Lun, L. S., Juma'at, N. B. A., et al. (2019). Review on security of Internet of Things authentication mechanism. *IEEE Access*, 7, 151054-151089. <https://doi.org/10.1109/ACCESS.2019.2947723>

Navas, R. E., & Toutain, L. (2018). Late: A lightweight authenticated time synchronization protocol for IoT. In *2018 Global Internet of Things Summit (GloTS)*, 1-6. IEEE. <https://doi.org/10.1109/GIOTS.2018.8534565>

Nikooghadam, M., & Amintoosi, H. (2020). An Improved Secure Authentication and Key Agreement Scheme for Healthcare Applications. In *2020 25th International Computer Conference, Computer Society of Iran (CSICC)*, IEEE, 1-7. <https://doi.org/10.1109/CSICC49403.2020.9050069>

Nivetha, R., & Shyla, S. I. (2021). Analyzing Data Security Issues and Solutions in Cloud Computing. *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*, 2(1).

Özcanhan, M. H., Dalkılıç, G., & Utku, S. (2014). Cryptographically supported NFC tags in medication for better inpatient safety. *Journal of medical systems*, 38(8), 61. <https://doi.org/10.1007/s10916-014-0061-x>

Özcanhan, M. H., & Utku, S. (2018). Attacking A PUF Based Mutual Authentication Protocol Designed For Internet of Things. *International Journal of Computer Science and Software Engineering (IJCSSE)*, 7(11), 270-274.

Park, J. H., Salim, M. M., Jo, J. H., Sicato, J. C. S., Rathore, S., & Park, J. H. (2019). CIoT-Net: a scalable cognitive IoT based smart city network architecture. *Human-centric Computing and Information Sciences*, 9(1), 1-20.
<https://doi.org/10.1186/s13673-019-0190-9>

Rajab, H., & Cinkelr, T. (2018). IoT based smart cities. In *2018 international symposium on networks, computers and communications (ISNCC)*, IEEE, 1-4.
<https://doi.org/10.1109/isncc.2018.8530997>

Raymond, D. R., & Midkiff, S. F. (2008). Denial-of-service in wireless sensor networks: Attacks and defenses. *IEEE Pervasive Computing*, 7(1), 74-81.
<https://doi.org/10.1109/mprv.2008.6>

Reddy, A. G., Suresh, D., Phaneendra, K., Shin, J. S., & Odelu, V. (2018). Provably secure pseudo-identity based device authentication for smart cities environment. *Sustainable cities and society*, 41, 878-885.
<https://doi.org/10.1016/j.scs.2018.06.004>

Sadhukhan, D., Ray, S., Biswas, G. P., Khan, M. K., & Dasgupta, M. (2021). A lightweight remote user authentication scheme for IoT communication using elliptic curve cryptography. *The Journal of Supercomputing*, 77(2), 1114-1151.
<https://doi.org/10.1007/s11227-020-03318-7>

Sangwan, A., & Singh, V. R. (2017). A secure authentication scheme for WiMax network and verification using scyther tool. *International Journal of Applied Engineering Research*, 12(11), 3002-3008.

- Sey, D. (2018). A survey on authentication methods for the Internet of Things. *PeerJ Preprints*, 6. <https://doi.org/10.7287/peerj.preprints.26474v2>
- Shah, D. K., Bharadi, V. A., Kaul, V. J., & Amrutia, S. (2015). End-to-end encryption based biometric SaaS: Using raspberry pi as a remote authentication node. *In 2015 International Conference on Computing Communication Control and Automation*, IEEE, 52-59. <https://doi.org/10.1109/ICCUBE.2015.19>
- Shah, D., Bharadi, V. (2016). IoT based biometrics implementation on Raspberry Pi. *Procedia Computer Science*, 79, 328-336. <https://doi.org/10.1016/j.procs.2016.03.043>
- Shah, T., & Venkatesan, S. (2018). Authentication of IoT device and IoT server using secure vaults. *In 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)*, IEEE, 819-824. <https://doi.org/10.1109/TrustCom/BigDataSE.2018.00117>
- Shariq, M., Singh, K., Bajuri, M. Y., Pantelous, A. A., Ahmadian, A., & Salimi, M. (2021). A Secure and reliable RFID authentication protocol using digital schnorr cryptosystem for IoT-enabled healthcare in COVID-19 scenario. *Sustainable Cities and Society*, 75, 103354. <https://doi.org/10.1016/j.scs.2021.103354>
- Sisodia, D. (2020). On the State of Internet of Things Security: Vulnerabilities, Attacks, and Recent Countermeasures. *University of Oregon*, Tech. Rep.
- Surendran, S., Nassef, A., & Beheshti, B. D. (2018). A survey of cryptographic algorithms for IoT devices. *In 2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, IEEE, 1-8. <https://doi.org/10.1109/LISAT.2018.8378034>

- Taş, O., & Kiani, F. (2020). Nesnelerin interneti (IoT) ve kablosuz algılayıcı ağların güvenliğine yapılan saldırıların tespit edilmesi ve önlenmesi. *Politeknik Dergisi*, 24(1),219-235. <https://doi.org/10.2339/politeknik.627825>
- Tawalbeh, L. A., Muheidat, F., Tawalbeh, M., & Quwaider, M. (2020). IoT Privacy and security: Challenges and solutions. *Applied Sciences*, 10(12). <https://doi.org/10.3390/app10124102>
- Tewari, A., & Gupta, B. B. (2017). Cryptanalysis of a novel ultra-lightweight mutual authentication protocol for IoT devices using RFID tags. *The Journal of Supercomputing*, 73(3), 1085-1102. <https://doi.org/10.1007/s11227-016-1849-x>
- Thakur, V., Indra, G., Gupta, N., Chatterjee, P., Said, O., & Tolba, A. (2021). Cryptographically secure privacy-preserving authenticated key agreement protocol for an IoT network: A step towards critical infrastructure protection. *Peer-to-Peer Networking and Applications*, 15(1), 206-220. <https://doi.org/10.1007/s12083-021-01236-w>
- Turkanović, M., Brumen, B., & Hölbl, M. (2014). A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the internet of things notion. *Ad Hoc Networks* 20, 96–112. <http://dx.doi.org/10.1016/j.adhoc.2014.03.009>
- Wang, K. H., Chen, C. M., Fang, W., & Wu, T. Y. (2018). On the security of a new ultra-lightweight authentication protocol in IoT environment for RFID tags. *The Journal of Supercomputing*, 74(1), 65-70. <https://doi.org/10.1007/s11227-017-2105-8>
- Yahuza, M., Idris, M. Y. I., Ahmedy, I. B., Wahab, A. W. A., Nandy, T., Noor, N. M., et al. (2021). Internet of drones security and privacy issues: Taxonomy and open challenges. *IEEE Access*, 9, 57243-57270. <https://doi.org/10.1109/access.2021.3072030>

- Yang, Z., Yue, Y., Yang, Y., Peng, Y., Wang, X., & Liu, W. (2011). Study and application on the architecture and key technologies for IOT. *In 2011 International Conference on Multimedia Technology*, IEEE, 747-751. <https://doi.org/10.1109/ICMT.2011.6002149>
- Yıldırım, K., Dalkılıç, G., & Duru, N. (2019). Security analysis of Hsiang m-coupon protocol. *Journal of the Faculty of Engineering and Architecture of Gazi University*, 34(4), 1705-1724. <https://doi.org/10.17341/gazimmfd.571490>
- Zargar, S., Shahidinejad, A., & Ghobaei-Arani, M. (2021). A lightweight authentication protocol for IoT-based cloud environment. *International Journal of Communication Systems*, 34(11). <https://doi.org/10.1002/dac.4849>
- Zhao, C., Shi, M., Cai, Z., & Chen, C. (2018). Research on the open-categorical classification of the Internet-of-things based on generative adversarial networks. *Applied Sciences*, 8(12), 2351. <https://doi.org/10.3390/app8122351>
- Ziegeldorf, J. H., Morchon, O. G., & Wehrle, K. (2014). Privacy in the internet of things: threats and challenges. *Security and Communication Networks*, 7(12), 2728–2742. <https://doi.org/10.1002/sec.795>