DOKUZ EYLÜL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

INCREASING SECURITY IN COMMUNICATION BETWEEN IoT DEVICES

by

Seyhun Yaşar ÖZKAL

July, 2021

İZMİR

INCREASING SECURITY IN COMMUNICATION BETWEEN IoT 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 Master of Science in Computer Engineering, Computer Engineering Program

by

Seyhun Yaşar ÖZKAL

July, 2021

İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"INCREASING SECURITY IN COMMUNICATION BETWEEN IoT DEVICES"** completed by **SEYHUN YAŞAR ÖZKAL** under supervision of **ASSOC. PROF. DR. MEHMET HİLAL ÖZCANHAN** and we certify that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

.....

Assoc. Prof. Dr. Mehmet Hilal ÖZCANHAN

Supervisor

.....	.....
Assoc. Prof. Dr. Gökhan DALKILIÇ	Assoc. Prof. Dr. Ahmet Tuncay ERCAN
Jury Member	Jury Member

Prof. Dr. Özgür ÖZÇELİK

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

First and foremost, I would like to thank my thesis supervisor Assoc. Prof. Dr. Mehmet Hilal ÖZCANHAN of the Computer Engineering Department/Faculty of Engineering at Dokuz Eylül University for his contributions to this study. With his disciplined and perfectionist attitude, he has always guided me to the best.

I would also like to acknowledge Hakan DALKILIÇ, who is a PhD candidate at the Department of Computer Engineering at Dokuz Eylül University. Whenever I encountered a problem during my research, he was ready to scaffold me.

I must express my very profound gratitude to my parents and to my elder brother Ceyhun ÖZKAL, who has never lost his faith in me and always believed in my success.

Seyhun Yaşar ÖZKAL

INCREASING SECURITY IN COMMUNICATION BETWEEN IoT DEVICES

ABSTRACT

Today's rapidly evolving technology is contributing to the digitalization of all human life information. Producing the sensitive information by using small-sized devices and transmitting it with least cost has become a requirement. In order to meet the requirement, information must be transmitted by a secure communication system using wireless technologies, ensuring confidentiality, integrity, and availability. At this conjunction, Internet of Things (IoT) ecosystem provides a network allowing devices with limited resources to communicate with each other. Data processing has evolved into automated tasks due to the increasing number of IoT devices, in daily lives and business. However, lack of a precise standardization in communication and information security at design phase of the IoT systems cause weaknesses of secrecy. In our study, we aim to provide information security by eliminating the secrecy weaknesses in the communication between IoT sensors and gateways. In this context, a communication protocol has been proposed in our study to ensure information security against known attacks on IoT devices. The proposed protocol provides mutual authentication through five communication steps. The protocol and its security analyses are presented in detail. The designed protocol was tested against known attacks on IoT devices, with the formal verification tool Scyther. Test steps and results are presented in detail. Analysis has shown that the proposed protocol is resistant to known attacks and provides a high degree of information security.

Keywords: Network attacks, information security, embedded systems, security model, Internet of Things devices, Scyther tool

NESNELERİN İNTERNETİ CİHAZLARI ARASINDAKİ İLETİŞİM GÜVENLİĞİNİN ARTIRILMASI

ÖZ

Günümüzün gelişen ve değişen teknolojisi, insan hayatındaki bilgilerin dijitalleşmesine sebep olmaktadır. Boyutları küçük olan cihazlarla mahrem sayılabilecek bilgileri üretmek ve en az maliyetle iletmek ihtiyaç haline gelmektedir. İhtiyacı gidermek için kablosuz teknolojilerle bilgilerin gizliliği, bütünlüğü ve devamlılığı sağlanarak güvenli bir haberleşme sistemi ile iletilmesi gerekmektedir. Bu noktada Nesnelerin İnterneti (Internet of Things, IoT) ekosistemi kısıtlı kaynaklara sahip olan cihazların birbiriyle iletişimini sağlayan bir ağ sunmaktadır. Günlük kullanımda ve iş hayatında sayısı hızla artmakta olan IoT cihazları sayesinde veri işleme otomasyon haline gelmektedir. Ancak bu sistemlerin haberleşmesinde kesin bir standardizasyonun olmaması ve tasarımları aşamasında bilgi güvenliğinin dikkate alınmaması gizlilik zafiyetlerine sebep olmaktadır. Çalışmamızda IoT sensörleri ve ağ geçitleri arasındaki iletişimde oluşan gizlilik zafiyetlerinin giderilerek bilgi güvenliğinin sağlanması amaçlanmaktadır. Bu kapsamda, çalışmamızda IoT cihazlarına yapılan bilinen saldırılara karşı bilgi güvenliğini sağlayacak bir protokol önerilmiştir. Önerilen protokol karşılıklı kimlik doğrulama özelliğine sahip, beş iletişim adımından oluşmaktadır. Önerilen protokol ve güvenlik analizleri detaylı olarak anlatılmıştır. Tasarlanan protokol IoT cihazlarına yapıldığı bilinen saldırılara karşı kabul görmüş bir protokol onaylama aracı olan Scyther ile test edilmiştir. Test aşamaları ve sonuçları detaylı şekilde sunulmuştur. Analizler önerilen protokolün bilinen saldırılara karşı dirençli olduğunu ve yüksek derecede bilgi güvenliği sağladığını göstermiştir.

Anahtar Kelimeler: Ağ saldırıları, bilgi güvenliği, gömülü sistemler, güvenlik modeli, Nesnelerin İnterneti cihazları, Scyther aracı

CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER ONE
INTRODUCTION

Today, with the development of information technologies, electronic devices facilitate social life and provide service in many different areas. The devices serve the users with specific commands. Embedded systems are devices that make up the electronic hardware and software sub-units. The systems have limited resources and are designed to perform specific functions.

Internet of Things (IoT) is a network of physical objects that exchange data using communication protocols. They are a subset of embedded systems. The IoT is created by adding software and hardware to electronic devices. The devices share their data in a network using a communication module. They have features such as generating, sending, receiving, and processing data. The devices are categorized into two by their resources as encrypting and non-encrypting.

Securing the communication of devices that cannot perform encryption is more difficult than devices that can perform encryption. Because sensors have more limited resources than gateways, resource-consuming operations in authentication protocols are performed by the gateway. As the resources of the devices decrease, so does their ability to encrypt. The situation makes it difficult to ensure confidentiality in authentication protocols. When the protocols are attacked, they must stop the attack or disrupt communication between the devices.

Sensors and gateways generate data and initiate the pre-preparation of communication according to the established authentication protocols. The devices connect to a network to send and receive their data. In order to ensure data security, communication steps are encrypted or different security components are used.

IoT systems have some communication features. The features are listed below:

- The system must be expandable or downsized in line with the increase or decrease in the number of users.

- The devices that make up the IoT system must have an integrated structure.
- The devices must be portable and replaceable.
- Communication steps must be carried out sequentially.
- Real-time queries of IoT devices and their data should be stored.
- Software that makes up the IoT system must be updateable and modifiable.

Machine to Machine (M2M) communication is the ability of one machine to communicate with another to share and exchange data. With the ever-increasing use of technology, Radio Frequency Identification (RFID) sensors, monitoring devices, sensors, communication circuits, biochips, vehicles, cars, buildings, smart cities interact with countless devices (Aydın & Dalkılıç, 2016). Examples for areas of use can be listed as home automation, life support systems, robotic applications, health services, industrial production, automation, and logistics, as shown in Figure 1.1.

Remote control of devices and integration of generated data with computer-based systems are the benefits of the IoT system. The objects constitute network systems that communicate and share information through various communication protocols. IoT ecosystems can run by connecting via the internet as well as without being connected to the internet. In addition, devices that communicate via the internet can connect to a server in the cloud. Communication without human intervention turns jobs into automation. Uninterrupted operation in the IoT system increases efficiency. Thus, it facilitates daily life, business life, and jobs in the global economy.

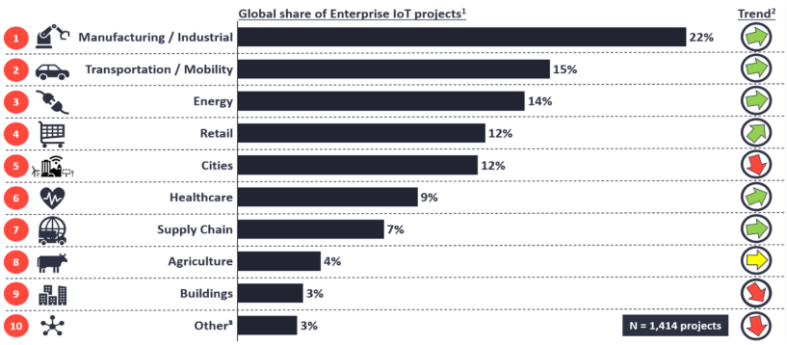

Figure 1.1 Top 10 IoT applications in 2020 (Scully, 2020)

As illustrated in Figure 1.2 Data-Information-Knowledge-Wisdom (DIKW) pyramid is a progressive concept of data. The DIKW pyramid goes from data at the base, upward through information, knowledge, and ending with wisdom concepts. Data is a collection of facts such as converted numbers, words, metrics, observations that computers can process. Data do not contain any comments. Information is the meaning of data. It is the stage of analyzing the collected data. Knowledge is a concept that is analyzed, processed, and integrated with different sources as a result of corporate or personal experiences. Wisdom is the latest advanced phase of data on which ethical values, rules, reasons are based. Wisdom also includes critical or practical evaluation in any case (Henkoğlu, 2019). Based on the statements, it is seen that the data constitutes the privacy of individuals or institutions and the need to ensure the security of the data. IoT devices also tend to cause security problems due to their limited resources, energy, memory, and processing power.

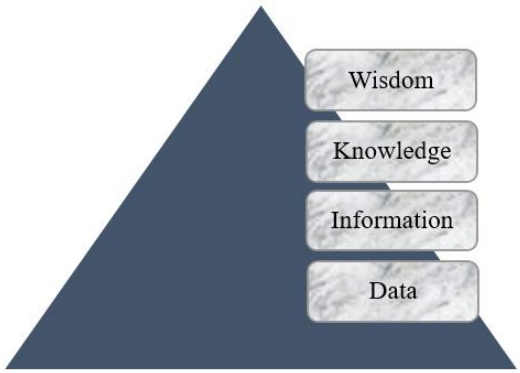

Figure 1.2 DIKW pyramid

Due to the abundance of digital information in our age, its security is becoming increasingly important. Information security is the protection of the confidentiality, integrity, availability of information. Violations of the criteria cause some information security vulnerabilities.

The production of new devices inevitably brings the existence of new communication protocols. As the emerging risks have created security measures as a necessity, information security must also be achieved in the communication of IoT

devices. By achieving physical security and network security in communication, it is possible to access the data according to the authorization of the relevant objects. Given that the theft of data might bring major problems, the information should be transferred correctly without being changed.

The application of IoT devices in many critical fields has been accompanied by attacks in the area which can violate the security model, including data confidentiality, integrity, and availability. The three main principles are based on ensuring information security. Data confidentiality refers to the unauthorized person's inability to access information, while data integrity means that information cannot be changed, deleted, or corrupted by unauthorized persons. Availability ensures that authorized users always have access to the data. Therefore, the development of secure communication methods as well as approaches suitable for IoT devices has come to be a sine qua non (Siponen & Oinas-Kukkonen, 2007).

Providing communication and security between the devices has primarily created the concept of authentication. Only the devices in communication with each other need to be able to understand, verify and prove each other. For example, the IPsec protocol, which is accepted as the gold standard and includes authentication and security, cannot be implemented in IoT devices due to the high consumption of resources. In addition, each step of communication increases network security by simplifying standard protocols, and the measures, though adding a cost to the resources of the IoT devices, contribute to the creation of security (Çakır & Özcanhan, 2011).

The present paper provides information about increasing security in communication between IoT devices. The relevant studies the IoT communication security have been analyzed, and potential security threats have been investigated. At the end of the study, an authentication protocol has been created to increase the security of the communication. The aim of the present study is to provide secure authentication, data exchange by applying correct, testing some known attacks and necessary procedures without consuming IoT resources completely.

The main contributions of the thesis can be listed as increasing security in communication between IoT devices, preventing some known attacks, providing mutual authentication, creating a lightweight protocol, and testing our protocol design by Scyther, a formal security tool.

The thesis is organized into seven main sections. In Chapter 1 of the thesis, IoT definition, features, usage areas, benefits, security vulnerabilities are mentioned. Related works, the background of IoT, and literature view are referred to in Chapter 2. In Chapter 3, security requirements of IoT devices, basic security principles, and cryptology science are explained. Requirements, design, communication steps, and security components of our protocol are explained in Chapter 4. Chapter 5 focuses on testing our protocol design against some known IoT attacks. Chapter 6 refers to materials and methods with formal security analysis tool Scyther. The last one includes the conclusions and future works.

CHAPTER TWO
RELATED WORK

Today, a large and growing body of research is carried out on the secure communication protocols of IoT devices. The rapid advancement of technology indicates that there will be attacks on the protocols, and that precautionary measures should be taken accordingly. According to previous research, various measures should be taken against some known attacks on the IoT devices with limited resources.

2.1 The Background of IoT

The beginning of the use of IoT is in the 1980s. A Coca-Cola machine at Carnegie Melon University is considered the first application. In 1991, a group of academics used a common coffee machine located on the upper floors at the University of Cambridge. Therefore, when they had to go upstairs to drink coffee, they sometimes saw the coffee machine empty. Therefore, they wanted to develop a camera system by which they could see the coffee machine. The system captured three images of the coffee machine per minute and transferred them to the computers of the academics, which enabled them to see the amount of coffee in the machine (Khvoynitskaya, 2019).

With developing and changing technology, the IoT system studies have increased. Authentication methods and security of protocols are being improved. However, smart devices do not have common hardware, algorithm, and interface. Therefore, it is seen as the biggest problem in the development of secure protocols. Due to the lack of specific standards, security vulnerabilities may exist in the protocols.

2.2 Literature Review

Some security precautions can simplify the IPsec standard and apply it to embedded systems. Security weaknesses in embedded systems as well as security standards are analyzed, and information is provided about the attacks caused by security vulnerabilities. It is emphasized that data communication should be performed by

hiding and encrypting. The suggestions are based on security protocols, architecture, and rules used in data security, encryption algorithms providing confidentiality, identification, and authentication, hashing algorithms providing data integrity, secret key pre-shares, and digital signature and certificates (Özcanhan, 2011).

Previous studies have investigated how patient falls can be prevented by using wearable sensors, and, if falling cases have occurred, how these falls can be alerted. The sensor with acceleration and angular acceleration measurement sends the patient data to Arduino Nano, which has an ATMega 328 microcontroller. After the data are transmitted to Arduino and inferences are calculated, they are transferred to the computer via Bluetooth module. Since many malicious listeners may access and change the data, data transmission via wearable sensors must be confidential. Advanced Encryption Standard is used as a precautionary measure (Dalkılıç & Özcanhan, 2016).

Another study tries to increase the efficiency of the patient monitoring system and to improve the correct medication management. They stated that EPC Global Class 1 Generation 2 based protocols were insufficient in drug administration. They analyzed that the EKATE protocol, which has increased security, is also known as three types of attacks. It has been shown that the proposed IMS-NFC protocol is more secure than EKATE and complies with International Organization for Standardization (ISO) communication and security standards. The IMS-NFC protocol involves a little more cost, but they mentioned that it is preferable due to its compliance with the standards (Özcanhan, Dalkılıç & Utku, 2014).

Passive tags, authentication of communication between reader and server, and exchange of data sent in the air environment in RFID technology are discussed in the literature. In IoT devices, secure key exchange, encryption methods, and standard protocols are simplified, and precautions are taken against attacks. The paper addresses the simplification of the Internet Key Exchange version 2 (IKEv2) protocol. The protocol is named “Enhanced Lightweight Authentication Protocol” (Dalkılıç, 2014).

The Quadratic Residue authentication protocol has been analyzed for security. The protocol is supported by the Chinese Remainder Theorem (QR-CR duo). The study stated that there are serious mutual authentication vulnerabilities in the QR-CR duo protocol. The presented analyzes have shown that both tag and reader impersonation attacks against the protocol are possible. The attacks resulted in the communication of data as plaintext and subsequently increased vulnerabilities (Özcanhan, 2015).

The security of EPC Gen-2 tags has been investigated in in-patient medication. The study examined two proposed protocols and demonstrated the new full-disclosure attack. They found that there are security vulnerabilities protocols in pseudo-random number generator and cyclic redundancy check function in which EPC Gen-2 is used. As a solution recommendation, a longer-size key and better security components should be created. The protocols should be implemented using shorter working distance and smarter labels (Özcanhan, Dalkılıç & Utku, 2013).

Another study tries to probe the security problems as well as their solutions in the protocol layers in IoT. Because IoT devices suffer from limited resources, the researchers try to sort out the problems through simple and effective security measures. Information security is increased by using symmetric and asymmetric encryption techniques. Their study concludes that security problems do not belong to any layer, and that some solutions should be produced by considering all the layers (Zhao & Ge, 2013).

Ultra-lightweight, cryptographic, mutual authentication protocols have been implemented to increase security in communication for RFID tags. The protocol is named the protocol as ULERAP. The protocol is shown in Figure 2.1. The paper summarizes the attacks on previous protocols and, shows more attacks. It has explained the types of active and passive attacks. Since the identification number has been not kept secret in the previous protocols, the security vulnerabilities have occurred in the protocols (Özcanhan & Dalkılıç, 2015).

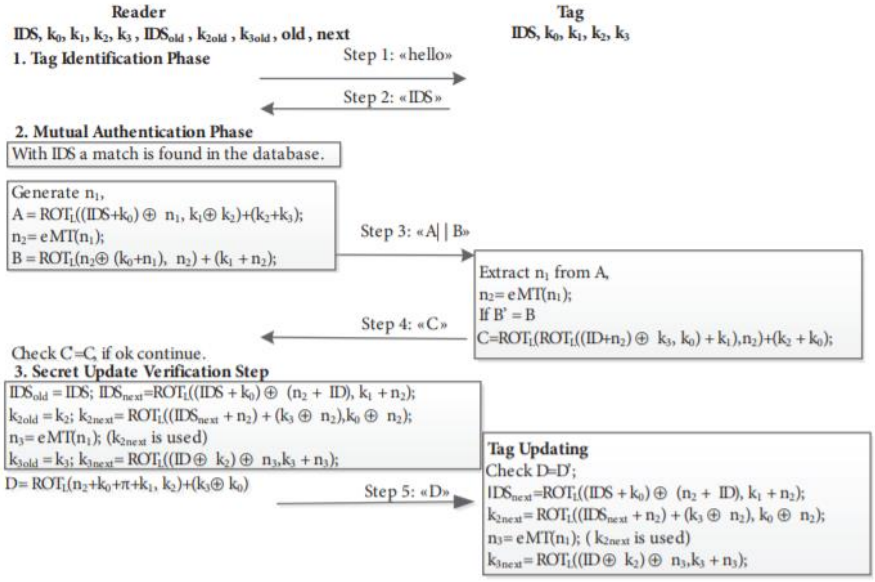

Figure 2.1 ULERAP steps (Özcanhan & Dalkılıç, 2015)

As mentioned in other studies, the security scenarios in IoT technology have been discussed by analyzing possible attacks and threats as well as offering possible solutions to improve security mechanisms in IoT networks. In addition, the differences between traditional network systems and IoT networks are investigated in detail on a layer basis. The researchers emphasize the security problems in both networks, calling for different approaches and techniques (Alabaa, Othmana, Hashema & Alotaibib, 2017).

Some weaknesses have also been identified in RFID authentication protocols regarding secure drug management. Attacks on the Kea protocol and failure in the creation of the protocol have been noted. The author has proved that private data in the protocol can be disclosed. The author has developed the Kea protocol and created the SC-SRP protocol. SC-SRP is lightweight cryptography and provides encryption with a variety of keys instead of a static key. The protocol shows that it yields better results in terms of performance and security (Özcanhan, 2014).

Another study in RFID technology has provided information about confidentiality, security issues, and mutual authentication protocol. Authentication of RFID tags in a wireless network environment using lightweight and ultra-lightweight protocols is

investigated. Electronic Product Code (EPC) C1 Gen2 standards are used in the implementation of the protocols. In the lightweight protocol, Cyclic Redundancy Check (CRC) function is shown to be the main security problem. It is mentioned that the ultra-lightweight protocol consists of bit-wise operations and threats (Ibrahim & Dalkılıç, 2019).

SAKM protocols have been analyzed using RFID tags for telecare medicine. Telecare providers use sensors, alarms to monitor and connect their users. It detects emergencies such as possible fire, flood, abnormal movement. In the study, two known types of attacks have been focused on. As a result of the analysis, desynchronization attack and full-disclosure attacks were determined to be vulnerable. The weakness in the protocol poses threats to the privacy and health of patients (Özcanhan, 2015).

The security analysis of the Near Field Communication (NFC) has been made, based on m-coupon protocol developed by Hsiang. With the m-coupon application used in mobile devices, companies can discount customers. Whether the protocol can manipulate the system by using the Scyther tool has also been analyzed. Suggestions are presented for the security vulnerabilities detected in the protocol. In the recently proposed m-coupon protocol, the vulnerabilities have been eliminated and a secure structure has been created (Yıldırım, Dalkılıç & Duru, 2019).

In Özcanhan et al. study in the field of medication, the EPC Gen-2 type tag does not support cryptographic algorithms due to its limited resources, but the reader's resources are sufficient. It has been stated that the use of Pseudo-Random Number Generator (PRNG) as an encryption algorithm is a security weakness. When the protocols consisting of NFC tags and EPC Gen-2 tags are analyzed, it has been shown that NFC is more secure (Özcanhan, Dalkılıç & Utku, 2014).

Many Internet Protocol (IP) based protocols are used in IoT communication. Secure data communication is provided by using Message Queue Telemetry Transport (MQTT) protocol, which is key to IoT communication due to lightweight communication protocol and consumption of limited resources. Once the data

generated by Eclipse-paho client are transferred to MQTT Broker called Mosquitto, they are sent to Android mobile applications and Linux machines via the Internet. While open Authorization 2.0 protocol is used for authorization, Hash-based Message Authentication Code (HMAC)-based one-time password method is used for authentication. An alternative solution has been introduced against known attacks by making use of these methods (Yerlikaya & Dalkılıç, 2018).

As a result of multiple trials on authentication in IoT devices, a recent study offers research on technologies, protocols, and applications leading the IoT. The aforementioned study helps readers gain an insight into the general architecture of the different components and protocols constituting IoT. The interaction between IoT, big data analytics, and cloud computing was also discussed within the scope of the study. The differences between IoT application protocols were analyzed (Al-Fuqaha, Guizani, Mohammadi & Aledhari, 2015).

Another study based its conclusions upon experiments on wired, wireless and 2G, 3G, and 4G connections on MQTT, Constrained Application Protocol (CoAP), and Advanced Message Queuing (AMQP) protocols in IoT communication. Bandwidth usage, number of messages transmitted per second, total number of packets generated by protocols were analyzed. Mosquitto and RabbitMQ software were used to implement MQTT and AMQP. While MQTT and AMQP reportedly have good performance in wired and wireless connections, CoAP are less dependent on the network. The study suggests that CoAP generates less traffic on the lossy networks (Chaudhary, Peddoju & Kadarla, 2017).

There is also other research building its conclusions on data transmission experiments over WebSocket and Hyper-Text Transfer Protocol (HTTP). Bandwidth consumption, frame-rate and data transmission time of the protocols were compared using SAGAT (Situation Awareness Global Assessment Technique) parameters. This article is based on the implementation of WebSocket as a teleoperation protocol. As a result of the tests, WebSocket has been suggested to be a more efficient protocol than HTTP (Srinivasan, Scharnagl & Schilling, 2013).

Another study provides an assessment of MQTT, CoAP, AMQP, HTTP, and communication protocols for IoT systems. The protocols have been defined and compared with each other as shown in Table 2.1. The properties, limitations and strengths of the protocols are explained. This study provides information about which protocol to choose according to the appropriate requirements in IoT systems. Message Size vs. Message Overhead, Power Consumption vs. Resource Requirement, Bandwidth vs. Latency, Reliability/QoS etc. Interoperability, Security vs. Provisioning, and M2M/IoT Usage vs. Standardization features are graphically compared (Naik, 2017).

Table 2.1 Analysis of MQTT, CoAP, AMQP and HTTP (Naik, 2017)

Criteria	MQTT	CoAP	AMQP	HTTP
1. Year	1999	2010	2003	1997
2. Architecture	Client/Broker	Client/Server or Client/Broker	Client/Broker or Client/Server	Client/Server
3. Abstraction	Publish/Subscribe	Request/Response or Publish/Subscribe	Publish/Subscribe or Request/Response	Request/Response
4. Header Size	2 Byte	4 Byte	8 Byte	Undefined
5. Message Size	Small and Undefined (up to 256 MB maximum size)	Small and Undefined (normally small to fit in single IP datagram)	Negotiable and Undefined	Large and Undefined (depends on the web server or the programming technology)
6. Semantics/ Methods	Connect, Disconnect, Publish, Subscribe, Unsubscribe, Close	Get, Post, Put, Delete	Consume, Deliver, Publish, Get, Select, Ack, Delete, Nack, Recover, Reject, Open, Close	Get, Post, Head, Put, Patch, Options, Connect, Delete
7. Cache and Proxy Support	Partial	Yes	Yes	Yes
8. Quality of Service (QoS)/ Reliability	QoS 0 - At most once (Fire-and-Forget), QoS 1 - At least once, QoS 2 - Exactly once	Confirmable Message (similar to At most once) or Non-confirmable Message (similar to At least once)	Settle Format (similar to At most once) or Unsettle Format (similar to At least once)	Limited (via Transport Protocol - TCP)
9. Standards	OASIS, Eclipse Foundations	IETF, Eclipse Foundation	OASIS, ISO/IEC	IETF and W3C
10. Transport Protocol	TCP (MQTT-SN can use UDP)	UDP, SCTP	TCP, SCTP	TCP
11. Security	TLS/SSL	DTLS, IPSec	TLS/SSL, IPSec, SASL	TLS/SSL
12. Default Port	1883/ 8883 (TLS/SSL)	5683 (UDP Port)/ 5684 (DLTS)	5671 (TLS/SSL), 5672	80/ 443 (TLS/SSL)
13. Encoding Format	Binary	Binary	Binary	Text
14. Licensing Model	Open Source	Open Source	Open Source	Free
15. Organisational Support	IBM, Facebook, Eurotech, Cisco, Red Hat, Software AG, Tibco, ITSO, M2Mi, Amazon Web Services (AWS), InduSoft, Fiorano	Large Web Community Support, Cisco, Contiki, Erika, IoTivity	Microsoft , JP Morgan, Bank of America, Barclays, Goldman Sachs, Credit Suisse	Global Web Protocol Standard

CHAPTER THREE
SECURITY REQUIREMENTS OF IoT DEVICES

Information security means ensuring the security of processes in information systems. The area is unstable, constantly changing process, and the attack prevention system must be kept up-to-date to ensure security. Features of information systems include numerous arrangements, regarding network features and communication technologies, which provide communication in connection with each other. The development and change of information systems may appear as vulnerabilities. IoT device communication takes place wirelessly in an insecure air environment as seen in Figure 3.1, implying that attacks might be launched by attackers. For this reason, necessary security principles should be established in authentication protocols (Li, Tryfonas & Li, 2016). Our study sets out to present information on precautions against some known attacks.

Figure 3.1 IoT devices in an insecure air environment (Chackak, 2018)

3.1 Security Principles for Electronic Devices in General

3.1.1 Confidentiality

Confidentiality is an essential feature in data communication, and necessary precautions should be adopted against obtaining information by unauthorized persons.

IoT devices, therefore, must prove their identity to each other, while unauthorized devices wishing to participate in the communication should be blocked. By no means should a device unable to prove its identity take part in the communication. Data transmission must be encrypted so that the attackers do not intrude the communication. To protect data from disclosure in protocol communication, encryption is an appropriate solution. The degree of security varies according to the algorithm used in encryption. The IoT device that receives the encrypted data must obtain it by analyzing the encrypted data (Atlam & Wills, 2020).

3.1.2 Integrity

Integrity means that data cannot be changed by unauthorized persons. In the communication of IoT devices, the data must reach the target unchanged. Attackers may want to intrude the communication, change or delete the data. If the IoT devices have sufficient computing power, encryption can be used to provide data integrity. The encryption can provide both checking data integrity and proving message content.

3.1.3 Availability

In information systems, the sources should be accessible and instantly available at any time. The situation is defined as availability. To ensure information and communication security, precautions should be taken against service disrupting attacks such as Denial of Service (DoS) which might prevent the system from launching a response (Mahfouz & Adjei-Quaye, 2017). In business life, the inaccessibility of information may stop processes and operations. Therefore, important problems such as production stoppages can occur.

As shown in Figure 3.2, confidentiality, integrity, and availability are accepted as basic security principles. In addition to the basic principles, other principles are explained below to increase the security of information systems.

Figure 3.2 Basic security principles (Tyson, 2019)

3.1.4 Reliability - Consistency

Communication between devices depends on certain rules. The system only has to carry out the operations it is programmed to do. It should not perform any missing or redundant operations. Reliability is the measurable consistency of the system and operations are processed as described. The combination of consistent transactions creates reliability. The IoT system must ensure consistency between the predicted and expected behavior.

3.1.5 Non - Repudiation

In the communication of two or more devices, communication due to connection problems or hardware reasons may not be completed. In another case, when there is a transfer of information between the two systems, the sender should not deny that it sent the data, nor should the receiver deny that it received the data. It's called non-repudiation.

3.1.6 Authentication

Authentication is the process of proving the identification used in the communication of devices. Firstly, the devices have to verify their identity in order to access the systems. The devices are supposed to provide access control for the systems by checking whether the authorized devices match the credentials in the database.

Necessary communication steps must be completed to ensure mutual authentication. The devices have to verify with each other the security components used in the communication steps.

3.1.7 Authorization

Authorization is the process of accepting permissions to an authenticated identity. Only authorized users should be connected to IoT devices, and access control should be made for the connected people. Each user or device has to operate only within the access authority. Authorization provide security mechanism to such as communication steps, data transmission, and user control. In addition, authorization is to ensure that transactions are carried out according to the degree of authority.

3.1.8 Accounting

The account principle is based on logging the entire operation of the system and providing a control mechanism. It is important to log the works and processes performed in each system and to examine any problems that may occur, so unexpected events likely to occur in network systems and software can be monitored. All the operations performed by users and devices are logged on a date basis so that the system can be alarmed and warned in case of an attack (Popescul, 2011).

3.2 Basic Cryptology Terminology for Security Principles

Cryptology is the encryption of data and the decryption of encrypted data to its original state. The concept is all of the techniques used to ensure information security. As seen in Figure 3.3, Cryptology is examined in two parts as cryptography and cryptanalysis (Coskun & Ülker, 2013).

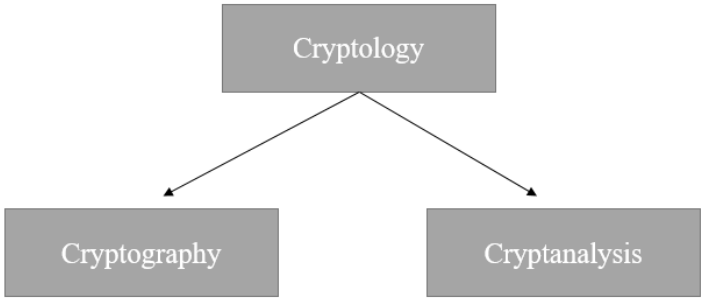

Figure 3.3 Sub-branches of cryptology science

The main purpose of the concept of cryptography is to send the data safely and receive it from the other side. It is the process of converting data from plaintext to ciphertext using encryption algorithms. Cryptography includes all methods for ensuring secure data communication. It contains various components in preventing attacks.

Cryptanalysis is the science of decryption. The concept is based on converting data from ciphertext to plaintext. Cryptanalysis also includes tries to recover information in cases where the key is unknown in ciphertext.

3.2.1 Security Key in Encryption and Decryption

The encryption and decryption of data are provided through the key. The component is used in symmetric and asymmetric encryption approaches. It is the basic element used in creating information security. It consists of a sequence of characters. Data are encrypted or decrypted using the key in communication protocols. For this reason, the privacy of the key is very important for security.

3.2.2 Plaintext in Encryption and Decryption

Plaintext is the original form and open state of data. While the devices are encrypting, they are taken as plaintext input. If ciphertext is sent with plaintext, the ciphertext can be predicted by accessing plaintext. When IoT devices provide mutual authentication, they can decrypt ciphertext by matching identities stored in plaintext.

3.2.3 Ciphertext in Encryption and Decryption

During the encryption process, plaintext is converted to encrypted text. The encrypted text cannot be understood by any device. Ciphertext is encrypted data. Even if the attacker can access encrypted data, they cannot do anything. However, if plaintext is sent along with the ciphertext, the attacker can try a brute force attack.

3.2.4 Encryption and Decryption

Encryption is a method of converting plaintext to ciphertext via key and algorithm. Decryption is to convert ciphertext into plaintext. The transformations are illustrated in Figure 3.4. Encryption and decryption have techniques and applications that involve mathematical expressions. Therefore, encryption has an important role in providing information security in communication protocols. In one-way encryption, after the data is encrypted, it cannot be decrypted. Two-way encryption is the decryption of data after it is encrypted. It consists of two sub-branches as symmetric encryption and asymmetric encryption.

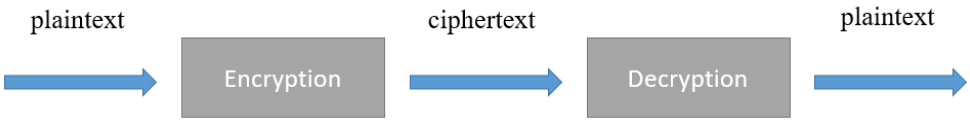

Figure 3.4 Encryption and decryption stage

3.2.4.1 Symmetric Encryption Approaches

The symmetric encryption approach allows the sender and receiver to protect data from other devices by using a private key and encryption and decryption algorithm. In the symmetric encryption method, only the private key used between devices can encrypt and decrypt the data. The process is demonstrated in Figure 3.5. Since only the private key is used in each step of communication, the key must be secret. If an attacker obtains the key used in communication, security vulnerabilities will occur. The attacker can decrypt the ciphertext.

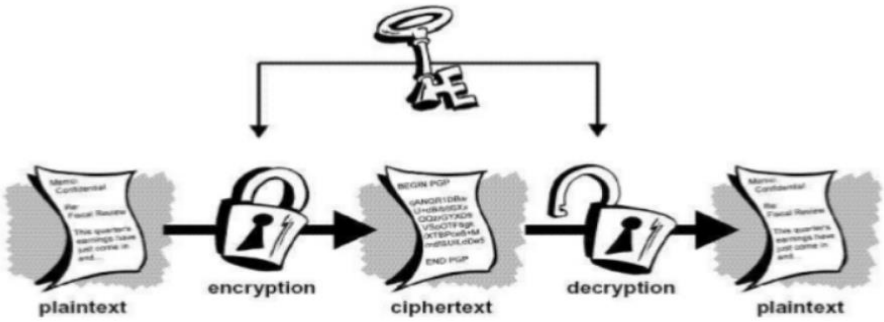

Figure 3.5 Symmetric encryption (Kumar, Munjal & Sharma, 2011)

In the symmetric encryption approach, algorithms are fast, can be used with hardware, and are secure. However, secure key distribution is difficult, there is a capacity problem, it is difficult to perform authentication and integrity principles securely.

Symmetric algorithms are divided into two as block encryption and stream encryption algorithms. Block encryption algorithms process data in blocks. Block cipher are algorithms that enable the encryption of fixed-length blocks. Stream encryption algorithms use the data as a bitstream. Every bit or byte in plaintext is encrypted. Voice communication in noisy environments can be given as an example of usage areas.

Data Encryption Standard (DES) is a block encryption method developed by IBM in 1978. The method divides the open text to be encrypted into blocks and encrypts each fragment independently. It applies the same process on blocks to decrypt. Although the DES algorithm has a key length of 64 bits, it is a system that uses 56 bits long symmetric encryption technique. (Suguna, Dhanakoti & Manjupriya, 2016).

Blowfish is a block cipher method produced in 1993 by Bruce Schneider. It is produced to replace the DES algorithm. It works with a 32-bit to 448-bit long key and has a high-speed performance. It needs more than 4 kilobytes of RAM.

The weakness of DES is insufficient key length and can be decrypted by attackers. It has been developed with a new algorithm known as Triple DES or 3DES to remedy the disadvantage. The key length is at most 168 bits. It is created by applying DES three times. Even if the 3DES algorithm increases security, its performance is slower than other encryption techniques of today.

Advanced Encryption Standard (AES) is a securely accepted method of symmetric encryption approaches. Today, it has validity in ensuring information security. It is internationally recognized as the encryption standard. It became officially active on May 26, 2002. It has replaced DES. It is secure and well-performing with a 128-bit key size, as well as providing stronger encryption using 192 and 256-bit keys. It is implemented on a variety of hardware, from 8-bit processor hardware to high-performance computer-based systems. It takes longer than the age of the world to decrypt the password in computers of today's technology (Padate & Patel, 2014).

3.2.4.2 Asymmetric Encryption Approaches

In the asymmetric encryption approach, devices create their encryption keys which can be defined as private and public. As clearly illustrated in Figure 3.6, the encryption algorithm uses two separate keys: public and private. After the encryption is performed through the public key, the decryption is performed with the private key. The public key is accessible to every device, but the relevant device must prove its ownership of the key. By using certificates, verification is made between the public key and the owner of the key. A new perspective has been brought to the problem of key distribution in encryption.

In asymmetric encryption methods, confidentiality, integrity, and authentication, the main three principles of cryptography, can be provided securely. Asymmetric encryption solution includes difficult mathematical functions. The functions contain prime factors of very large numbers. The reliability of the algorithm increases with the increase in the quantity of prime numbers used. The prime numbers cause slowness in encryption and decryption processes. Rivest-Shamir-Adleman (RSA), Diffie-

Hellman, Digital Signature Algorithm (DSA), Elliptic Curve Cryptography (ECC) are given as examples of asymmetric encryption methods (Beşkirli, Özdemir & Beşkirli, 2019).

Asymmetric encryption algorithms consume too much Central Processing Unit (CPU) and Random Access Memory (RAM) resources. Therefore, these algorithms are not preferable for IoT devices. They mostly provide secure communication by using computers that are unlimited in terms of resources.

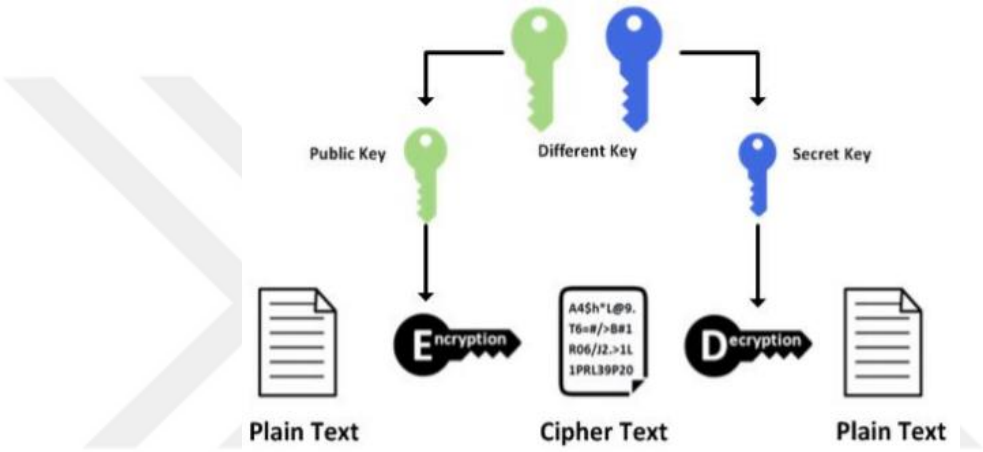

Figure 3.6 Asymmetric encryption (Jallouli, 2017)

3.2.4.3 One-way Encryption Approaches

Apart from symmetric and asymmetric encryption methods, there are also one-way encryption methods. Hash function algorithm is used most in the field. The algorithm is used in areas such as securely encrypted data storage, digital signature, and message verification code. As with encryption methods, original data cannot be recovered. It has functions such as MD5, SHA1, SHA2, SHA3. There is a possibility that different inputs will come out as the same output after passing through the hash function. It is called a collision. A collision case is undesirable and reduces the reliability of the function. The most ideal algorithm should not have collision and cannot convert the encrypted data to its original state (Sobti & Geetha, 2012).

3.3 Symmetric Encryption Algorithms Used in IoT Devices

Cryptography is a combination of applications and techniques that improve the security of only authorized devices by achieving security in the communication of devices with each other. Various algorithms are used for encryption in cryptographic protocols. In order to resist the protocols against attacks, the appropriate key should be selected for IoT devices. Key exchange, one of the parameters in security enhancement, must be made after secure communication is provided. The same key is not used continuously, and measures can be taken against attacks. In our protocol, AES encryption algorithm will be used to provide security requirements. The features of the algorithm are suitable for ensuring data confidentiality. It is accepted as an international security standard.

3.4 The Need for the Pseudo Names in Protocol Design

Random Number Generator is key to secure data communication between IoT devices. The random numbers cannot be predicted and the same number must not be produced again. Pseudo-random number is a method that IoT devices use to verify their identity in communication. As seen in Figure 3.7, the purpose of pseudo randomization is to prevent the IoT devices with limited resources from producing the same number for a certain period in number generation. Using initialization vectors (IV) are carried out for better randomization. However, resource usage should be considered. The method can be preferred according to the computing power of the IoT devices.

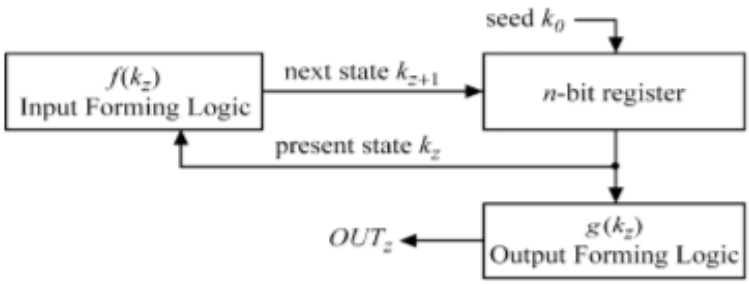

Figure 3.7 PRNG schema (Addabbo, Alioto, Fort, Pasini, Rocchi & Vignoli, 2007)

As is known, each device has an identification number that represents its identity. After the data communication is completed, the repetition of the identities is monitored by the attacker, creating a security problem. The attacker may wish to send a message by imitating the identification numbers, therefore devices should change their identification number (ID) to pseudo-ID at the end of the communication. Therefore, attackers intruding on the network will not be able to realize which devices are communicating (Wallace, Moran, Novak, Zhou & Sun, 2015). In our protocol, pseudo names and numbers are used to increase security in mutual authentication.

CHAPTER FOUR
OUR PROPOSED DESIGN

Sensors and gateways are basic hardware that has functions such as data sending and receiving in IoT communication. Sensors convert measurable values from the environment into meaningful data. These are values such as temperature, humidity, pressure, distance, smoke, motion, etc. Gateway provides IoT communication from device to device or from device to cloud. The gateway contains more resources than the sensor.

Our protocol includes a communication system consisting of a sensor and a gateway that have an ID, key, and random number in the first case. ID and number are processed as pseudo, and pseudo-random numbers are used for authentication purposes. AES is utilized as the encryption algorithm. Our protocol is named ISCMA (Increasing Security in Communication with Mutual Authentication). The parameters and the conventions used in our protocol are given in Table 4.1, while the designed authentication protocol is given in Figure 4.1.

Table 4.1 The symbols used in our protocol design

Symbol	Description
j	Session count number
ID_s^j	Pseudo identification number of sensor, in j^{th} step
ID_s^{j+1}	Pseudo identification number of sensor, in $(j+1)^{th}$ step
Key_s^j	Key of sensor, in j^{th} step
Key_s^{j+1}	Key of sensor, in $(j+1)^{th}$ step
$controller_s$	Sensor authentication verifier
r_s	Pseudo random number generated by sensor
r_g	Pseudo random number generated by gateway
E	Encrypt
D	Decrypt
AES	Advanced Encrption Standart
$E^j_{Key\_s}$	The symmetric encryption with in j step key of sensor
$D^j_{Key\_s}$	The symmetric decryption with in j step key of sensor
$\oplus$	Bitwise (XOR) Operation

The notations of Table 4.1 are explained below:

- The j value refers to the number of each session in ISCMA protocol.
- ID^{j}_{s} indicates the pseudo identification number of the sensor in each session.
- ID^{j+1}_{s} indicates the pseudo identification number of the sensor in the next session. The ID^{j+1}_{s} is generated by the gateway in each session and then sent to the sensor.
- Key^{j}_{s} refers to the symmetric encryption key that the sensor and gateway commonly use in each session.
- Key^{j+1}_{s} refers to the symmetric encryption key that the sensor and gateway commonly use in the next session. The key is generated by the gateway in each session and then sent to the sensor.
- $controller_s$ value verifies that the first session of communication has been completed.
- r_s is required to verify the identity of the sensor to the gateway.
- r_g is required to verify the identity of the gateway to the sensor.
- E shows that symmetric encryption is performed.
- AES symmetric encryption algorithm is used in our protocol.
- $E^{j}_{Key\_s}$ refers to symmetric encryption in each step.
- $D^{j}_{Key\_s}$ refers to symmetric decryption in each step.
- $\oplus$ is a bitwise operation. It is used to complicate the $controller_s$ value.

4.1 Our Protocol Implementation

Our mutual authentication protocol provides communication between the sensors and the gateways. The design of the proposed protocol has been implemented based on security principles. Our protocol is described using expressions in Table 4.1. The protocol is made up of five steps, as seen in Figure 4.1. All the five steps are interconnected to improve security in communication. Each step is explained in detail below.

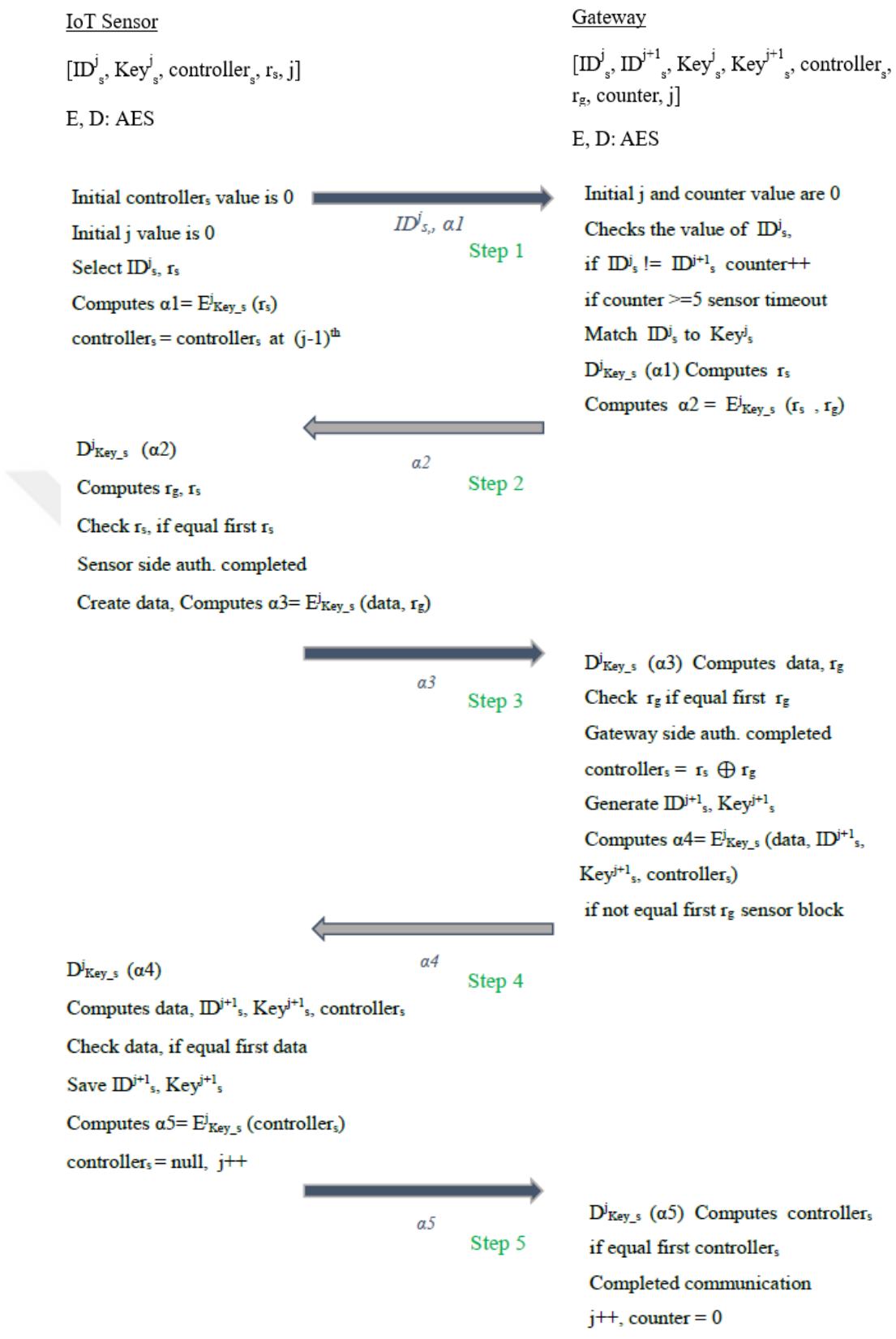

Figure 4.1 Our protocol implementation

4.1.1 Step 1 of Our Protocol Design

Upon starting the communication, the sensor encrypts the number it has generated without encrypting its ID and subsequently sends it to the gateway. The sensor sends its number to the gateway to prove its identity. The controller value, which verifies that all communication steps are finished, is zero in the first case.

As seen in Figure 4.2, after checking the ID sent by the sensor from the database and matching it with the relevant sensor, the gateway decrypts the encrypted data by selecting the appropriate key in the database. The gateway then sends a response by encrypting the number of the sensor and the number it generates in order to verify its ID to the sensor. Using the counter, the gateway checks the sensor ID changing in each session of communication and warns the system in case of an attack by blocking the sensor.

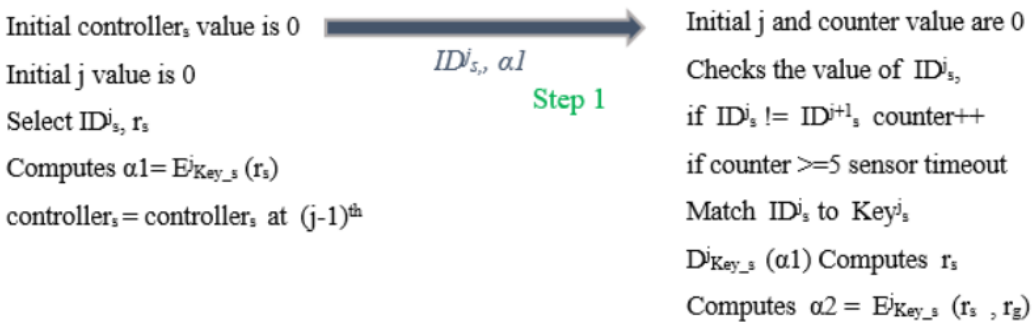

Figure 4.2 Step 1 of our protocol design

4.1.2 Step 2 of Our Protocol Design

Figure 4.3 shows that the sensor decrypts the incoming encrypted packet through its key. When the sensor opens the packet, it possesses the sensor number in the packet as well as the number of the gateway. In the first step, it tries to validate the number it has sent with the sensor number in the incoming packet. Once it is established that the two numbers are matched correctly, authentication on the sensor side is completed. After the processes, the sensor generates data and has to send the number coming from the gateway in order to prove itself to the gateway. The sensor then encrypts and sends

the number along with the data it has created. Then, the gateway must decrypt the packet and prove the identity of the sensor.

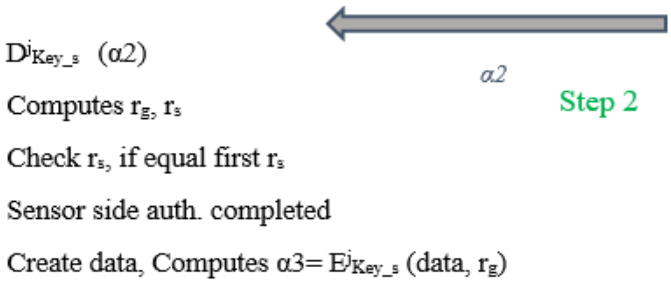

Figure 4.3 Step 2 of our protocol design

4.1.3 Step 3 of Our Protocol Design

In the third step of the communication, the gateway opens the encrypted packet from the sensor and checks whether the gateway number coming from the sensor and the gateway number in the first case is the same. If the two numbers are identical, the authentication process is completed successfully. The controller is then assigned by carrying out an exclusive-or process of the sensor and gateway number. By not giving the controller a fixed value, the attacker is prevented from recording the same value in the event of a possible attack.

The new key and new ID of the sensor to be used in the new session are generated by the gateway and sent to the sensor for use since the sensor has limited resources and can perform limited operations. The key is updated with each new session because it must be private. If the key is obtained by the attacker, the encrypted packet can be decrypted. The gateway encrypts the data transmitted from the sensor, new ID, new key, and controller. The gateway sends it to the sensor shown in Figure 4.4. The reason for sending the data from the sensor to the same sensor again is to check whether the data is transmitted correctly.

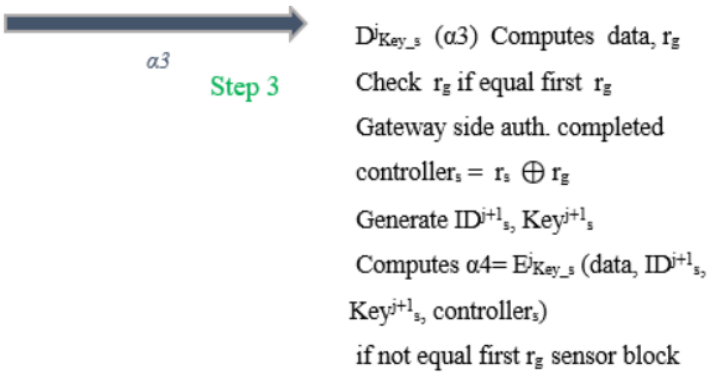

Figure 4.4 Step 3 of our protocol design

4.1.4 Step 4 of Our Protocol Design

In the fourth step of the communication, the sensor checks the data it produces by opening the encrypted packet with the data coming from the gateway. If both data are the same, it indicates that the data is properly transmitted by encrypting.

As indicated in Figure 4.5, the sensor saves new ID and key because it will use the components in the new session. The sensor uses resources efficiently by using the new key and ID generated by the gateway in the next session. Since the protocol continues as defined, only the controls value is encrypted and sent in the next step. The first session of communication is over on the sensor side.

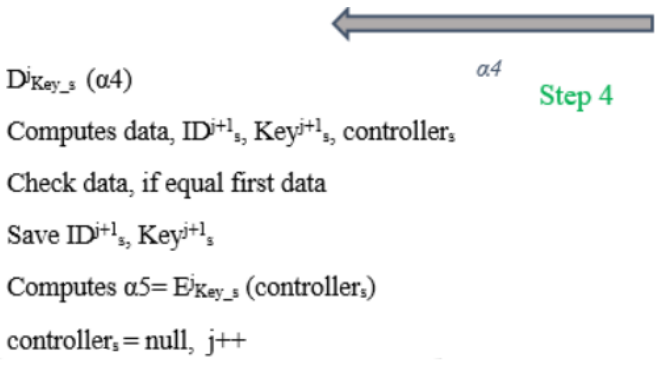

Figure 4.5 Step 4 of our protocol design

4.1.5 Step 5 of Our Protocol Design

In the last step of the communication, the gateway opens the encrypted packet by performing security checks through the same key. As seen in Figure 4.6, the gateway completes the communication by verifying both the first controller value and controller value coming from the sensor. In the next session, our protocol does not repeat itself using the new identity, key, controller, and counter values. Security measures are increased through the components and the AES encryption.

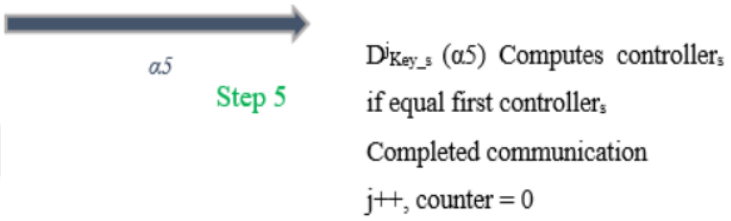

Figure 4.6 Step 5 of our protocol design

CHAPTER FIVE
TESTING OUR DESIGN AGAINST SOME KNOWN IoT ATTACKS

5.1 Attack Types Resisted in Our Design

Cybersecurity is the protection of computer-based systems from malicious attacks. The attacks are aimed at accessing, modifying, deleting sensitive information, and disrupting business processes. Our proposed protocol is resistant to some known attacks. The section explains how ISCMA protocol is resistant to the attacks. In the next chapter, the protocol was analyzed using Scyther, an automatic formal verification tool.

5.1.1 Malicious Code Injection Attack

Malicious Code Injection is a type of attack in which communication by unauthorized people has listened, and harmful codes are placed on the devices as illustrated in Figure 5.1. Incorrect authentication can also try to change data exchange operations, hence the encryption algorithm should be chosen to be strong, and every step of the communication should be interconnected (Deogirikar & Vidhate, 2017). As the length of the encryption key length increases, the power of encryption increases. However, resource usage increases accordingly. In ISCMA protocol, resistance is provided by using the AES encryption algorithm. An attacker who does not know the key will not be able to intervene. The key must be kept secret during communication and key exchange is provided in each new session of communication.

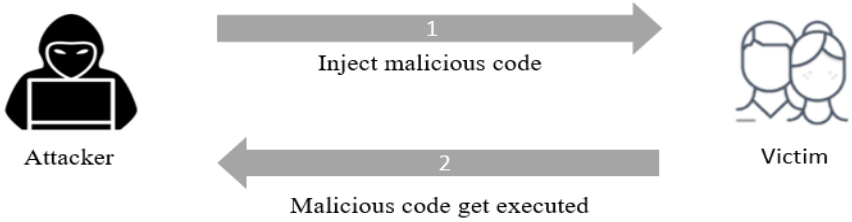

Figure 5.1 Malicious code injection

5.1.2 Man-in-the-Middle Attack

A man-in-the-middle attack, which captures packets on the network and obtains information, is the interception of communication between the two connections in the network and the capture of various data. Figure 5.2 shows that the attacker aims to obtain all the traffic that occurs in the network. In our study, data transmission is not possible without authentication. By providing a strong encryption algorithm, key exchange, and authentication methods among IoT devices, an unauthorized device can be prevented from participating in the network. The attacker can only capture the encrypted packet during communication. However, the attacker will not be able to decrypt the encrypted packet.

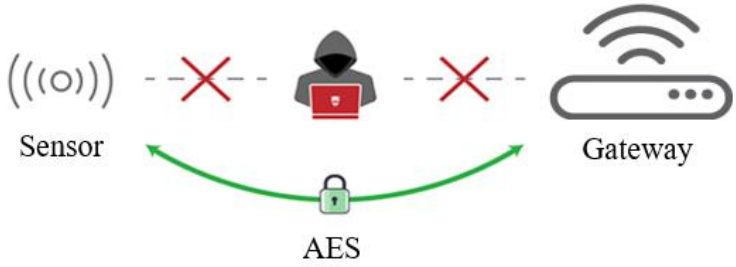

Figure 5.2 Man-in-the-middle-attack

5.1.3 DoS/DDoS Attack

Denial of Service (DoS) or Distributed Denial of Service (DDoS) is an attack method aimed at consuming the resources of the devices and disrupting the service. The attacker renders the system inoperable or slows it down by sending packages to the system regularly and continuously, which makes the devices unavailable as well as ceasing the communication as shown in Figure 5.3. If an attacker makes a repeated request from a device, it is called a DoS, and if multiple devices are attacked, it is called a DDoS attack (Özcanhan, Dalkılıç & Utku, 2014). In such cases, protocols should be analyzed, and control mechanisms should be defined in order to know the attacks in advance and then mitigate them. In our protocol can be warned from the

attack by using a counter mechanism and disabled ID. When an attack is detected, it can be disabled attacking devices.

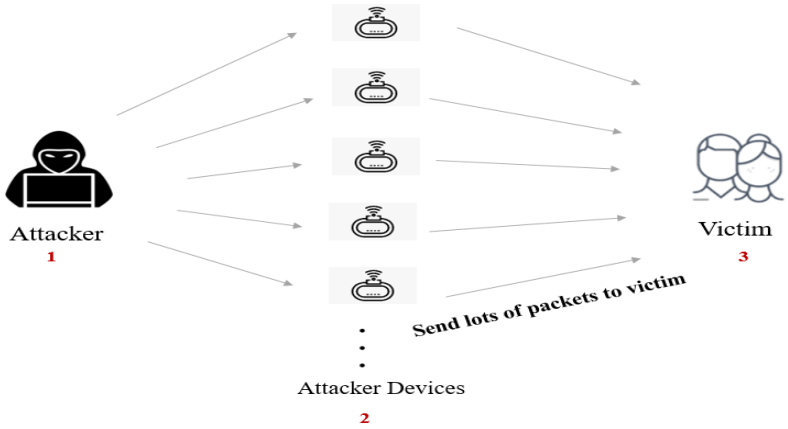

Figure 5.3 DoS/DDoS attack

5.1.4 Replay Attack

Replay attack refers to an attacker following the steps in communication and reusing the packets as demonstrated in Figure 5.4. In the attack, during the communication between IoT sensor and gateway, the attacker captures the packets and sends them to the receiver repeatedly. To gain the fake trust of the system, an attacker usually sends a packet received by the target device. The situation is mostly used in authentication process (Rughoobur & Nagowah, 2017). Even if the attacker captures the packets, the attacker cannot obtain the contents of the package due to the encryption algorithm used. In addition, measures are taken by using mechanisms, such as pseudo-number generator, and counter. After using pseudo-IDs and numbers in communication, they are disabled and changed. Therefore, the attacker cannot attack using the same ID every time.

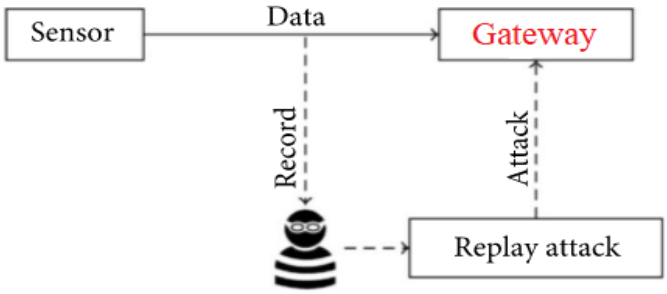

Figure 5.4 Replay attack

5.1.5 Desynchronization Attack

A desynchronization attack is the interruption of one of the communication steps and sending a different packet. In our study, each step has to occur sequentially to complete the communication between the sensor and the gateway. There are common components in the content of each step that connect the packet with the previous and next steps. Our protocol takes precautions against the attack by checking random numbers, data value, and controller value produced by the sensor and gateway.

5.1.6 Identity Spoofing Attack

Identity spoofing is the type of attack in which the attacker pretends to be a device on the network. Even if an unknown device shows itself as a sensor or a gateway in our protocol, it cannot open the encrypted package because of secret key information. Sensor ID and key are changed each session by our protocol after the communication is completed. When the communication is completed in our protocol, the old ID is disabled. If the old ID is sent again by an unknown device, our protocol detects attacks. Therefore, the gateway blocks the attacking sensor.

5.2 Attack Types Outside Our Scope

Since our protocol design is not an IP based communication structure, such attacks are not related to our protocol. Our protocol is applied up to the second layer of the Open Systems Interconnection (OSI) model. We have no claim that our protocol is one hundred percent secure. Some attacks may be possible in our protocol.

5.2.1 Side-Channel Attack

The attack tries to obtain sensitive information by reaching the circuits of the electronic devices or through its data such as timing information, power consumption,

electromagnetic waves, temperature values, and sounds. When the electromagnetic information is linked with confidential information in the device, it is called side-channel information. The attack can threaten data confidentiality, integrity, and availability (Ordu & Örs, 2006).

5.2.2 Tampering Attack

Tampering attack aims to obtain information by reaching circuits of the hardware. The attack enters the area of the devices and physically interferes with their nodes. Information can be stolen by applying components such as probes or protocol analyzers to the circuits of the devices. To prevent information theft, IoT devices can be physically hidden, making them difficult to find. In the event of an attack, the information may be deleted or communication may be interrupted.

5.2.3 Jamming Attack

The attack jams the communication network by generating a strong signal. There are different attack types such as constant, deceptive, random, and reactive jammer. The attack is effective against all types of networks. The attack can occur by sending continuous radio signals or sending deceptive packet. Using methods such as packet delivery rate, signal level, and frequency mixing techniques determines whether the communication is abnormal.

CHAPTER SIX
MATERIALS AND METHODS

6.1 Formal Testing by Using Scyther Tool

Cas Cremers developed the Scyther tool in 2007. Scyther is a structure prepared for the requirements of formally analyzing security protocols and vulnerabilities of security protocols. An important feature of the Scyther tool is that it compiles multiple protocols. As seen in Figure 6.1, the Scyther tool includes a graphical user interface (GUI) and its descriptions are in Security Protocol Description Language. It is based on algorithms that examine the protocol automatically and can analyze most possible some known attacks. Scyther tool can be used to find vulnerabilities arising from the construction of the protocols

When the protocols are analyzed using the Scyther tool, possible attacks are found on the sender and receiver sides. If there is a vulnerability in the protocol, an attack graph is created and tracked by the Scyther tool. If the Scyther output is falsified, there is at least 1 attack. If the Scyther output is verified, there are no attacks or no attacks within bounds (Dalal, Shah, Hisaria, & Jinwala, 2010).

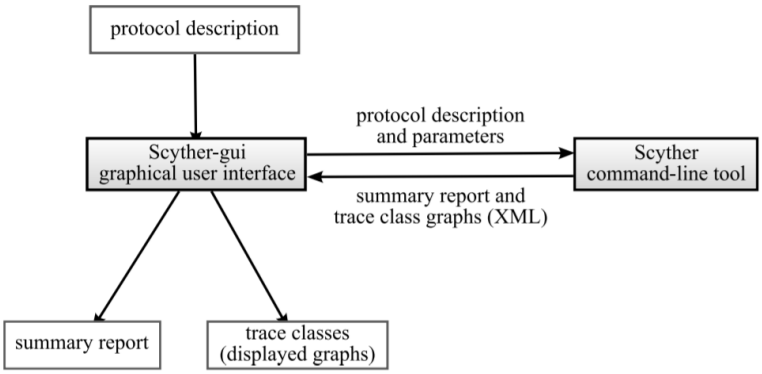

Figure 6.1 Scyther graphical user interface diagram (Cremers, 2006)

The Scyther tool analyzes protocols to provide information about whether they are secure or not. There are many protocols analyzed using the Scyther tool. These analyses are available on the official website of Scyther. The tool also shows possible

attacks that have occurred as a flowchart. Using the Scyther tool, known as IKEv1, IKEv2 protocols and such as ISO / IEC 11770, ISO / IEC 9798 standards can be analyzed. In addition, an example of protocol analysis using the Scyther tool is IEEE 802.16 standard Worldwide Interoperability for Microwave Access (WiMAX), which is a wireless broadband system (Kahya, Ghoualmi & Lafourcade, 2012), Secure Mobile Banking Authentication Scheme (S-Mbank) (Putra, Sadikin & Windarta, 2017), a Lightweight Authenticated Time (LATe) Synchronization Protocol, which is based on Internet Engineering Task Force (IETF) standards (Navas & Toutain, 2018).

Installing the Scyther tool in the Windows operating system is more difficult than installing it on Linux-based systems. Therefore, virtualization technology is used in our study to facilitate the installation phase. There are many platforms on which virtualization technology is applied. OraclaVM VirtualBox, which is one of the free virtualization solutions, has been preferred in our work.

6.2 Virtualization Platform by Oracle VM VirtualBox

Virtualization refers to the operation of multiple operating systems by sharing the same physical equipment resources. Virtualization systems create a logical layer between the user and the hardware, preventing the user from directly accessing physical system resources. The systems reduce the number of physical equipment needed. It ensures the use of system resources with high efficiency and reduces hardware costs. For this reason, virtualization technology is widely used today.

Oracle VM VirtualBox is free software that enables virtual machines to run using the virtualization technology of processors. It is suitable for both x86/x64 architectures. Supported host operating systems include Windows, Linux, and macOS. Oracle VM VirtualBox installation can be installed free of charge when accessing its official web page (Oracle, 2021). In our research, the current operating system we use is the 64-bit professional version of Windows 10. Figure 6.2 shows that the virtualization platform is created by installing Oracle VM Virtualbox version 6.1. In our work, Scyther was installed by Debian operating system, a Linux distribution.

Minimum system requirements should be 1-gigabyte RAM, 10-gigabyte hard drive space (Debian, 2021).

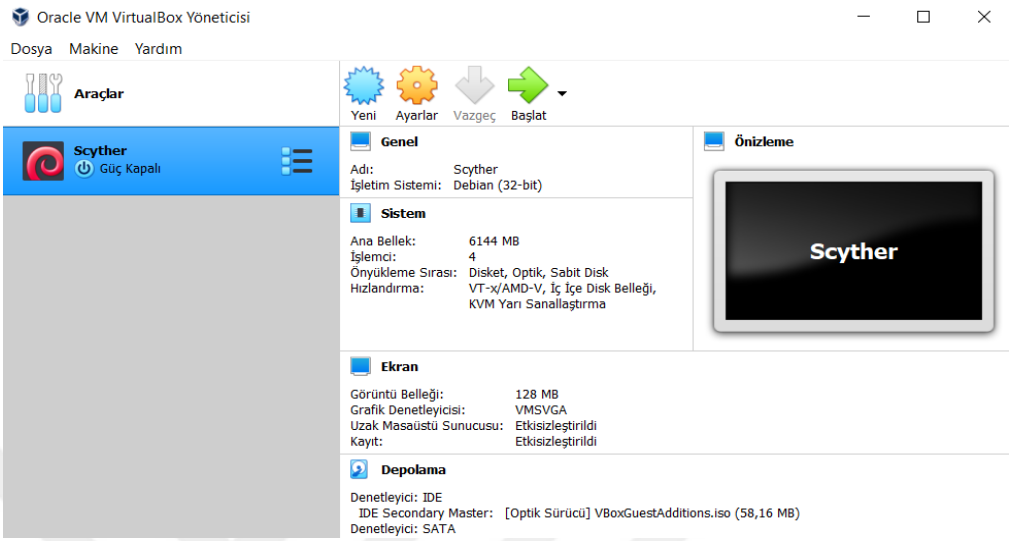

Figure 6.2 Oracle VM VirtualBox

6.3 Installation of Scyther Tool

Scyther tool is available for the Windows, Linux, and Mac OS platforms. Older versions of the tool are 1.0, 1.1, and 1.2. The latest stable version of the Scyther tool, which is used for testing our protocol, is version 1.1.3, released on April 4, 2014. Scyther tool installation can be installed free of charge by accessing the Scyther web page (Cispa, 2021). The GraphViz library is used by the Scyther tool to draw graphs. There are Graphviz installations for different Linux-based systems. Executable packages are below:

- Ubuntu distribution for command $ sudo apt install graphviz
- Fedore distribution for command $ sudo yum install graphviz
- Debian distribution for command $ sudo apt install graphviz
- Redhat Enterprise or CentOS distribution for command $ sudo yum install graphviz

Python language is used in Scyther's graphical user interface. However, 3 versions of the Python language are not supported by the Scyther tool. It using the wxPython

library to draw widgets. Scyther tool is started by executing the scyther-gui.py command in the Scyther directory (Scyther, 2021).

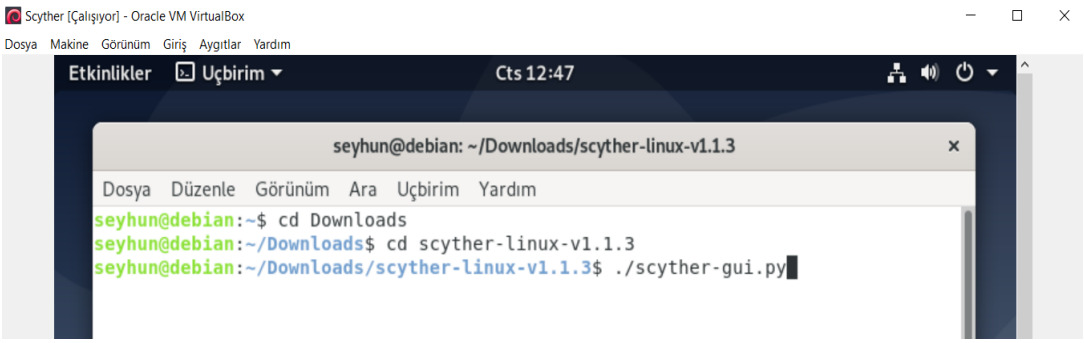

Figure 6.3 Scyther run commands

In the study, the relevant file path and command were executed as follows. It is easy to run the Scyther tool by typing the 3 commands below. Figure 6.4 shows that the Scyther tool version 1.1.3 is running.

- cd Downloads
- cd scyther-linux-v1.1.3
- ./scyther-gui.py

Figure 6.4 Scyther v1.1.3 interface

6.4 Scyther Tool Language and Definitions

Scyther is limited to a fixed cryptographic set and consists of basic elements. For example, XOR operation and global counters cannot be defined in Scyther. The main

purpose of the language is to create protocols defined by a set of roles. Some keywords by Scyther in ISCMA protocol are explained below (Scyther, 2021). The keywords were used in our algorithm design for testing our protocol.

- send and recv: The send and recv keywords represent sending and receiving a packet, respectively. Each send event has a corresponding recv event.
- match: The keyword checks the accuracy of the sent and received data by matching them.
- claim: It is used to perform security tests.
- Non-injective agreement (Niagree): Niagree's feature proves that packets are transmitted securely and sequentially. In the event of a possible known attack, the message transmitted between the parties cannot be decrypted or re-sent.
- Non-injective synchronization (Nisynch): Nisynch proves that one of the communicating devices believes that the message it receives is from an authenticated device. Whether the mutual authentication is successful or not is known by the statement.
- Alive: Alive aims to ensure that an intended communication party has executed its events. It is an authentication form. Alive means that the devices are connected to each other in the communication of the devices. In addition, the data operations of the parties are executable.
- Secret: Secret is proof that the data was sent encrypted. It states that the data cannot be accessed by unknown devices as plaintext.
- Session-Key Reveal (SKR): It states that an encryption key is valid only in one session. In this way, the same key is not used in other sessions.
- Symmetric-key infrastructure: The structure is shown as the predefined k. k is the key in the session.
- Commit: It is based on monitoring and running signal of security components in communication protocol. It provides whether a matching variable exists in tracing data accuracy.
- Weakagree: It proves that the protocol resists impersonation attacks. An impersonation attack occurs when an unknown device is involved in communication. If protocols have not mutually authenticated, an attacker could

use the vulnerabilities. An attacker could implement harmful components in the protocol.

6.5 Testing Our Protocol Using Scyther Tool Language

In order to prove how secure our protocol is, the protocol has been analyzed by the Scyther tool. The Scyther tool includes roles and data types to define different security protocol components shown in Figure 6.5. Communication roles are defined when creating the protocol. In our study, the roles are defined as IoTSensor and Gateway. The usertype keyword is used to create definitions such as string. The fresh variable is used when generating random numbers. It is used only in one session. The keyword var refers to a variable. It stores components sent by roles in communication (Yang, Prinz, & Oleshchuk, 2016). // statement is used to comment in the code.

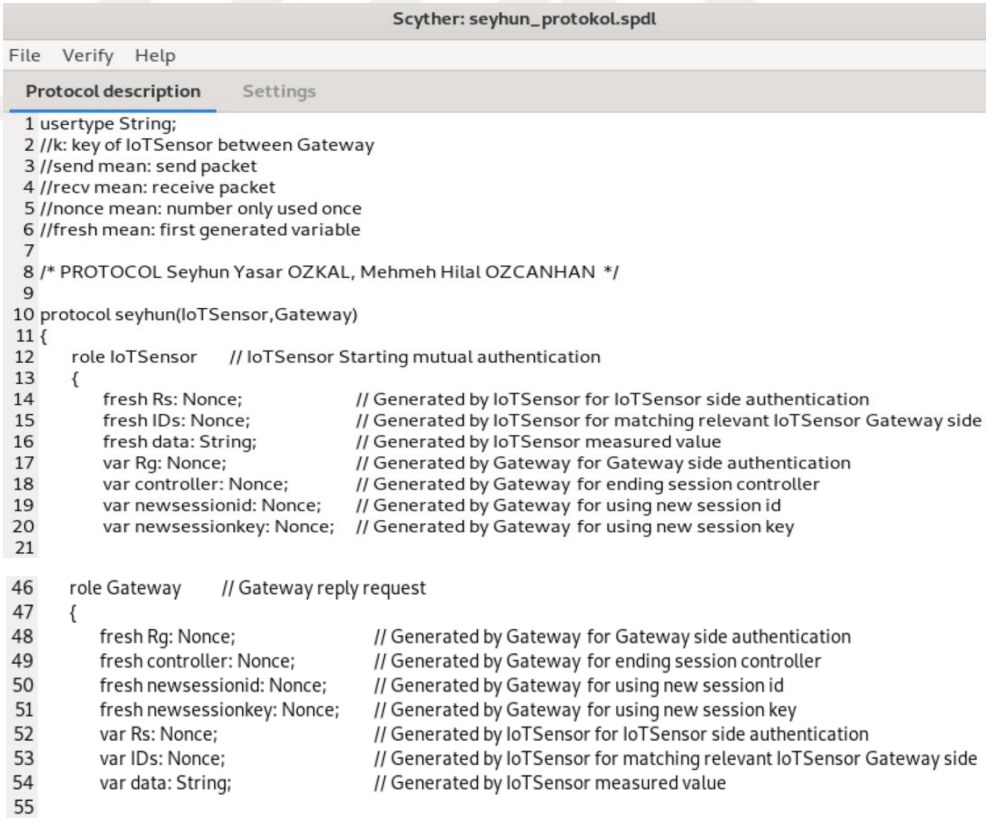

Figure 6.5 Description of variables for our protocol by scyther tool

The source code in the Scyther tool of our protocol design is in the Appendix section. The code segments of our protocol in the Scyther tool are explained in detail below:

6.5.1 Description of Code Segment 1 for IoTSensor Side

Our protocol has IotSensor and Gateway as roles in the Scyther tool. The key used in symmetric encryption is shown as k. The fresh variable type is used to define the first generated variables. The number only used once (nonce) variable type indicates that the generated number can only be used once. The send keyword means that roles in the protocol send their packets. The keyword recv refers to the roles in the protocol receiving the packet sent to it.

Figure 6.6 shows that a random number R_s that the sensor uses for mutual authentication is generated. The sensor generates its pseudo-ID_s to identify itself to the gateway. Since our protocol uses the symmetric encryption approach, it uses one key each session. The ID_s is sent to match the key of the relevant sensor in the gateway's database, R_s is sent as encrypted for authentication purposes.

```
20
21      fresh Rs: Nonce;            // Generated by IoTSensor for IoTSensor side authentication
22      fresh IDs: Nonce;           // Generated by IoTSensor for matching relevant IoTSensor Gateway side
23
24      send_1(IoTSensor,Gateway,IDs,{Rs}k(IoTSensor,Gateway));
25      //ID send for matching relevant key, Rs send for authentication
26
```

Figure 6.6 Description of code segment 1 for IoTSensor side by Scyther tool

6.5.2 Description of Code Segment 2 for Gateway Side

The gateway receives the packet from the sensor. The gateway decrypts the packet by choosing the appropriate key according to the sensor's ID. It obtains ID_s and R_s. In order to use the ID_s and R_s in the packet coming from the sensor, the variables are saved as var type variables. Then, it generates its own random number R_g that it will

use for authentication. It sends the number and the number from the sensor by encrypting it together shown in Figure 6.7.

```
58
59      var IDs: Nonce;                 // Generated by IoTSensor for matching relevant IoTSensor Gateway side
60      var Rs: Nonce;                  // Generated by IoTSensor for IoTSensor side authentication
61      fresh Rg: Nonce;                // Generated by Gateway for Gateway side authentication
62
63      recv_1(IoTSensor,Gateway,IDs,{Rs}k(IoTSensor,Gateway));
64      //ID recv for matching relevant key, Rs recv for sensor authentication
65
66      send_2(Gateway,IoTSensor,{Rs,Rg}k(Gateway,IoTSensor));
67      //Rs send for matching and verifying first Rs, Rg send for gateway authentication
68
```

Figure 6.7 Description of code segment 2 for Gateway side by Scyther tool

6.5.3 Description of Code Segment 3 for IoTSensor Side

The sensor receives the packet from the gateway and decrypts it with its key. As a result of this decryption, it obtains R_s and R_g. Then, the sensor matches and validates the obtained R_s against the initial R_s. R_g will be used to authenticate the gateway with the sensor in the next step. In order to use R_g in the packet coming from the gateway, the variable is saved as var type variables. When the sensor matches R_s and the first R_s, it authenticates to the gateway. After successful authentication, it generates data that can be measured from the physical environment. As seen in Figure 6.8, the sensor continues to communicate by encrypting the data and R_g sending it to the gateway.

```
27
28      var Rg: Nonce;                  // Generated by Gateway for Gateway side authentication
29      fresh data: String;             // Generated by IoTSensor measured value
30
31      recv_2(Gateway,IoTSensor,{Rs,Rg}k(Gateway,IoTSensor));
32      //Rs recv for matching and verifying first Rs, Rg recv for gateway authentication
33
34      match(Rs,{Rs}k(Gateway,IoTSensor));
35      //Rs verify OK, IoTSensor side authentication completed
36
37      send_3(IoTSensor,Gateway,{Rg,data}k(IoTSensor,Gateway));
38      //Rg send for for gateway authentication, data send for measurable value
39
```

Figure 6.8 Description of code segment 3 for IoTSensor side by Scyther tool

6.5.4 Description of Code Segment 4 for Gateway Side

The gateway decrypts the packet transmitted from the sensor. It obtains data and R_g. In order to use data in the packet coming from the sensor, the variable is saved as var type variables. When the gateway matches R_g and the initial R_g, it authenticates to the sensor. In this way, mutual authentication is completed successfully. Gateway generates the newsessionid, newsessionkey for use in the new session. It also generates the $controller_s$ to check whether each session of communication is over. Gateway re-sends the data to the sensor to check if it is generated correctly and transmitted to it. As the last operation in the step, the gateway sends data, newsessionid, newsessionkey and $controller_s$ as ciphertext as shown in Figure 6.9.

```
78
79      var data: String;                   // Generated by IoTSensor measured value
80      fresh controller: Nonce;            // Generated by Gateway for ending session controller
81      fresh newsessionid: Nonce;          // Generated by Gateway for using new session id
82      fresh newsessionkey: Nonce;         // Generated by Gateway for using new session key
83
84      recv_3(IoTSensor,Gateway,{Rg,data}k(IoTSensor,Gateway));
85      //Rg recv for matching and verifying first Rg, data recv for measurable value
86
87      match(Rg,{Rg}k(IoTSensor,Gateway));
88      //Rg verify OK, Gateway side authentication completed
89
90      send_4(Gateway,IoTSensor,{data,newsessionid,newsessionkey,controller}k(Gateway,IoTSensor));
91      //send  for matching and verifying first data, send  for newsessionid and newsessionkey next session
92
```

Figure 6.9 Description of code segment 4 for Gateway side by Scyther tool

6.5.5 Description of Code Segment 5 for IoTSensor Side

Figure 6.10 shows that the sensor receives the packet from the gateway and decrypts it. Sensor obtains data, newsessionid, newsessionkey, and $controller_s$. When the sensor matches data and the initial data, it verifies the right data value. In order to use newsessionid, newsessionkey, and $controller_s$ in the packet coming from the sensor, the variable is saved as var type variables. Thus, all variables are defined in the Scyther tool. The sensor will use the newsessionid and newsessionkey generated by the gateway in the next session. Therefore, the sensor uses its resources efficiently and security is increased because the same values are not used again. Since the protocol

continues as defined, only the $controller_s$ is sent encrypted. The first session of communication on the sensor side is completed.

```
38
39      var newsessionid: Nonce;     // Generated by Gateway for using new session id
40      var newsessionkey: Nonce;    // Generated by Gateway for using new session key
41      var controller: Nonce;       // Generated by Gateway for ending session controller
42
43      recv_4(Gateway,IoTSensor,{data,newsessionid,newsessionkey,controller}k(Gateway,IoTSensor));
44      //recv for matching and verifying first data, recv for newsessionid and newsessionkey next session
45
46      match(data,{data}k(Gateway,IoTSensor));
47      //Data verify OK
48      send_5(IoTSensor,Gateway,{controller}k(IoTSensor,Gateway));
49      //controller is sended for receiving the data correctly by gateway
50
```

Figure 6.10 Description of code segment 5 for IoTSensor side by Scyther tool

6.5.6 Description of Code Segment 6 for Gateway Side

In the last segment of the communication, the gateway decrypts the packet transmitted from the sensor. It obtains $controller_s$ value. When the gateway matches the $controller_s$ and the initial $controller_s$ value, it verifies all communication components. All the process of the protocol is completed and communication is provided securely as indicated in Figure 6.11. In the new session of the protocol, all variables are renewed and do not repeat themselves.

```
 99
100      recv_5(IoTSensor,Gateway,{controller}k(IoTSensor,Gateway));
101      //controller recv for matching and verifying first controller and finish session
102      match(controller,{controller}k(IoTSensor,Gateway));
103      //controller verify OK, Completed communication
104
```

Figure 6.11 Description of code segment 6 for Gateway side by Scyther tool

6.5.7 Security Analysis of Our Protocol

Security analysis of our protocol design was carried out by the Scyther tool formally. IoT sensor and gateway sides have been tested separately. Communication is provided by encrypting all components except the ID_s of the sensor, which is pseudo.

The ID is only used to match the key with the relevant sensor in the gateway's database. Using the claim keyword, communication steps have been tested sequentially (Figure 6.12). Our proposed protocol has been evaluated from Secret, Alive, Weakagree, SKR, Niagree, Nisync and Commit tests by the Scyther tool. All of these claim statements used in testing our protocol are explained one by one below:

- The secret claim requires secrecy for given parameters such as Rs, Rg, data, newsessionid, newsessionkey, and controllers value. Confidentiality is ensured by using the symmetric encryption approach.
- The Alive claim shows that IoT sensor and gateway are connected to each other. Data operations of the IoT sensor and the gateway are executable commands.
- The Weakagree claim proves that our protocol resists impersonation attacks. It is claimed that an unknown device cannot be involved in communication.
- The SKR claim mentions that an encryption key generated by the gateway for use in subsequent sessions is sent to the IoT sensor in encrypted form. The encryption key is used only in one communication session, and the same key is not used in other sessions.
- The Niagree claim proves that packets are transmitted securely and sequentially in communication between IoT sensor and gateway. In the event of a possible known attack, the packets transmitted between the parties cannot be decrypted or re-sent.
- The Nisynch claim shows that IoT Sensor and gateway believes that the message it receives is from an authenticated device. In our protocol, the IoT sensor is an initiator and the gateway is a responder. Since mutual authentication of our proposed protocol is successfully performed, it is resistant to known attacks. Thus, the packets sent and received in our protocol are completed for mutual authentication.
- Commit claim checks whether a matching variable exists in tracing data accuracy in communication protocols. Rs, Rg, data, and controllers values used in our protocol are variables that prove the accuracy in communication. Since these variables are interconnected in all steps of our five-step protocol, they can also prove the accuracy of all data sent encrypted. Thus, the IoT sensor and gateway take early precautions against attacks through these variables.

Scyther formal test tool shows possible attacks on the interface as a flowchart. However, since the test tool is a fixed cryptographic set, it only classifies status and comment in communication protocols where attacks do not occur.

Rs, Rg, data, newsessionid, newsessionkey, and controller value which are the components that make up our protocol, are sent encrypted. Our protocol is resistant to man-in-the-middle attack using encryption algorithm. The Scyther tool has tested it with Secret claim and qualified it as secure.

The use of mutual authentication and encryption algorithm in our protocol indicates that malicious code injection is resistant to attack. Analysis of Weakagree and Nisynch claim shows resistance to the attack.

In our protocol, the old ID and encryption key are disabled after each session. As a result of the analysis of the SKR claim, our protocol can be designated as resistant to Identity Spoofing Attack.

Each communication step in our protocol is interconnected. Pseudo random numbers, data, and controller value are matched and checked after each step in which they are processed. Commit claim confirms these matches. Niagree claim's analysis states that each communication step is secure and sequential. In this way, our protocol has been shown to be resistant to desynchronization attack.

The security variables and encryption key in our protocol change with their new value in each session. For a communication session to be completed, the communication steps must occur in sequence. As a result of the analysis of SKR and Niagree claim, resistance to replay attack can be achieved in our protocol.

```
51
52        //Verification of  Our Protocol for IoTSensor Side
53
54        claim_IoTSensor1(IoTSensor,Secret,Rs);
55        claim_IoTSensor2(IoTSensor,Secret,Rg);
56        claim_IoTSensor3(IoTSensor,Secret,data);
57        claim_IoTSensor4(IoTSensor,Secret,newsessionid);
58        claim_IoTSensor5(IoTSensor,Secret,newsessionkey);
59        claim_IoTSensor6(IoTSensor,Secret,controller);
60        claim_IoTSensor7(IoTSensor,Alive);
61        claim_IoTSensor8(IoTSensor,Weakagree);
62        claim_IoTSensor9(IoTSensor,SKR,k);
63        claim_IoTSensor10(IoTSensor,Niagree);
64        claim_IoTSensor11(IoTSensor,Nisynch);
65        claim_IoTSensor12(IoTSensor,Commit,Gateway,Rs,Rg,data,newsessionid,newsessionkey,controller);
66
```

```
107
108       //Verification of  Our Protocol for Gateway Side
109
110       claim_Gateway1(Gateway,Secret,Rs);
111       claim_Gateway2(Gateway,Secret,Rg);
112       claim_Gateway3(Gateway,Secret,data);
113       claim_Gateway4(Gateway,Secret,newsessionid);
114       claim_Gateway5(Gateway,Secret,newsessionkey);
115       claim_Gateway6(Gateway,Secret,controller);
116       claim_Gateway7(Gateway,Alive);
117       claim_Gateway8(Gateway,Weakagree);
118       claim_Gateway9(Gateway,SKR,k);
119       claim_Gateway10(Gateway,Niagree);
120       claim_Gateway11(Gateway,Nisynch);
121       claim_Gateway12(Gateway,Commit,IoTSensor,Rs,Rg,data,newsessionid,newsessionkey,controller);
122
```

Figure 6.12 Our protocol verification by the Scyther for Sensor and Gateway sides

The statement “no attacks within bounds” means that the attacks will not be successful at the security boundaries in the Scyther tool. However, there may be a possibility of an attack, except for the scope of the Scyther tool. The statement “no attack” proves that the attack will not succeed even in bounded or unbounded situations. If the attack was successful in our proposed protocol, it would result in a comment of “at least X attack(s)” in the Scyther tool. The results of our protocol tested by using the Scyther tool are satisfactory, as shown in Figure 6.13.

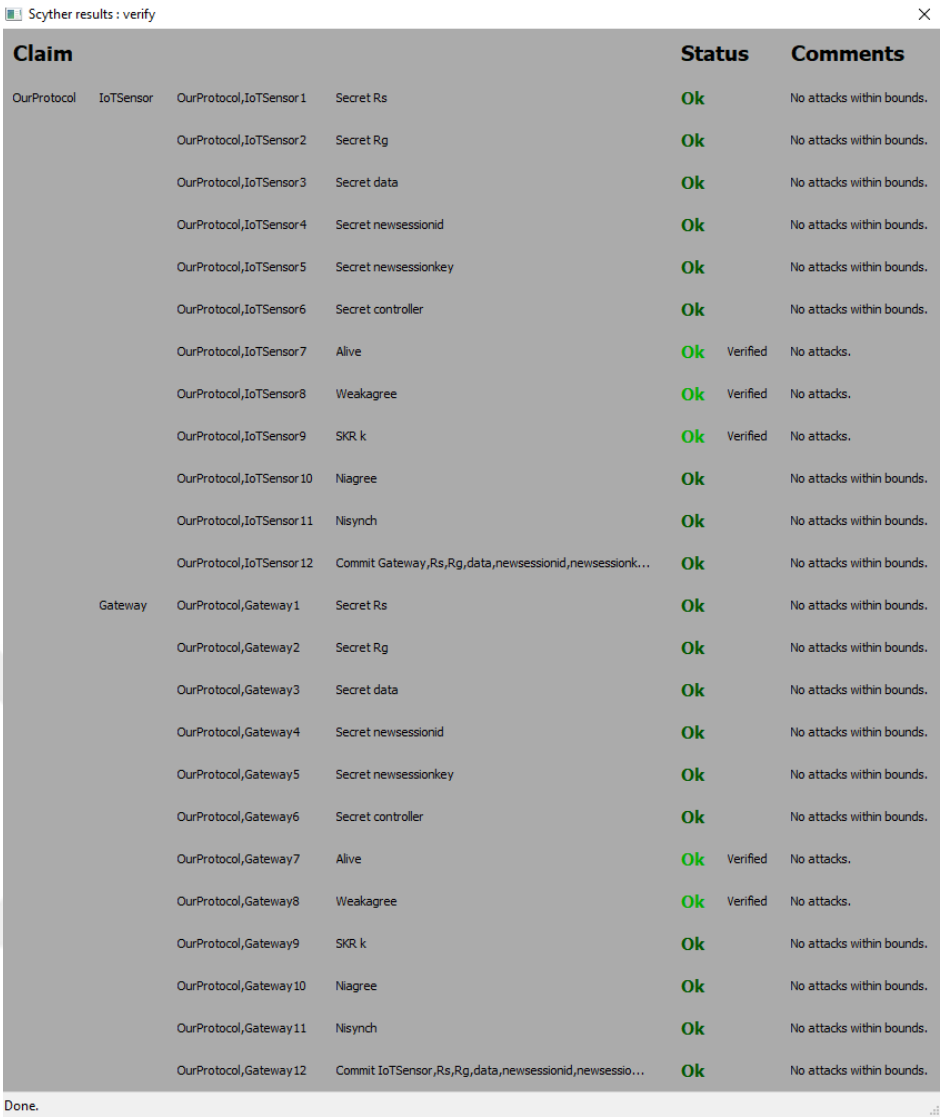

Figure 6.13 Verifying our protocol against some knowns attacks by the Scyther tool

From the verification result in Figure 6.13, it appears that the proposed protocol does not contain any security vulnerabilities within bounds verified by the Scyther tool. Our protocol has successfully passed 12 different security tests separately in IoT sensor and gateway roles. The results prove that the protocol is resistant to known attacks. As a result, the protocol is suitable for increasing security in communication between IoT devices.

CHAPTER SEVEN
CONCLUSION AND FUTURE WORKS

With its advantage of facilitating our lives, the IoT ecosystem is rapidly expanding in every area of human activity. As a result, the volume of information exchange is booming. However, information security problem arises due to the lack of specific standardization in the wireless communication of IoT devices. With limited resources, most of the flourishing IoT devices fail to provide basic information security services. Therefore, attacking the transmission between IoT devices for causing data breaches has become a popular hacker activity. Confidentiality, integrity, and availability are three basic principles of information security needed for secure IoT communication. However, security solutions in resource-stricken IoT communication must be lightweight, flexible, and convenient.

In our study, we presented a protocol in order to increase security in IoT communication of resource-stricken devices. Our proposed five-step protocol includes mutual authentication, pseudo-ID, encryption techniques for secrecy, integrity, and availability in IoT information flow. Control, authentication, and counter mechanisms are used to increase the resistance against some known IoT attacks. But, our protocol is not involved with physical tampering, jamming, or some forms of side-channel attacks. Close coupling of each step with the next one increases security. The symmetric encryption key is changed in each session. AES algorithm has been preferred for stronger security.

A detailed formal and informal security analysis demonstrated that our work resists some known attacks that other protocols have failed. Our protocol has been tested by the formal security verification tool Scyther. The details of the description of our protocol and the verification results have been presented. The security analyses have shown that our designed protocol is secure against many known attacks.

By updating future encryption standards, our protocol can be adapted to the standards and continue to be used. Future work should focus on better communication

standards in IoT devices. The new methods should aim at creating a network by reducing resource consumption and improving security.

REFERENCES

Addabbo, T., Alioto, M., Fort, A., Pasini, A., Rocchi, S., & Vignoli, V. (2007). A class of maximum-period nonlinear congruential generators derived from the rényi chaotic map. *IEEE Transactions on Circuits and Systems I: Regular Papers, 54* (4), 816-828.

Alabaa, F.A., Othmana, M., Hashema, I.A.T., & Alotaibib, F. (2017). Internet of Things security: a survey. *Journal of Network and Computer Applications, 88,* 10-28.

Al-Fuqaha A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: a survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials, 17* (4), 2347-2376.

Atlam, H.F., & Wills, G.B. (2020). IoT security, privacy, safety and ethics. *Digital Twin Technologies and Smart Cities,* 1-27.

Aydın, Ö., & Dalkılıç, G. (2016). Nesnelerin İnterneti için sözderastsal sayı üreteci: birleştirilmiş doğrusal geri beslemeli öteleyici saklayıcı. *Akıllı Teknoloji ve Akıllı Yönetimi,* 121-129.

Beşkirli, A., Özdemir, D., & Beşkirli, M. (2019). Şifreleme yöntemleri ve RSA algoritması üzerine bir inceleme. *Avrupa Bilim ve Teknoloji Dergisi, Özel Sayı 2019,* 284-291.

Çakır, H.Ş., & Özcanhan, M.H. (2011). Gömülü sistem uygulamalarına yapılan saldırılar ve ağ bağlantılı gömülü sistemlerin güvenliğinin sağlanması. *Elektrik-Elektronik ve Bilgisayar Sempozyumu 2011,* 120-124.

Chackak, E. (2018). *8 types of security threats to the IoT.* Retrieved March 20, 2021, from https://www.cyberdb.co/8-types-of-security-threats-to-the-iot.

Chaudhary, A., Peddoju, S. K., & Kadarla, K. (2017). Study of internet-of-things messaging protocols used for exchanging data with external sources. *In 2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 666-671.

Cispa. (2021). *Scyther: installation.* Retrieved March 12, 2021, from https://people.cispa.io/cas.cremers/scyther/install-generic.html

Coskun, A., & Ülker, Ü. (2013). Ulusal bilgi güvenliğine yönelik bir kriptografi algoritması geliştirilmesi ve harf frekans analizine karşı güvenirlik tespiti. *Bilişim Teknolojileri Dergisi, 6* (2), 31-39.

Cremers, C. J. F., (2006). *Scyther: semantics and verification of security protocols*. PhD Thesis, Eindhoven University, Eindhoven.

Dalal, N., Shah, J., Hisaria, K., & Jinwala, D. (2010). A comparative analysis of tools for verification of security protocols. *International Journal of Communications, Network and System Sciences 3* (10), 779-787.

Dalkılıç, G. (2014). An enhanced lightweight authentication (ELA) protocol for WISP based RFID applications. *Life Sci J 2014, 11* (10), 878-885.

Dalkılıç, H., & Özcanhan, M.H. (2016). A sample for secure sensor data collection: data secured fall prevention and fall detection sensor. *2016 24th Signal Processing and Communication Application Conference (SIU)*, 389-392.

Debian. (2021). *Debian: user manual.* Retrieved February 5, 2021, from https://www.debian.org/doc/manuals/users-guide/users-guide.en.pdf

Deogirikar, J., & Vidhate, A. (2017). Security attacks in IoT: a survey. *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC),* 32-37.

Henkoğlu, T. (2019). Bilgi saklama süreçlerinde yapay zekâ sistemlerinin kullanımına yönelik risk değerlendirmesi. *Arşiv Dünyası, 6* (2), 134-147.

Ibrahim, A., & Dalkılıç, G. (2019). Review of different classes of RFID authentication protocols. *Wireless Networks, 25* (3), 961-974.

Jallouli, O. (2017). *Chaos-based security under real-time and energy constraints for the Internet of Things.* PhD Thesis. Universite De Nantes, Nantes.

Kahya, N., Ghoualmi, N., & Lafourcade, P. (2012). Formal analysis of PKM using Scyther tool. *2012 International Conference on Information Technology and e-Services,* 1-6.

Khvoynitskaya, S. (2019). *The history and future of the Internet of Things.* Retrieved February 20, 2021, from https://www.itransition.com/blog/iot-history

Kumar, Y., Munjal, R., & Sharma, H. (2011). Comparison of symmetric and asymmetric cryptography with existing vulnerabilities and countermeasures. *IJCSMS International Journal of Computer Science and Management Studies, 11* (3), 60-63.

Li, S., Tryfonas, T., & Li, H. (2016). The Internet of Things: a security point of view. *Internet Research, 26* (2), 337-359.

Mahfouz, M.A., & Adjei-Quaye, A. (2017). Information security in an organization. *International Journal of Computer, 24* (1), 100-116.

Naik, N. (2017). Choice of effective messaging protocols for IoT systems: MQTT,

CoAP, AMQP and HTTP. *In 2017 IEEE International Systems Engineering Symposium (ISSE),* 1-7.

Navas, R.E., & Toutain, L. (2018). LATe: A lightweight authenticated time synchronization protocol for IoT. *2018 Global Internet of Things Summit (GIoTS),* 1-6.

Oracle. (2021). *VirtualBox: user manual.* Retrieved February 3, 2021, from http://download.virtualbox.org/virtualbox/UserManual.pdf

Ordu, L., & Örs S.B. (2006). Yan kanal analizi saldirilarina genel bakis. *Ulusal Elektronik İmza Sempozyumu Bildiriler Kitabı,* 242-249.

Özcanhan, M.H. (2011). *Security and reliability in embedded systems*. PhD Thesis, Dokuz Eylul University, İzmir.

Özcanhan, M.H. (2014). Improvement of a weak RFID authentication protocol making drug administration insecure. *Life Science Journal 2014, 11* (10), 269- 276.

Özcanhan, M.H. (2015). Analysis of a recent hash based RFID authentication protocol intended for telecare medicine. *International Research Journal of Engineering and Technology (IRJET), 2* (4), 1520-1524.

Özcanhan, M.H. (2015). Analysis of a recent quadratic residue based authentication protocol for low-cost RFID tags. *International Journal of Novel Research in Engineering and Science, 2* (1), 7-13.

Özcanhan, M.H., & Dalkılıç, G. (2015). Mersenne twister-based RFID authentication protocol. *Turkish Journal of Electrical Engineering & Computer Sciences, 23* (1), 231-254.

Özcanhan, M.H., Dalkılıç, G., & Utku, S. (2013). Analysis of two protocols using EPC Gen-2 tags for safe inpatient medication. *2013 IEEE INISTA*, 1-6.

Özcanhan, M.H., Dalkılıç, G., & Utku, S. (2013). Is NFC a better option instead of EPC Gen-2 in safe medication of inpatients. *International Workshop on Radio Frequency Identification: Security and Privacy Issues*, 19-33.

Özcanhan, M.H., Dalkılıç, G., & Utku, S. (2014). Cryptographically supported NFC tags in medication for better inpatient safety. *Journal of Medical System, 38* (8), 1-15.

Padate, R., & Patel, A. (2014). Encryption and decryption of text using AES algorithm. *International Journal of Emerging Technology and Advanced Engineering, 4* (5), 54-9.

Popescul, D. (2011). The confidentiality-integrity-accessibility triad into the knowledge security. a reassessment from the point of view of the knowledge contribution to innovation, *Proceedings of The 16th International Business Information Management Association Conference*, 1338-1345.

Putra, D.S.K., Sadikin M.A., & Windarta, S. (2017). S-Mbank: secure mobile banking authentication scheme using signcryption, pair based text authentication, and contactless smart card. *2017 15th International Conference on Quality in Research (QiR) International Symposium on Electrical and Computer Engineering*, 230-234.

Rughoobur, P., & Nagowah, L. (2017). A lightweight replay attack detection framework for battery depended IoT devices designed for healthcare. *International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS),* 811-817.

Scully, P. (2020). *Top 10 IoT applications in 2020.* Retrieved December 15, 2021, from https://iot-analytics.com/top-10-iot-applications-in-2020.

Scyther. (2021). *Scyther: user manual.* Retrieved March 2, 2021, from https://ics.upjs.sk/~jirasek/krp/scyther-manual.pdf

Siponen, M.T., & Oinas-Kukkonen, H. (2007). A review of information security issues and respective research contributions. *The Database for Advances in Information Systems, 38* (1), 60-81.

Sobti, R., & Geetha, G. (2012). Cryptographic hash functions: a review. *International Journal of Computer Science Issues, 9* (2), 461-479.

Srinivasan, L., Scharnagl, J., & Schilling, K. (2013). Analysis of WebSockets as the new age protocol for remote robot tele-operation. IFAC Proceedings Volumes, 46 (29), 83-88.

Suguna, S., Dhanakoti, V., & Manjupriya, R. (2016). A study on symmetric and asymmetric key encryption algorithms. *International Research Journal of Engineering and Technology, 3* (4), 27-31.

Tyson, J. (2019). *CIA model.* Retrieved April 5, 2021, from https://blog.jamestyson.co.uk/the-cia-and-dad-triads

Wallace, K., Moran, K., Novak, E., Zhou, G., & Sun, K. (2016). Toward sensor-based random number generation for mobile and IoT devices. *IEEE Internet of Things Journal, 3* (6), 1189-1201.

Yang, H., Prinz, A., & Oleshchuk, V. (2016). Formal analysis and model checking of a group authentication protocol by Scyther. *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP),* 553-557.

Yerlikaya, Ö., & Dalkılıç, G. (2018). Authentication and Authorization Mechanism on Message Queue Telemetry Transport Protocol. *2018 3rd International Conference on Computer Science and Engineering (UBMK),* 145-150.

Yıldırım, K., Dalkılıç, G., & Duru, N. (2019). Hsiang m-kupon protokolünün güvenlik analizi. *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi, 34* (4), 1705-1724.

Zhao, K., & Ge, L. (2013). A survey on the Internet of Things security. *2013 9th International Conference on Computational Intelligence and Security*, 663-667.

APPENDICES

APPENDIX A: Code Implementation of Our Protocol Using Scyther Tool

APPENDIX A.1 Description of Code Segment for IoT Sensor Side

```
usertype String;
//k: key of IoTSensor between Gateway
//send mean: send packet
//recv mean: receive packet
//nonce mean: number only used once
//fresh mean: first generated variable
/* PROTOCOL Seyhun Yasar OZKAL, Mehmeh Hilal OZCANHAN */
protocol OurProtocol(IoTSensor,Gateway)  // OurProtocol specification
{
role IoTSensor // IoTSensor role specification
 {
fresh Rs: Nonce; // Generated by IoTSensor for IoTSensor side authentication
fresh IDs: Nonce; // Generated by IoTSensor for matching relevant IoTSensor Gateway side
fresh data: String; // Generated by IoTSensor measured value
var Rg: Nonce; // Generated by Gateway for Gateway side authentication
var controller: Nonce; // Generated by Gateway for ending session controller
var newsessionid: Nonce; // Generated by Gateway for using new session id
var newsessionkey: Nonce; // Generated by Gateway for using new session key
```

APPENDIX A.1 continues

send_1(IoTSensor,Gateway,IDs,{Rs}k(IoTSensor,Gateway)); //ID send for matching relevant key, Rs send for authentication

recv_2(Gateway,IoTSensor,{Rs,Rg}k(Gateway,IoTSensor)); //Rs recv for matching and verifying first Rs, Rg recv for gateway authentication

match(Rs,{Rs}k(Gateway,IoTSensor)); //Rs verify OK, IoTSensor side authentication completed

send_3(IoTSensor,Gateway,{Rg,data}k(IoTSensor,Gateway)); //Rg send for for gateway authentication, data send for measurable value

recv_4(Gateway,IoTSensor,{data,newsessionid,newsessionkey,controller}k(Gateway,IoTSensor)); //recv for matching and verifying first data, recv for newsessionid and newsessionkey next session

match(data,{data}k(Gateway,IoTSensor)); //Data verify OK

send_5(IoTSensor,Gateway,{controller}k(IoTSensor,Gateway)); //controller is sent for receiving the data correctly by gateway

APPENDIX A.1 continues

```
    claim_IoTSensor1(IoTSensor,Secret,Rs);
    claim_IoTSensor2(IoTSensor,Secret,Rg);
    claim_IoTSensor3(IoTSensor,Secret,data);
    claim_IoTSensor4(IoTSensor,Secret,newsessionid);
    claim_IoTSensor5(IoTSensor,Secret,newsessionkey);
    claim_IoTSensor6(IoTSensor,Secret,controller);
    claim_IoTSensor7(IoTSensor,Alive);
    claim_IoTSensor8(IoTSensor,Weakagree);
    claim_IoTSensor9(IoTSensor,SKR,k);
    claim_IoTSensor10(IoTSensor,Niagree);
    claim_IoTSensor11(IoTSensor,Nisynch);
    claim_IoTSensor12(IoTSensor,Commit,Gateway,Rs,Rg,data,newsessionid,ne
wsessionkey,controller);
```

APPENDIX A.2 Description of Code Segment for Gateway Side

```
role Gateway // Gateway role specification
{
 fresh Rg: Nonce; // Generated by Gateway for Gateway side authentication
fresh controller: Nonce; // Generated by Gateway for ending session controller
fresh newsessionid: Nonce; // Generated by Gateway for using new session id
fresh newsessionkey: Nonce; // Generated by Gateway for using new session key
var Rs: Nonce; // Generated by IoTSensor for IoTSensor side authentication
var IDs: Nonce; // Generated by IoTSensor for matching relevant IoTSensor Gateway side
var data: String; // Generated by IoTSensor measured value

recv_1(IoTSensor,Gateway,IDs,{Rs}k(IoTSensor,Gateway)); //ID recv for matching relevant key, Rs recv for sensor authentication

send_2(Gateway,IoTSensor,{Rs,Rg}k(Gateway,IoTSensor)); //Rs send for matching and verifying first Rs, Rg send for gateway authentication

recv_3(IoTSensor,Gateway,{Rg,data}k(IoTSensor,Gateway)); //Rg recv for matching and verifying first Rg, data recv for measurable value

match(Rg,{Rg}k(IoTSensor,Gateway)); //Rg verify OK, Gateway side authentication completed
```

APPENDIX A.2 continues

send_4(Gateway,IoTSensor,{data,newsessionid,newsessionkey,controller}k(Gateway,IoTSensor)); //send for matching and verifying first data, send for newsessionid and newsessionkey next session

recv_5(IoTSensor,Gateway,{controller}k(IoTSensor,Gateway)); //controller recv for matching and verifying first controller and finish session

match(controller,{controller}k(IoTSensor,Gateway)); //controller verify OK, Completed communication

claim_Gateway1(Gateway,Secret,Rs);

claim_Gateway2(Gateway,Secret,Rg);

claim_Gateway3(Gateway,Secret,data);

claim_Gateway4(Gateway,Secret,newsessionid);

claim_Gateway5(Gateway,Secret,newsessionkey);

claim_Gateway6(Gateway,Secret,controller);

claim_Gateway7(Gateway,Alive);

claim_Gateway8(Gateway,Weakagree);

claim_Gateway9(Gateway,SKR,k);

claim_Gateway10(Gateway,Niagree);

claim_Gateway11(Gateway,Nisynch);

claim_Gateway12(Gateway,Commit,IoTSensor,Rs,Rg,data,newsessionid,newsessionkey,controller);