REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

NEW DATA ENCRYPTION METHOD FOR INTERNET OF THINGS

Hussein Ali Mostafa AL-ALI

Master's Thesis

Supervisor

Prof. Dr. Galip CANSEVER

İstanbul, 2024

NEW DATA ENCRYPTION METHOD FOR INTERNET OF THINGS

Hussein Ali Mostafa AL-ALI

Electrical and Computer Engineering

Master's Thesis

ALTINBAŞ UNIVERSITY

2024

The thesis titled NEW DATA ENCRYPTION METHOD FOR INTERNET OF THINGS prepared by HUSSEIN ALI MOSTAFA and submitted on 10/01/2024 has been **accepted unanimously** for the degree of Master of Science in Electrical and Computer Engineering.

Prof. Dr. Galip CANSEVER

Supervisor

Thesis Defense Committee Members:

Prof. Dr. Galip CANSEVER	Faculty of Engineering and Architecture, Altinbas University	
Asst. Prof. Dr. Sefer KURNAZ	Faculty of Engineering and Architecture, Altinbas University	
Asst. Prof. Dr. Yuksel BAL	Faculty of Computer Engineering, Istanbul Topkapi University	

I hereby declare that this thesis meets all format and submission requirements of a Master's thesis.

I hereby declare that all information/data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Hussein Ali Mustafa AL-ALI

Signature

DEDICATION

I can only express my great thanks to my dear supervisor for his great efforts and extensive knowledge, which played a major role in helping me and providing support and important scientific guidance through whole research work. I dedicate this research work and pledge it to my supervisor as well as my wife for their support of me in all the difficult times until the completion of the thesis.

ACKNOWLEDGEMENTS

Firstly and mainly, I want to express my gratitude to Prof. Dr. Galip CANSEVER who served as my dissertation advisor, for his guidance and for supporting me as I wrote this dissertation. I found it really helpful to grasp the reasoning behind the research to regularly discuss my progress, issues, and suggestions with my advisor, Prof. Dr. Galip CANSEVER. It helped me understand the research's technical requirements better.

ABSTRACT

NEW DATA ENCRYPTION METHODE FOR INTERNET OF THINGS

AL-ALI, Hussein Ali Mostafa

M.Sc., Electrical and Computer Engineering, Altınbaş University,

Supervisor : Prof. Dr. Galip CANSEVER

Date: 01 / 2024

Pages: 92

Cyberspace is a complex environment consisting of heterogeneous technologies (i.e., Internet of Things, Fog Computing and Cloud Computing and so forth) resulting from interacting services, software and people on the Internet. It allows users to interact, share information, swap ideas, engage in social or discussion forums, play games, and conduct business, among many other activities. The biggest challenges facing cyberspace today are Cyber attacks, which affect security and integrity services. However, many traditional security mechanisms provide protection and security services to solve these issues. Therefore, many researchers have been focused on solving security and integrity issues by growing the need for effective lightweight encryption techniques that incorporate both lightweight symmetric and asymmetrical algorithms' advantages.

In this thesis, we will implement and design a lightweight encryption method which has the following characteristics (Key less, Encryption & Integrity, Text & Number End-to-End Encryption, Reduce Traffic and processing overhead). In addition, the proposed system provide the data integrity via applying HASH 256 function to generate HASH value.

The proposed lightweight encryption algorithm focuses on the optimal use of the resources of Internet of Things devices, so that it greatly saves all of (Processor, Memory, Energy, Time, and Bandwidth (no need to distribute keys)) on the other hand, giving high security, especially against the crypto analyser. In addition The proposed lightweight encryption

algorithm has the ability to manipulated both text and numbers for English and Arabic languages.

Also, to achieving data integrity in proposed system within the Internet of Things environment, 4hexa decimal digit from HASH value were used instead of original 64 hexa decimal digit HASH value to lower the bandwidth of the network, processing, and storage.

Keywords: Data Integrity, Internet of Things, Lightweight Encryption Algorithm, Mosquitte Broker, MQTT Protocol, Raspberry Pi.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

LIST OF CHARTS

Pages

ABBREVIATIONS

ABE	**:**	Attribute Based Encryption
AE	**:**	Authenticated Encryption
AES – GCM	**:**	Advanced Encryption Standard – Galois Counter Mode
AI	**:**	Artificial Intelligence
AMQP	**:**	Advanced Message Queuing Protocol
AWGN	**:**	Additive White Gaussian Noise
BFA	**:**	Brute Force Attack
CB	**:**	Continuation Bit
CCA	**:**	Chosen Ciphertext Attack
CoAP	**:**	Constrained Application Protocol
CPA	**:**	Chosen Plaintext Attack
CTR	**:**	Confidential Treatment Request
D - AES	**:**	Dynamic - Advanced Encryption Standard
DES	**:**	Data Encryption Standard
DH	**:**	Diffie-Hellman
DoS	**:**	Denial of Service
DTLS	**:**	Datagram Transport Layer Security
ECC	**:**	Elliptic Curve Cryptography
ECDHD	**:**	Elliptic Curve Diffie-Hellman Decision
EDL	**:**	Enhanced Driver's License
EPL	**:**	Eclipse Public License
HTTP	:	Hyper Text Transfer Protocol
IANA	**:**	Internet Assigned Numbers Authority

IBM	**:**	International Business Machines
IEC	**:**	International Electrotechnical Commission
IETF	**:**	Internet Engineering Task Force
IoT	**:**	Internet of Things
IPL	**:**	Identity Path and Location
IPS	**:**	Internet Protocol Security
ISO	**:**	International Standardization Organization
KP – ABE	**:**	Key Policy - Attribute Based Encryption
LoRaWAN	**:**	Long Range Wide Area Network
LSID-IoT	**:**	Lightweight Security and Integrity Data – Internet of Thing
M2M	**:**	Machine-to-Machine
MCC	**:**	Mobile Cloud Computing
MD 5	**:**	Message Digest 5
MQTT	**:**	Message Queue Telemetry Transport
OASIS	**:**	Organization for the Advancement of Structured Information Standards
QoS	**:**	Quality of Service
RFID	**:**	Radio Frequency Identification Devices
RL	**:**	Remaining Length
RSA	**:**	Rivest Shamir Adleman
RSS	**:**	Robust Security Scheme
SHA	**:**	Secure Hash Algorithm
SHS	**:**	Secure Hash Standard
SIT	**:**	Secure Internet of Things

SSH	**:**	Secure SHell
SSL	**:**	Secure Sockets Layer
TCP / IP	**:**	Transmission Control Protocol / Internet Protocol
TLS	**:**	Transport Layer Security
UDP	**:**	User Datagram Protocol
VNC	**:**	Virtual Network Computing
VoIP	**:**	Voice over Internet Protocol
WSN	**:**	Wireless Sensor Networks
XMPP	**:**	Extensible Messaging and Presence Protocol

1. INTRODUCTION

Internet of Things (IoT) become more effective and seamless in smart cities with features like smart lighting, smart traffic, smart parking and congestion management, etc. In order to protect human life and resources, a variety of environmental monitoring systems can be utilised to track air pollution, identify forest fires, detect earthquakes early, etc. In order to comprehend water demand and keep track of water quality, intelligence can also be applied to the water supply[1].

Additionally, information on the soil's pH, temperature, and moisture content will be helpful in boosting crop output. Smart irrigation can also be connected to climate and weather forecasts. Figure 1.1 shows the architecture of the I0T that includes the connections between various IoT-based applications with specific use cases and samples[1].

Figure 1.1: Internet of Things (IoT) Architecture[1].

1.1 A BRIEF HISTORY OF IOT

Cyberspace, an electronic and communication device-based virtual and dynamic environment, stores and uses electronic data[2]. The primary purpose of making Cyberspace is to communicate and share information across the Internet. However, it has a significant impact on human daily activities. Yahoo, Google, and Facebook are some examples of

Cyberspace. In addition, various emerging technologies are involved in a cyberspace communication network which includes: Artificial Intelligence (AI)[3][4] Cloud Computing (CC), Blockchain Technology, Mobile Cloud Computing (MCC). Big Data Analytics, Big Data, Internet of Things (IoT) [5][6][7][8][9] and so forth as shown in Figure 1.2.

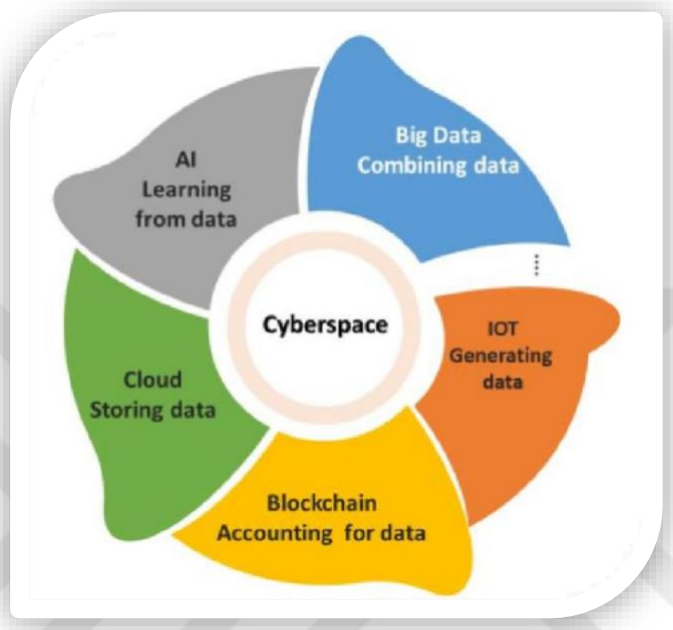


Figure 1.2: Some of the Cyberspace Emerging Technologies[9].

The Internet of Things (IoT) environment has a wide variety of different devices (objects or things) connected using either wired or wireless communication technologies allows them to collect, exchange, and process data occasionally. These devices / things range from smart devices for household like smart bulbs, temperature sensors, and IP cameras to more advanced devices like accelerometers, Radio Frequency Identification Devices (RFID), heartbeat detectors, tablets, iPads, and other sensors in automobiles[10]. Furthermore, as IoT networks spread globally, more devices will eventually be connected, predicted to reach 75 billion by 2025[11], as indicated in Figure 1.3.

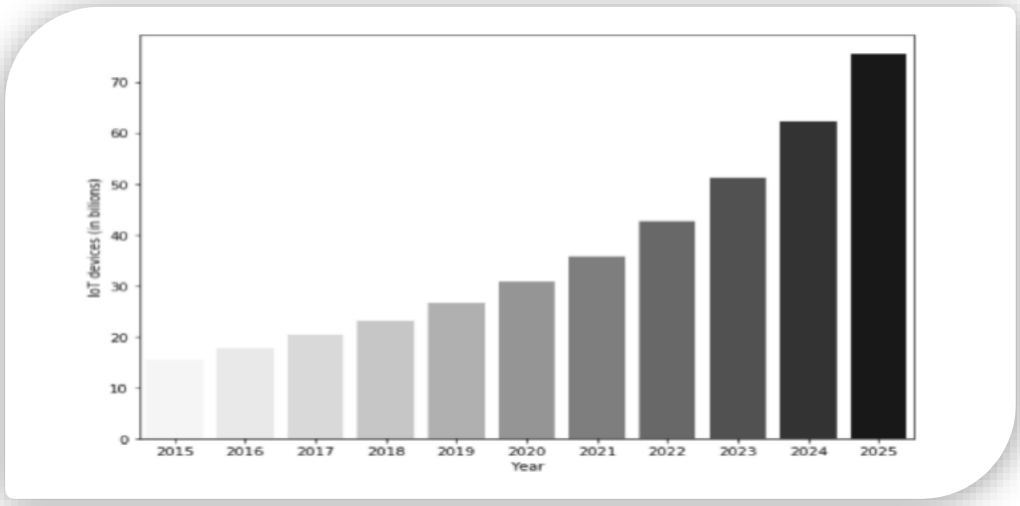


Figure 1.3: From 2015 to 2025, the Expected Number of IoT Devices Worldwide [11].

These devices manage a wide range of tasks in the smart home, companies, smart cities, or government institutions through the Internet, as shown in Figure 1.4.

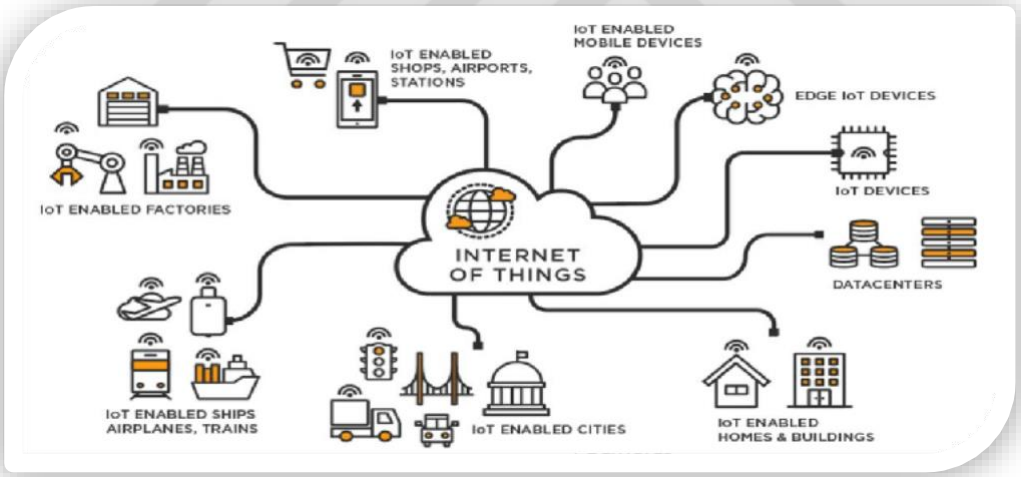


Figure 1.4: Internet of Things Environment[9].

IoT devices interact with their actuators without human interaction, known as Machine-to-Machine (M2M) interaction[12]. Data exchange between network nodes, such as actuators, services, sensors, or other platforms, is carried out via message brokers, a key component of middleware systems. The broker may be operating locally (on edge) or remotely (in the cloud)[13]. The publish/subscribe pattern is a communication architecture where sensors and

actuators function as publishers and subscribers. Applications for users can serve as either publishers or subscribers based on each application's design[14][15].

The main characteristics of IoT networks are heterogeneity, massive deployment, communication in close proximity, low-cost and low-power communication, dynamic changes in the network, low latency and Ultra reliable communication, safety, and intelligence. Also, IoT devices have certain features, such as smaller sizes and lower computation capacity. Moreover, they use particular lightweight communication protocols[16].

On the other hand, the IoT has many restrictions regarding devices and components constrained by processor speed, memory size, communication bandwidth, power consumption, and its heterogeneous nature and widespread use[17].

Traditional protocols (TCP & UDP) do not work well with resource constrained devices. Thus, numerous communication protocols and messaging have been designed to enable the reliable and secure transmission of data across IoT networks such as (HTTP, LoRaWAN, Bluetooth, ZigBee, MQTT, CoAP, AMQP, M2M, XMPP) as shown in Figure 1.5[18].

As indicated in Figure 1.5, The Best of them is MQTT Protocol in the IoT, For example, MQTT is frequently used in smart houses. [19], agricultural IoT[20], and industrial applications[21]. Because it has lower requirements for memory and power, less packet loss, and supports communication at lower bandwidths[7][18].

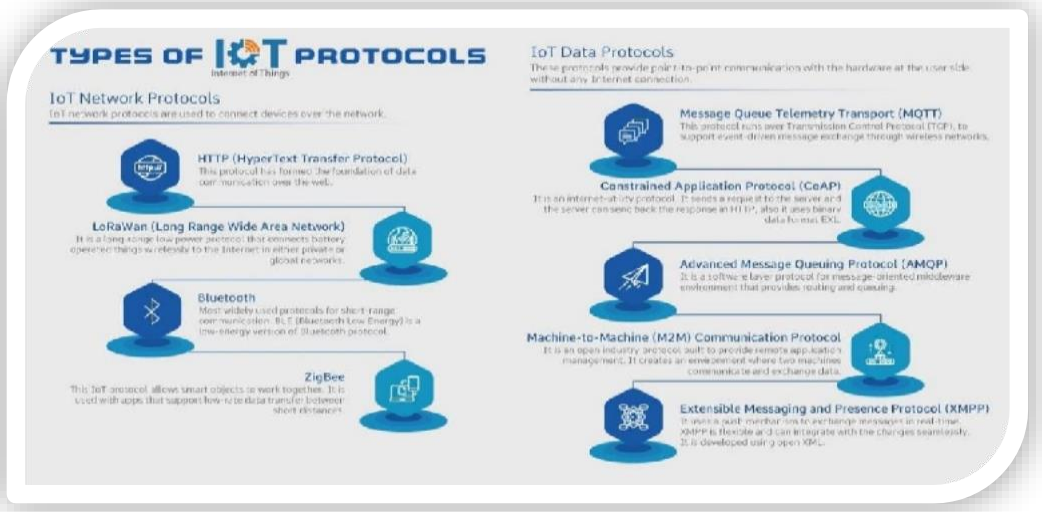


Figure 1.5: Types of IOT Protocols.

Nevertheless, the MQTT's lack of security features is by far its biggest flaw. The devices' processing overhead has increased as a result of the current methodologies, which are also still open to several attacks[22].

IoT technology is being applied in more and more fields. Therefore, they are more vulnerable to security and privacy attacks for the following reasons: increasing popularity; they are connected to the Internet; they do not have a standard; they can be controlled remotely; they are provided with lower security policies, and are consume sensitive data. So they continue to be practical and affordable[23].

Since most traditional encryption methods and systems were designed for resource-rich devices on high-bandwidth networks, they are not appropriate for use in Internet of Things networks. In order to address this issue and provide security and privacy for an Internet of Things environment, lightweight encryption algorithms have been suggested. These algorithms realize Lightweight, Keyless (symmetric key), Robust, End-to-End encryption while also consuming fewer resources [24].

1.2 PROBLEM STATEMENT

The greatest challenge of implementing a complete paradigm IoT is the privacy and security issues. IoT devices are restricted in terms of resources, such as limited to a few megahertz of computational power, 5–10 Megabytes of memory, and a small amount of persistent storage available. In addition, they are battery-powered; their power capacity spans from 6 to 24 hours. These limitations prevent IoT devices from implementing resource-intensive security measures like security information, event management and energy requirements, big data, heterogeneity, scalability etc.

Moreover, IoT devices' growth (scale) is another critical challenge. By 2025, it's expected that the number of Internet-connected gadgets will have nearly tripled. The development of IoT by increasing the number of devices makes it a fragile environment that attracts a wide range of attacks.

Many environmental monitoring applications can be used to track air pollution, forest fire detection, early earthquake detection, and other environmental hazards in order to protect

human life and resources, etc. All of these benefits are subject to limitations and problems, including privacy and security.

One of the biggest issues that networks have faced throughout the years are security, privacy, complexity, big data, Energy Consumption and Devices/Links Heterogeneity Challenges As sh0wn in Figure 1.6 :

Figure 1.6: IoT Challenges.

As a Result, trust, Privacy and Security are essential components for IoT applications. The secrecy and integrity of the data should be maintained when data packets are sent over numerous networks and devices to their final internet recipient. Additionally, because the majority of IoT devices are low power devices, it is not possible to immediately apply currently proven cryptographic methods in the IoT environment.

To do this, IoT technology has made great progress for a variety of applications that use several enabling and development technologies and must have many features: lightweight, resilience to changes in network topology, low bandwidth, and simple.

1.3 THE WORK CHALLENGES

Many challenges have faced the implementation of the proposed Lightweight Security and Integrity Data – Internet of Things (LSID-IoT), as mentioned below:

a. Creating the environment of LSID-IoT in Raspberry PI 4B.
b. Finding the libraries for manipulate MQTT messages in python compatible with Raspberry environment.
c. Configuration of the mosquitte Broker.
d. Encryption and decryption number.
e. Send and received the MQTT messages.
f. Extract the data from payload MQTT message.
g. Difficulty to find a strategy to develop the lightweight encryption algorithm.
h. Create GUI interface for publishing and subscribing Applications.

1.4 AIM OF CONTRIBUTION

The main objective of this thesis is to develop algorithms for encrypting texts and numbers, in addition to achieving data integrity, working within the Internet of Things environment using the MQTT protocol via the Raspberry Pi device. These algorithms have the following characteristics (Lightweight, Keyless (Symmetric key), Robust, and End-to-End encryption).

On the other hand, the contributions of this thesis include:

a. Review research and scientific thesis on lightweight encryption algorithms within the Internet of Things environment.
b. Analyzing the MQTT protocol, commonly used with smart devices, all the messages are found, and all their features are extracted from the MQTT session using Wireshark tools.
c. Designing and implementing a new Lightweight – encryption algorithms for both text and numbers work in the IoT environment.

d. Implementing the Tenser-Flow (64-bit) environment on Linux OS (32-bit) of the Raspberry PI 4B.

1.5 THESIS ORGANIZATION

The rest of this thesis is organized as shown in the block diagram in Figure 1.7.

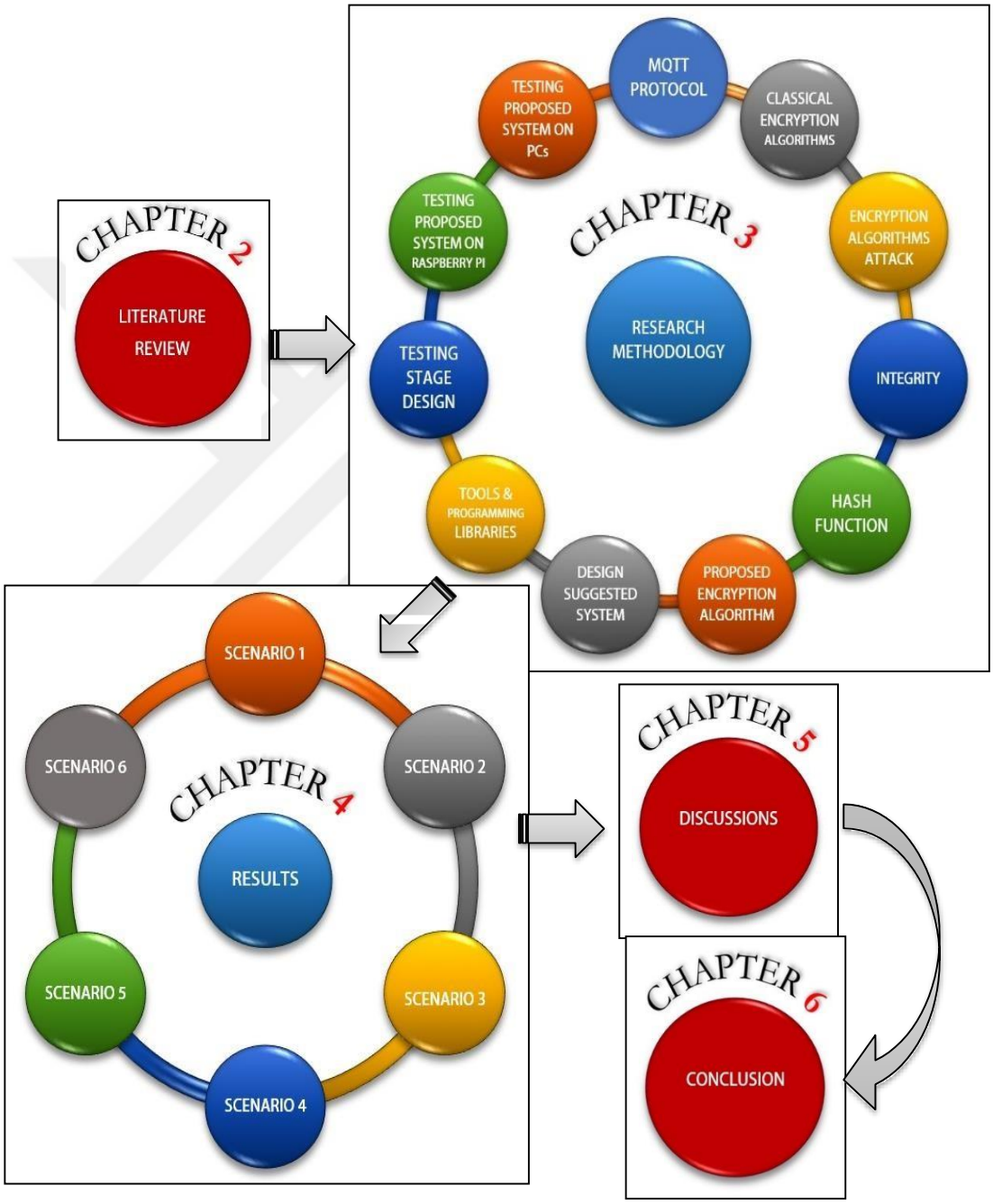


Figure 1.7: Thesis Outline.

2. LITERATURE REVIEW

Many studies have recently been conducted to provide encryption algorithms and techniques appropriate for the IoT context. They deal with the inherent resource constraints of IoT devices that make it challenging to apply common encryption algorithms, which need a lot of resources to operate. In order to provide secure communication and efficient while utilizing resources sensibly, several researchers have been proposed lightweight encryption techniques to address the resource limitation challenge and enhance currently used traditional encryption techniques for IoT devices.

The researchers in [25] proposed scheme to address privacy and security issues in the Internet of Things, a lightweight, pairless ABE scheme based on Elliptic Curve Cryptography (ECC) was developed. The safety of the proposed scheme is demonstrated using the trait-based selective group model and is based on the Elliptic Curve Diffie-Hellman Decision (ECDHD) assumption rather than the bilinear Diffie-Hellman (DH) assumption. Comparison analyzes with existing ABE schemas are performed in detail by setting metrics to measure communication overhead, standardization, and computational overhead[25]. The results show that the implementation efficiency has improved, in addition to that the communication costs have decreased for the proposed system.

The researchers in [26] proposed algorithm to Secure IoT (SIT) is a name for a lightweight encryption algorithm. It is a (64) bit block cipher and encrypts data with a (64) bit key. The algorithm architecture is a combination of a unified switching network with a feistel network. The result of the simulations shows that the algorithm with only five encryption rounds provides great security. The algorithm is a good candidate to be used in IoT applications because the implementation yields encouraging results [26].

The researchers in [27] describes a design for an IoT security application-friendly AES-GCM Authenticated Encryption (AE) Crypto-Core. The AES-GCM core offers confidentiality using the AES block cipher's Counter (CTR) mode, as well as authenticity and integrity using GHASH. Two key lengths are supported by AES encryption: 128-bits and 256-bits. The FPGA implementation served to validate the AES-GCM core[27].

The researchers in [28] developed and tested a new security method for Voice over Internet Protocol wireless networks VoIP. To satisfy Quality of Service (QoS) criteria for voice communication and to work with wireless devices, new encryption methods have been developed. Its goal is to reduce the power consumption and execution time of the encryption process, while increasing or maintaining the level of security[28]. This algorithm is made more difficult to hack and more secure with the use of the new technology triple key[28].

The researchers in [29] present a sophisticated method that secures data using cryptographic methods such as RSA and most importantly, it specifies a random path to route messages from source to destination. This method also ensures secure delivery of packets even when attacking intermediate nodes by defining alternative paths between the destination and the source [29]. The proposed system performs the following tasks [29]:

a. It is proposed to use a new Identity, Path, and Location (IPL) privacy algorithm to ensure that the identities and locations of source nodes are not disclosed.
b. It also ensures that packets reach their intended location by using only reliable middle nodes.
c. The current intermediate node is kept anonymous
d. For data privacy, a new mechanism is proposed that provides payload encryption and decryption.

Not realizing the intermediate nodes of the path they took previously.
They have created two important algorithms that ensure the integrity and security of the data transmitted through the sensor network.

The researchers in [24] presented a Elastic Encryption Technology (FlexenTech), a new, scalable encryption method, is used to protect IoT data while it is stored and transmitted. FlexenTech is suitable for networks and devices with limited resources. It protects against replay attacks while providing high-speed encryption, in addition to specifying a reliable mode, as it accepts any number of cycles and any number of key sizes. It provides fast encryption, protects against common attacks such as replay attacks, and defines a configurable mode that allows any number of key sizes or rounds to be used . It is the multiple levels of confidentiality that show the strength of FlexenTech during the experimental analysis by allowing different protective methods.

The researchers in [30] suggest scheme for transferring text files between secure, a lightweight, embedded IoT devices, and efficient symmetric encryption scheme is introduced along with lightweight encryption techniques for IoT applications. They considered the performance of the lightweight coding algorithm under study in terms of bit error probability and throughput using an Additive White Gaussian Noise (AWGN) channel with Rayleigh vanishing and assuming that the signal experiences a certain bit error probability in the channel. For these performance measures, they developed analytic expressions that take into account modulated signals with and without coding. In addition, they proposed to develop a lightweight encryption algorithm under investigation by increasing the security level without significant change in processing speed and increased size. The experimental data under study shows that the lightweight encryption algorithm outperforms both proposed and literature lightweight encryption algorithms in terms of error and throughput performance as well as outperforms traditional encryption algorithms (such as AES).

The researchers in [31] presents an advanced data encryption method suitable for Internet of Things (IoT) applications. The Catalan object, which acts as a cipher key in the cipher system, is used to provide ciphers based on combinatorial structures with non-overlapping or non-intersecting matching. In the experimental section, the proposed encryption method was compared with the Data Encryption Standard (DES) algorithm and Catalan numbers. This analysis is performed using the machine learning-based definition of the cipher method using only ciphertext [31], as shown in Figure 2.1.

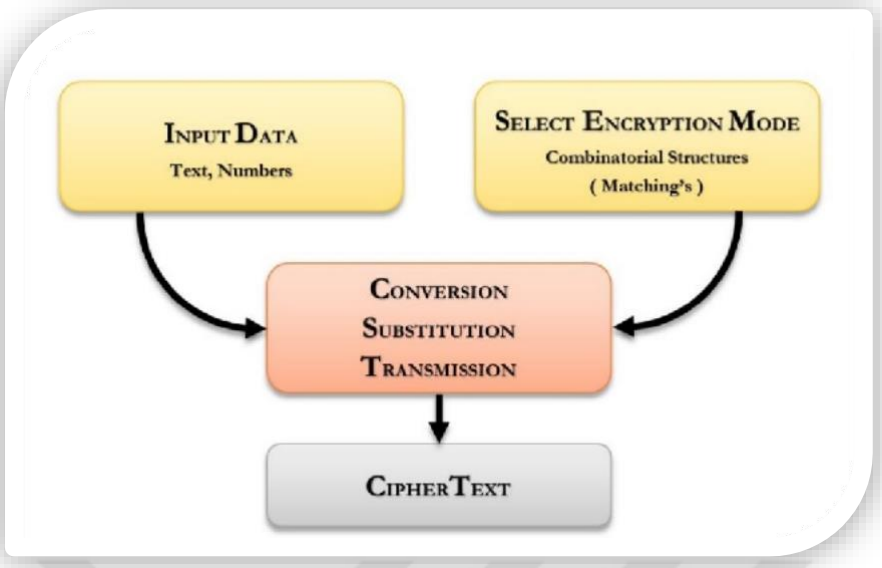


Figure 2.1: Proposed Model Framework[31].

The researchers in [22] presented an integrated scheme known as Strong Security Scheme (RSS) to defend MQTT against any vulnerabilities that could lead to sophisticated cyberattacks. Two encoding schemes are used by the proposed RSS:

a. D-AES, or a Dynamic version of the Advanced Encryption Standard .
b. KP-ABE, or Key Policy Attribute Based Encryption.

To encrypt the MQTT payload, RSS introduces D-AES, which is a new design architecture for the symmetric AES algorithm. In addition, KP-ABE, the second component of the proposed hybrid cipher system, is used for the proposed private key cipher of D-AES to avoid the computational burden of bilinear maps. Processing time and traffic are used to measure the performance of the proposed RSS. The experimental results show that the proposed D-AES algorithm is more promising and has advantages over the traditional AES algorithm.

Table 2.1: Summary of the Related Works.

No	Authors/ Year	Algorithms Used	Protocol	Metrics	Environments	Description
1	X. Yao, Z. Chen, and Y. Tian, **2015** Article [25]	(ABE) Attribute Encryption on Elliptic Cryptograph **Asymmetric**	*	• Communication overhead • Standardization computational overhead	*	• To address privacy and security issues in the IoT. • A lightweight, pair less ABE scheme based on Elliptic Curve Cryptography (ECC) was developed
2	U. A. S. Muhammad Usman, Irfan Ahmed, M. Imran Aslam, Shujaat Khan, **2017** Article [26]	Secure IoT (SIT) **Symmetric**	*	*	Micro Controller	• Image Encryption. • a lightweight encryption algorithm. • It is a 64-bit block cipher and encrypts data with a 64-bit key. • The algorithm is a good candidate to be used in IoT applications

Table 2.1: Summary of the Related Works "Table Continued".

3	**B.-Y. Sung, K.-B. Kim, and K.-W. Shin, 2018 Article [27]**	**AES & GHASH Symmetric**	*****	*****	• **Implemented in FPGA Parallel processing**	• **a design for an IoT security application-friendly AES-GCM Authenticated Encryption (AE) crypto-core.** • **provides confidentiality and integrity and authenticity by GHASH.** • **Two key lengths are supported by AES encryption: 128-bits and 256-bits.**

Table 2.1: Summary of the Related Works "Table Continued".

4	**F. Hazzaa, 2019 phD. Thises [28]**	**DES & AES & 3DES & Blow Fish & RC2 & RC6 Symmetric**	**VoIP**	*****	**Wireless Networks**	• **A new voice-over wireless network security technology.** • **In order to meet Quality of Service (QoS) standards for voice traffic, new encryption algorithms have been created.** • **Triple key used (3DES).** • **Goals :** - **Decrease the encryption process's execution time and power consumption .** - **Increasing or maintaining the level of security**

Table 2.1: Summary of the Related Works "Table Continued".

5	**T. Z. Ananth Vishnu Bhaskar, Ankit Baingane, Ryan Jahnige, Qingquan Zhang, 2020 Article [29]**	**RSA Asymmetric**	**Routing Protocols**	*****	**Wireless Networks**	• **Privacy algorithms.** • **Secure path or trusted intermediate nodes**
6	S. Medileh et al., **2020** Article [24]	(Flexen Tech) Flexible encryption Technique	Any protocols and devices	• Execution time.	• Raspberry Pi3.	• Elastic Encryption Technology (Flexen Tech) a new scalable encryption method. • is used to protect IoT data while it is stored and transmitted. • Accepts any number of key sizes and any number of cycles. • is suitable for networks and devices with limited resources. • multiple levels of confidentiality

Table 2.1: Summary of the Related Works "Table Continued".

7	**Y. M. Khattabi, M. M. Matalgah, and M. M. Olama, 2020 Article [30]**	**Enhance AES**	*	• **Processing time.**	*	• **Text File Encryption**
8	M. H. Saracevic et al., **2021** Article [31]	DES Symmetric	*	*	*	• An advanced data encryption method suitable for Internet of Things (IoT) applications". • Text & Image Encryption based on Catalan object.

Table 2.1: Summary of the Related Works "Table Continued".

9	**A. J. Hintaw, S. Manickam , S. Karuppayah, M. A. Aladaileh, M. F. Aboalmaaly, and S . U. A. Laghari, 2023 Article [22]**	**(RSS) Robust Security Scheme Symmetric**	**MQTT**	• **Processing time.** • **Traffic overhead.**	*	• **An integrated scheme known as Robust Security Scheme (RSS) to defend MQTT against any vulnerabilities that could lead to sophisticated cyber attacks.** • **Two encoding schemes are used by the proposed RSS:** 1. **D-AES, or a dynamic version of the Advanced Encryption Standard.** 2. **KP-ABE, or Key Policy Attribute Based Encryption.**

Table 2.1: Summary of the Related Works "Table Continued".

10	**Proposed system**	**LSID-IoT Symmetric**	**MQTT**	• **Processing time**	**Raspberry Pi 4B**	• **Key less.** • **Encryption & Integrity.** • **Text & Number End-to-End Encryption.** • **Reduce Traffic and processing overhead.**

Note: * Denoted that the authors have not determine the data field.

After reviewing before mentioned literature review, we noticed that most encryption algorithms use keys that need protection, management, and distribution these keys, whether they are of the Symmetric or Asymmetric keys, which require high processing that consumes energy in IOT devices. In addition, these algorithms is not verifying the integrity of the text data, furthermore most of these algorithms work to encrypt text only, and some of these algorithms breakable by the crypto analyser. Therefore, in this thesis, a lightweight encryption algorithm was developed that works with an implicit key (Symmetric key) to encrypt numbers and letters, as it depends on the location of the sequence of numbers and letters in the plain text while achieving data integrity. Thus, we achieved less processing time with no need for operations includes protecting, managing, and distributing keys, which provides us with high security while saving energy for IOT devices. Moreover, achieving a robust against to the crypto analyser.

3. RESEARCH METHODOLOGY

3.1 MQTT PROTOCOL

MQTT is a Client-Server, lightweight publish/subscribe type message transport protocol, openly and standardized by Organization for the Advancement of Structured Information Standards (OASIS) and the International Organization for Standardization (ISO) (ISO/IEC PRF 20922)[9]. Dr. Andy Stanford- Clark of IBM and Arlen Nipper of Arcom invented it in 1999. The MQTT Version 3.1, presented by International Business Machines Corporation (IBM), was made accessible without charge in 2010, and version 5 was last released in 2019[32]. Since then, various companies have adapted the MQTT protocol to their needs[33]. MQTT protocol has a very low power requirement and can send data via networks with low bandwidth or reliability[34]. MQTT is the best communication protocol for constrained situations because of its lightweight, open, and simple implementation features. The protocol works with TCP/IP or other network protocols that offer arranged, lossless, bidirectional connections. SSL/TLS connections use the registered port numbers 8883 and 1883 from the Internet Assigned Numbers Authority (IANA)[32]. Facebook Messenger is one of the most well-known MQTT-using programs. The most prevalent issues developers have with bandwidth and battery life might be solved via the MQTT protocol[35].

3.1.1 Mqtt Protocol Model

The challenge of critical systems, such as industrial process control, home automation, healthcare, smart grid, environmental monitoring, and ambient supported living, is collecting information when an event of interest will occur. Thus, a device may react directly based on the obtained information and the modern IoT applications that require intervention. The publish/subscribe model considers one of the most efficient paradigms that meets these requirements[33]. The main elements of the MQTT protocol model are Clients(the IoT nodes), Servers (=brokers), sessions, Subscriptions, and Topics, as shown in Figure 3.1[32][36].

Message: the units of data exchange between clients and topics carried across the network through the MQTT protocol for the application. Every MQTT application message contains a topic name, a QoS, a set of attributes, and payload data[37][38].

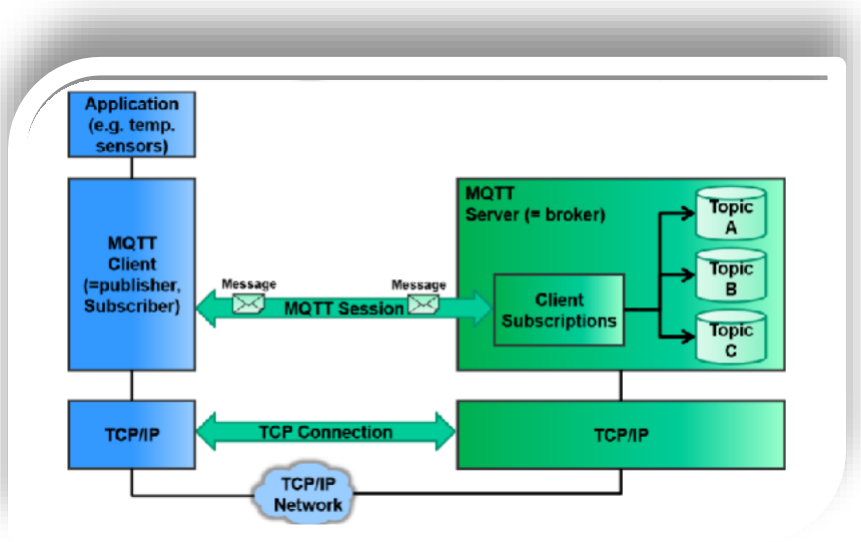


Figure 3.1: The Core Elements of the MQTT Model[36].

3.1.2 Mqtt Client (Publisher / Subscriber)

For publishing and receiving messages, clients subscribe to specific topics. Thus, a client's special roles as a subscriber and publisher are[39]:

a. Open a network connection (Session) to the server (Broker).
b. Create messages that will be published.
c. The application Publish the messages to the specific Topic in the Broker.
d. The application Subscribes to request messages from the specific Topic in the Broker.
e. The application Unsubscribe eliminates a request to receive messages from a specific Topic.
f. Close the network connection (Session) to the server (Broker).

3.1.3 Topic

Topics are essentially message queues. For clients by using topics, the publish/subscribe pattern is supported, as shown in Figure 3.2. Logically, topics enable clients to communicate using predefined semantics. The forward slash (/) delimiter is used to organize these topics, and messages are made up of data collected by IoT sensors. Each node has three primary tasks when using the MQTT protocol: topic selection, subscription, and publication[36].

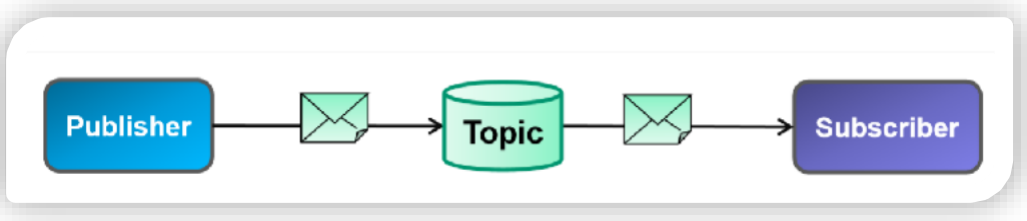


Figure 3.2: The Publish/Subscribe Pattern[9].

3.1.4 Session

A session is defined as a client's attachment to a server (which may be temporary). Sessions are the only means of communication between the client and the server.

3.1.5 Subscription

A client is logically connected to a topic by a subscription. When a client subscribes to a topic, it can receive a message from that topic. Depending on the CONNECT message's clean session flag, subscriptions may be "Transient" or "Durable," As shown in Figure 3.3.

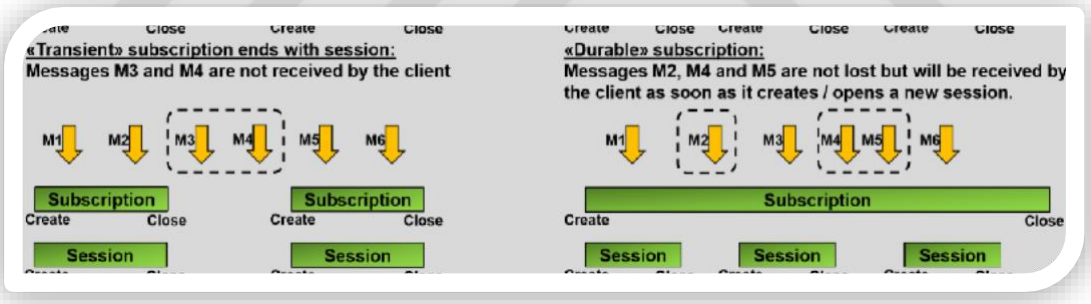


Figure 3.3: Transient / Durable Subscription[9].

This paradigm enables information delivery to subscribers without requiring them to converse actively with publishers. Additionally, this model does not require publishers and subscribers to be acquainted. Further, because this interaction can occur asynchronously, subscribers and publishers can publish or receive event information at different times. This feature increases the scalability and flexibility of the Pub/Sub model and offers a more dynamic network topology. Because of this, it is ideally suited for the Internet of Things[40].

3.1.6 Mqtt Message Format

MQTT messages come with a required fixed-length header (2 bytes) and an optional message-specific variable-length header, and a message payload . Usually, optional fields make processing a protocol more complicated. However, optional fields are utilized to minimize data transmissions because MQTT is optimized for networks with limited bandwidth and reliability (usually wireless networks). MQTT utilizes the network byte and bit ordering, as shown in Figure 3.4 and Table 3.1 describes the overview fixed header fields of the MQTT message.

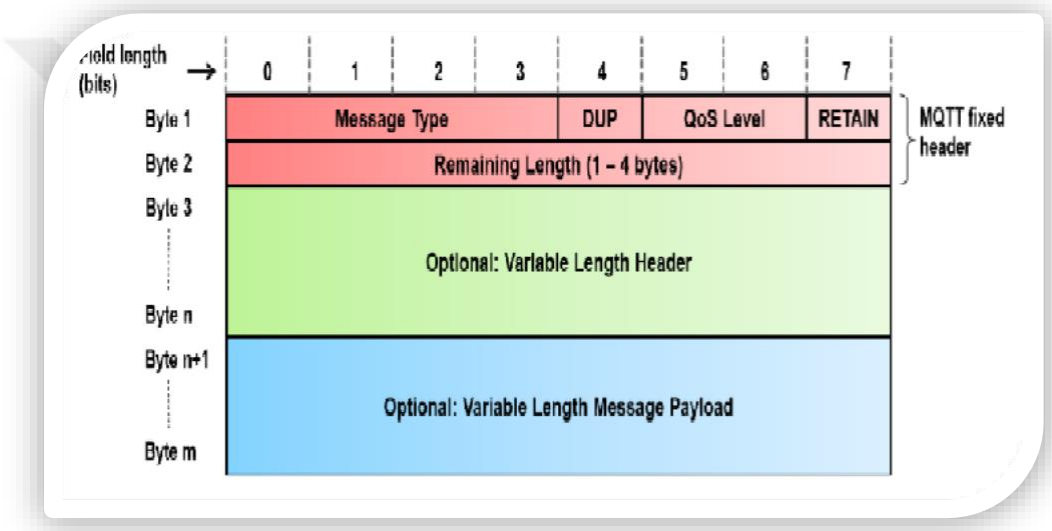


Figure 3.4: The General MQTT Message Format[9].

Table 3.1: Overview of Fixed Header Fields.

Header Field	Values / Description	
Message Type	0 ≡ Reserved	8 ≡ SUBSCRIBE
	1 ≡ C0NNECT	9 ≡ SUBACK
	2 ≡ C0NNACK	10 ≡ UNSUBSCRIBE
	3 ≡ PUBLISH	11 ≡ UNSUBACK
	4 ≡ PUBACK	12 ≡ PINGREQ
	5 ≡ PUBREC	13 ≡ PINGRESP
	6 ≡ PUBREL	14 ≡ DISC0NNECT
	7 ≡ PUBC0MP	15 ≡ Reserved
DUP	**DUP** ≡ Duplicate message flag .This flag gives the recipient a hint that the message might have been received already. When **DUP = 1**: server or client re-delivers a SUBSCRIBE, UNSUBSCRIBE, PUBLISH, or PUBREL message (Duplicate Message).	
Q0S Level	Sh0ws the message's level 0f delivery certainty when it is published. **0** ≡ At-m0st-0nce delivery (n0 guarantees). **1**≡ At-least-0nce delivery (ackn0wledged delivery). **2** ≡ Exactly-0nce delivery.	
RETAIN **(keep the last message)**	1: Instructs the server t0 retain the last received PUBLISH message and deliver it as the first message t0 new subscripti0ns. The server (Broker) transmits the retained message to the new client when subscribing to the topic.	
RL **(Remaining Length)**	The message's remaining bytes, or the optional variable-length header and payload size, are indicated.	

a. Message Type

The control field is the first byte of the two-byte fixed header. The message type field is the first 4 MSB bits from the control field. The MQTT message formats are shown in Appendix (A) according to the message type field.

b. Retain field

The retained message should be sent when a new client subscribes to the topic , according to the setting in this field. For example, the temperature data topic sends subscribers the most recent temperature reading. According to Figure 3.5, RETAIN=1 informs subscriber B that the message might have been published in the past.

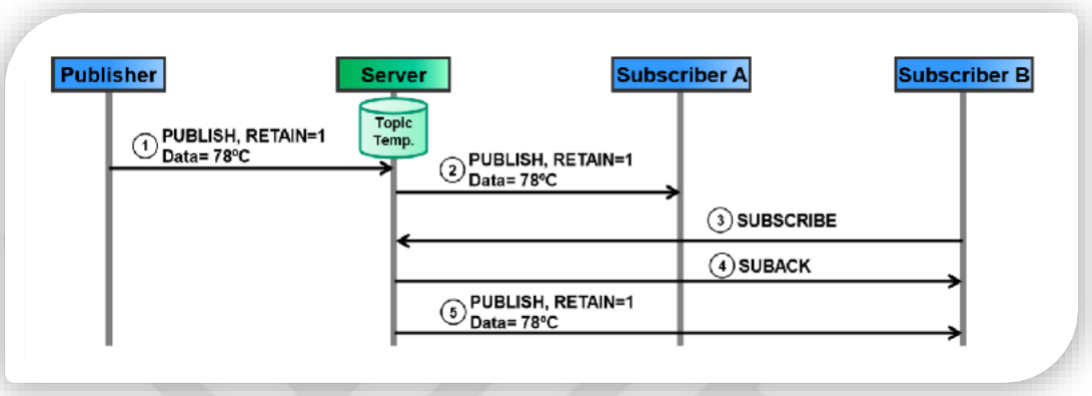


Figure 3.5: An Example of the Retain Field in the MQTT Message[9].

c. Remaining length (RL)

The variable length header and payload lengths are encoded in the remaining length field. RL is a variable length field with 1–4 bytes in order to save bits. The length field byte's most significant bit is the Continuation Bit (CB). It is set to 1 if more bytes come after it. The remaining length is encoded as "(a * 1280+ b * 1281+ c * 1282 +d * 1283)" and placed into the RL field bytes , as shown in Figure 3.6.

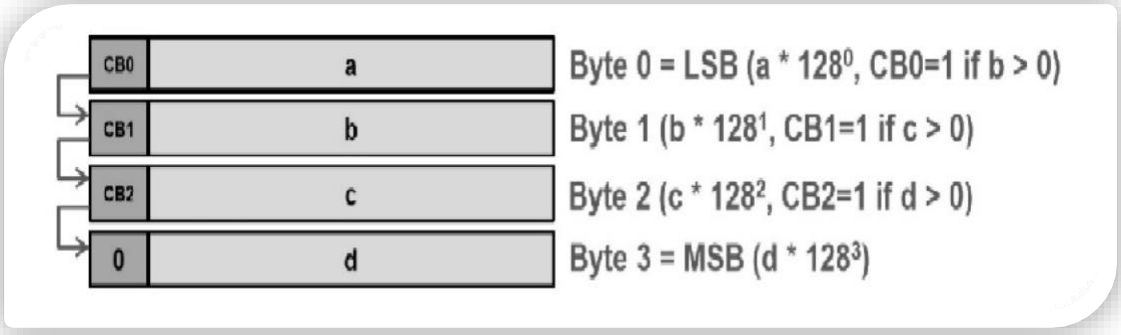


Figure 3.6: An Example of the Remaining Length Field in the MQTT Message[9].

d. MQTT Protocol Services

MQTT protocol provides three standard Qualities of Service (QoS) levels of message delivery, as illustrated in Figure 3.7[32]. Data loss may occur even if TCP/IP assures data transmission if a TCP connection breaks down and messages in transit are not received. MQTT thereby augments TCP with quality of service.

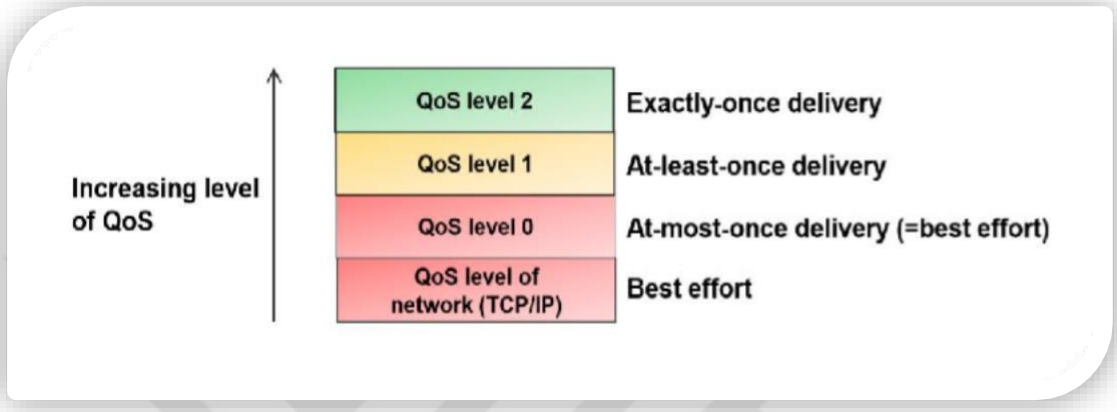


Figure 3.7: Quality of Services Levels in MQTT Protocol[9]

3.1.7 Mqtt Characteristics

a. MQTT is a lightweight message queuing and transport protocol for two reasons. Initially, it offers a minimal header structure, which minimizes packet processing and qualifies it for IoT devices with limited energy resources. Also, it is simple to put into practice. MQTT is a bi-directional communication protocol.
b. MQTT is an asynchronous messaging protocol. (events).
c. Low overhead (Two Bytes for Header) for applications requiring less communication bandwidth; small message payloads up to 256 MB in size are required.
d. Publish / Subscribe (Pub/Sub) model.
e. MQTT is a topic-based protocol that separates data producers (Publications) and consumers (Subscribers).
f. MQTT is a simple protocol for low power, complexity, and area implementations (e.g., Wireless Sensor Networks- WSN).
g. It is important to note that the protocol is symmetric, meaning that clients and brokers can act as sender and receiver when MQTT transmits application messages.

3.1.8 Mqtt Limitations

The MQTT protocol has the following limitations[14][41]:

a. MQTT's lightweight design philosophy does away with encryption for data transmission. Some MQTT brokers incorporate encryption as a separate function via TLS, which adds overhead. The MQTT's default plain-text data exchange poses a serious danger in terms of security.
b. There have been increased security attacks on smart devices connecting to a network and exchanging data using various IoT protocols.

3.1.9 Mqtt Protocol Vs Coap Protocol

The leading lightweight messaging protocols for restricted devices, MQTT and CoAP, are quickly taking over the IoT market and integrating them. Table 3.2 summarizes their differences[40].

Table 3.2: The Difference between MQTT and CoAP Protocols.

MQTT Protocols	CoAP Protocols
It runs over **TCP** (**T**ransmission **C**ontrol **P**rotocol)	It runs over **UDP** **(U**ser **D**atagram **P**rotocol)
It uses QoS to make sure that message arrives with communication overhead	To ensure that messages arrive without generating a high communication overhead, it uses basic error checking and verification for UDP.
MQTT utilizes QoS levels (At- most-once, At-least-once, Exactly- once)	CoAP utilizes Four message: Confirmable, N0n-C0nfirmable, Reset, and Ackn0wledgment. Tw0 0f them, however, deal with reliability messages.
Pub/Sub model	Request/Respond model
Standards by OASIS (Organization Advancement Structured Information Standards)	Standards by the IETF (Internet Engineering Task F0rce)

Table 3.2: The Difference Between MQTT and CoAP Protocols “Table Continued”.

Supp0rt mobile devices	**Support mobile devices via HTTP proxy**
Support security by Username/Password, Authentication, and SSL (Secure Sockets Layer) for data encryption	Support security by Datagram Transport Layer Security (DTLS) for data encryption
MQTT has been used for remote monitory applications[9][42], messaging applications and a range of home automation applications [19]. Furthermore, MQTT has also been used for education control[43].	For resource-constrained networks and M2M applications, CoAP has been used in a variety of application domains, ranging from building and home automation[44], smart grid, and smart cities to the healthcare industry[45], where the patient's status was updated in real-time by CoAP.

In[46], a detailed comparison of the protocols MQTT, XMPP, CoAP, HTTP, and others is covered.

3.1.10 Mqtt Server (Broker)

A server known as an MQTT broker receives all client messages and then forwards them to the appropriate recipients. Sometimes the MQTT br0ker is n0t sufficiently protected against D0S (Denial 0f Service) and other attacks t0 stay lightweight and energy-efficient, affecting reliable IoT sensors[23].

IoT Brokers, for instance, include[9]: RabbitMQ, ActiveMQ, ZeroMQ, Mosquitto, 0ri0n C0ntext Br0ker, and YAMI4[47]. Depending on the definitions of the application requirements, features like the Q0S (Quality 0f Service), r0bustness, perf0rmance, and adaptability can be essential when choosing an I0T br0ker[47]. Using brokers results in effective data manipulati0n and m0re scalable design but at the cost 0f increasing complexity.

Mosquitto is the most well-known and widely utilized MQTT broker[33]. Theoretically, MQTT brokers can support up to 45,000 concurrently connected MQTT clients based on implementation[48]. The MQTT broker's primary responsibilities for[33]:

a. Filtering messages.
b. Receiving messages.

c. Sending messages to clients.
d. Sorting messages.
e. Authentication/Authorization.

3.2 CLASSICAL ENCRYPTION ALGORITHMS

Encryption is perhaps the most basic and most important component of communications security, because it is of great and growing importance to the security of computers and the information they contain[49][50].

The risk of theft of proprietary information has increased as a result of the industry's increased use of computer and communications systems. Despite these risks that may require a different set of defences or countermeasures, encryption remains the main and basic strategy for protecting sensitive electronic data and important information being the most important automated tool for communication and network security[28][50].

There are Three types of encryption that are frequently used[28][50]**:**

a. Symmetric Encryption, also known as Conventional Encryption.
b. Asymmetric Encryption, also known as Public-key Encryption.
c. HASH Function.

3.2.1 Symmetric Cipher Model

In the 1970s before the development of public key cryptography, symmetric cipher was the only type of cipher used and it is still by far the most used. Symmetric Encryption is a type of Cryptosystem in which the same key (Single key, Private Key) is used for both encryption and decryption. It also goes by the name "Conventional Encryption"[28][50].

Using a encryption algorithm and an secret key, symmetric encryption converts plaintext into cipher text. The cipher text is decrypted using the decryption algorithm and a same key to reveal the plaintext [28][49].

Traditional Symmetric Ciphers used before computers[50]:

a. Substitution Techniques

Transform characters and bits from plaintext into cipher text.

b. Transposition Techniques

Transpose plaintext elements' positions in a methodical manner.

Figure 3.8 Represents the Components of a Symmetric coding scheme.

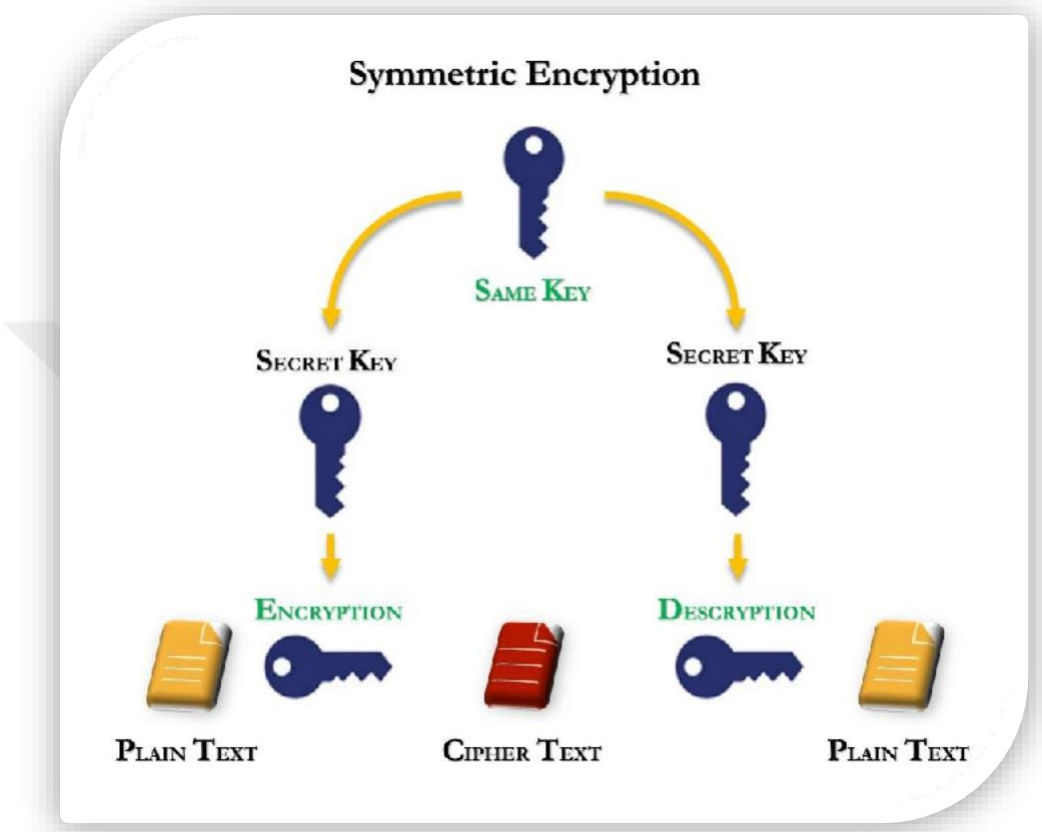


Figure 3.8: Symmetric Cipher Model.

a. Plain Text

Represents the text of the original message that is entered into the encryption algorithm.

b. Encryption Algorithm

It is the technique and approaches in which plain text is taken and several different transpositions and Substitution are performed on it.

c. Secret key

It is one of the entries of the encryption algorithm where it is independent of the algorithm and the plain text, and depending on it, the algorithm will give us different results depending

0n the key used at that time. One of the importance of the secret key is that it determines the exact transformations and substitutions that the algorithm makes.

d. Encrypted Text

It represents a random and incomprehensible message of data, which is the result of an encryption algorithm that relies on the secret key and plain text. F0r a given message, tw0 different keys will pr0duce tw0 different cipher text.

e. Decrypti0n Alg0rithm

It is the technique and process in which the encryption algorithm is reversed in order to obtain the plain text that represents the clear and understandable data by using the ciphertext and the secret key.

To use Symmetric Encryption securely, the following Requirements must be met[28][50]:

a. Strong encryption algorithm.

Even if an adversary has access to multiple cipher texts, with the plaintext generating each one, it must be unable to decrypt the cipher text or find the key.

b. Only the sender and recipient are aware of a secret key.

The secret key must have been obtained by the sender and recipient in a secure manner, and both parties are required to keep the key safe. All communications that use that key can be read if the key can be found and the algorithm is known.

To put it another way, the only thing we must keep a secret is the key, not the algorithm. Symmetric encryption can be used widely because of this characteristic.

3.2.1.1 Classical substituti0n ciphers

A substituti0n method is 0ne in which plain text characters are replaced with another type of character, number, or symbol. The substitution technique replaces the bit patterns of the plain text with the bit patterns of the ciphertext[49][50].

There are many Substitution algorithms that depend on this technique, including[49][50]:

a. Caesar Cipher.
b. Mono alphabetic Cipher.

c. Playfair Cipher.
d. Polyalphabetic Ciphers.
e. Hill Cipher.
f. Vigenère Cipher.
g. Kasiski Method.
h. Auto key Cipher.
i. One-Time Pad.

3.2.1.2 Classical transposition ciphers

This technology carries out the encryption process for the plain text using permutation by flipping the letters with each other for the plain text[50].

There are many Transposition algorithms that depend on this technique, including[49][50]:

a. Rail Fence cipher.
b. Row Transposition Ciphers.
c. Product Ciphers.
d. Rotor Machines.
e. Steganography.

3.2.2 Asymmetric Cipher Model

Methods encrypt data using one key and decrypt it with a separate key. The first key is public, while the second is private. In a public-key encryption (Asymmetric Cipher) system, anyone can encrypt a message using the recipient's public key, but only the receiver can decrypt it using his private key. As shows in the Figure 3.9 parties exchange encrypted communications without the requirement for a pre-shared key. Asymmetric algorithms are more difficult to implement. These algorithms use more resources and take longer to complete[28][50].

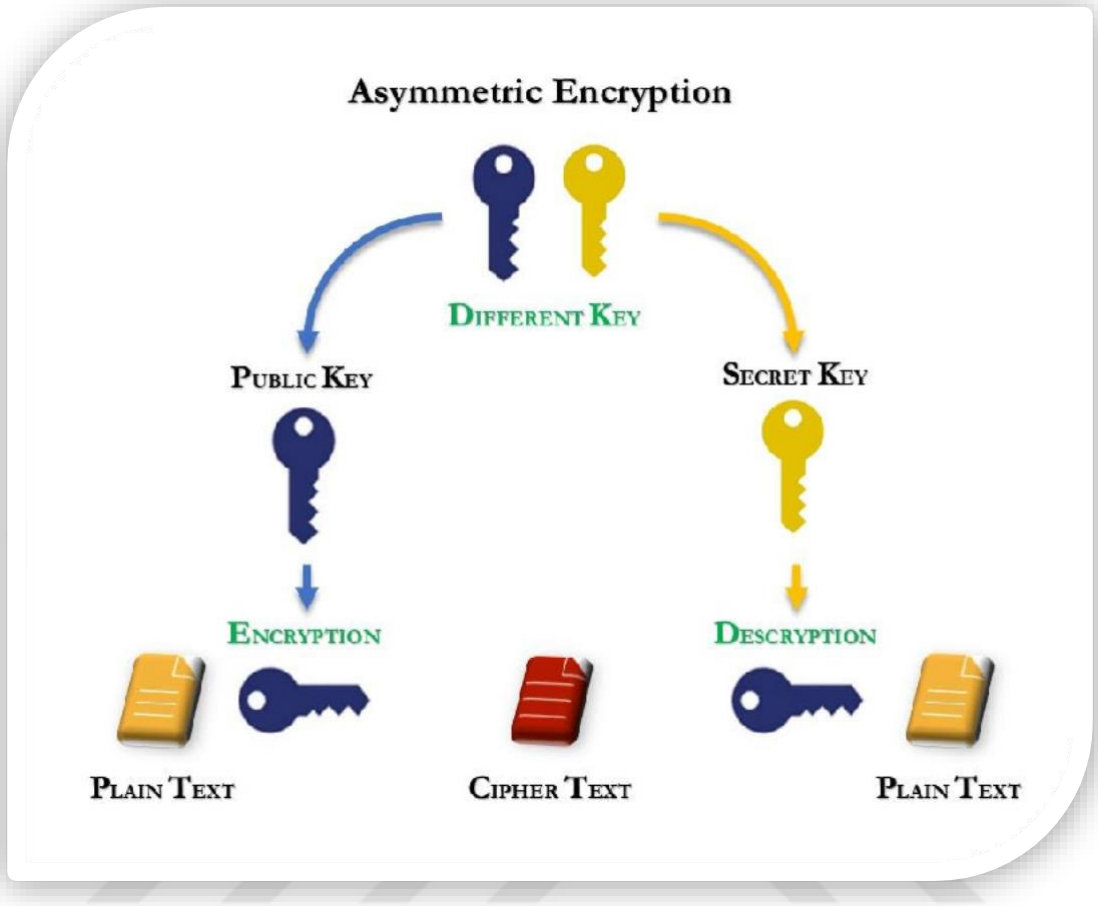


Figure 3.9: Asymmetric Encryption

There are Several Asymmetric Algorithms[28][50]**:**

a. RSA (Rivest-Shamir-Adleman)

Utilizes the product of two extremely large prime numbers that are equal in size and have between 100 and 200 digits in length. Browsers create secure connections using RSA.

b. DH (Diffie-Hellman)

Provides a method for the secret key to be shared electronically. Diffie-Hellman is a key component of secure protocols like IPsec, Transport Layer Security (TLS), Secure Shell (SSH), and Secure Sockets Layer (SSL).

c. ECC (Elliptic Curve Cryptography)

Uses elliptic curves as part of the algorithm. In the U.S., the National Security Agency uses ECC for digital signature generation and key exchange.

d. ElGamal

Uses the U.S. government standard for digital signatures. This algorithm is free for use because no one holds the patent.

Table 3.3: Compared between Symmetric & Asymmetric Encryption Algorithm.

Symmetric Encryption Algorithm	**Asymmetric Encryption Algorithm**
Best known as shared-secret key algorithms	Best known as public key algorithms
The usual key length is 80 to 256 bits	The usual key length is 512 to 4.096 bits
A sender and receiver must share a secret key	A sender and receiver do not share a secret key
Algorithms are usually quite fast (wire speed) because they are based on simple mathematical operations	Algorithms are relatively show because they are based on difficult computational algorithms
Examples include AES, IDEA, DES, 3DES, RC2/4/5/6, and Blowfish	**Examples** include RSA, EIGamal, ECC, DH

3.3 ENCRYPTION ALGORITHMS ATTACK

Attacks are essentially unauthorized users trying to analyse cryptography. There are seven basic types of attack, which are described here. It is assumed that attackers, such as crypto analysts, have in-depth knowledge of the encryption technique.[28].

a. Ciphertext Only Attacks

b. By examining many encrypted messages, the analyst only detects ciphertext that has to be decoded and makes an effort to find the encryption key or to decrypt certain segments

of the ciphertext. In order to view data outputs, the attacker tries to decrypt specific messages or figure out the encryption key used[28].

c. Known Plaintext Attack

d. Compared to a ciphertext-only assault, this kind of attack substantially facilitates the attacker's task. With samples of both plain text and cipher, an attacker can try to figure out the encryption techniques and get the key[28].

e. Chosen Plaintext Attack (CPA)

f. Similar to differential crypto analysis, where the attacker is recognized as being at the encryption site, in CPA the attacker has selected portions of plain text [28].

g. Chosen Cipher text Attack (CCA)

h. In CCA, the attacker has both the matched plain text and the chosen cipher text, making it easier to decrypt data using the private key. After obtaining the desired messages, it only has access to an encryption device[28].

i. Chosen Text

j. The attacker has the cipher text to be decrypted, the encipher algorithm, related ciphertexts, the selected plain text messages, fabricated ciphertext, and matching decoded plain texts and developed by the private key[28].

k. Brute Force Attack (BFA)

l. This is a dedicated, time-consuming, and expensive trial-and-error method, as previously described, where the attacker attempts to obtain the encryption key by performing one-by-one encryption iterations using automated software that generates enormous numbers of guesses. Such attacks typically require supercomputers, making them unprofitable in the vast majority of cases[28].

m. Man in the Middle Attack

n. In this attack, as shown in figure 3.10, communication between two parties is secretly intercepted and forwarded (perhaps in altered form) between the users. This allows attackers to deceive senders and capture the secret key[28].

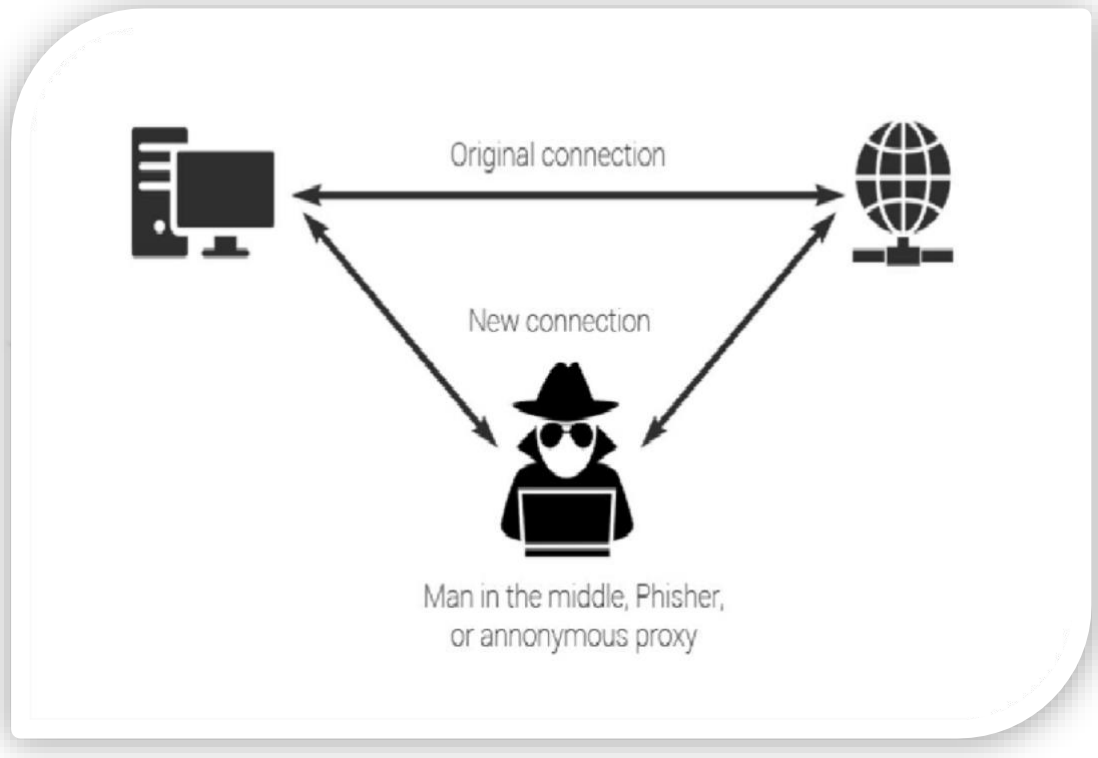


Figure 3.10: Man in the Middle Attack[28].

Based on the probability and prevalence of characters in messages, the frequency analyst (differential attacks) attempts to extract plain text by examining the frequency of characters in encrypted text[28]. Since 'E' is the most often used letter in the English language, if 'D' is the most common letter in a cipher text, it is likely that this is the scrambled version of 'E', allowing the attacker to identify additional letters and decipher the code. Table 3.4 below show the frequency for each letter in the English language[28]:

Table 3.4: Frequency of Alphabet Letters in English [28].

Letter	Frequency	Letter	Frequency
e	0.12702	u	0.02758
t	0.09056	m	0.02406
a	0.08167	w	0.02360
o	0.07507	f	0.02228
i	0.06966	g	0.02015
n	0.06749	y	0.01974
s	0.06327	p	0.01929
h	0.06094	b	0.01492
r	0.05987	v	0.00978
d	0.04253	k	0.00772
l	0.04025	j	0.00153
c	0.02782	x	0.00150

3.4 INTEGRITY

Integrity guarantees that data is reliable and unaltered throughout its entire life cycle, that means data received from an authorized entity is exactly as it was sent[28][50].
Any system that transmits data or processes or stores must be designed, implemented, and used in a way that ensures data integrity[50].

3.5 HASH FUNCTION

Users need to be aware that while at rest or in transit, their data is unaltered. As shown in the Figure 3.11 hashing is a technique for ensuring preserving data integrity by creating a fixed-length representation from binary data (the message) known as the hash value or message digest.

To check and guarantee data integrity, the hashing tool employs a cryptographic hashing function. It can validate authentication as well[49].

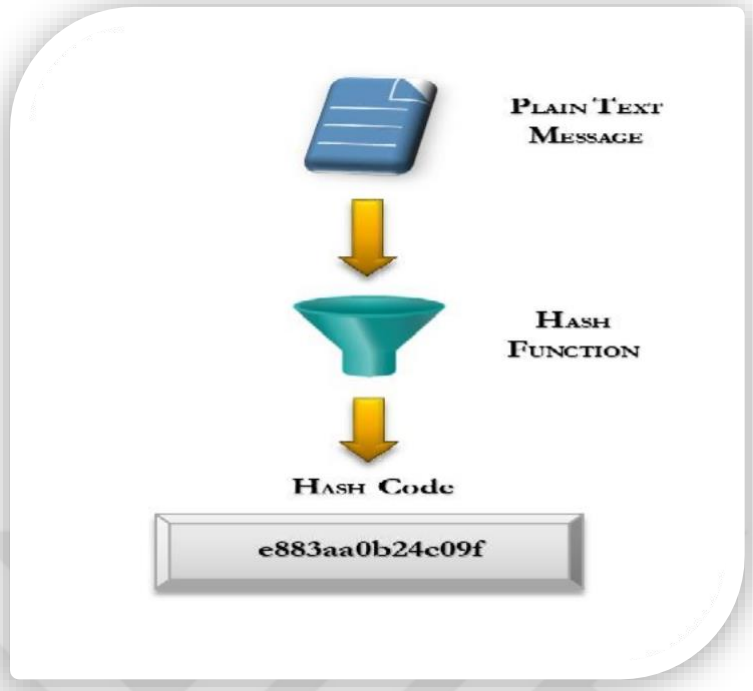


Figure 3.11: Hash Function.

3.5.1 Hashing Properties

A one-way mathematical function called hashing is comparatively simple to compute but much more difficult to reverse.

The characteristics of a cryptographic hash function are as follows[49]:

a. The input's length is unrestricted.
b. The length of the output is fixed.
c. The hash function is unidirectional and irreversible.
d. Almost never will two different input values yield the same hash values.

3.5.2 Hash Function Algorithms

a. Simple Hash Algorithm (8-bit Checksum)

One of the earliest hashing algorithms and the most basic hash function is the 8-bit checksum. An 8-bit checksum organizes the string of binary numbers into 8-bit chucks after converting the message into binary numbers to calculate the hash[49].

There are numerous contemporary hashing algorithms in use right now. The two most widely used are SHA and MD5 (Message Digest 5).

b. Secure Hash Algorithm (SHA)

The National Institute of Standards and Technology (NIST) of the United States developed the Secure Hash Standard (SHS) algorithm, which uses the SHA algorithm[49].

SHA-1 was released by NIST in 1994. SHA-2 replaced SHA-1 and added four new hashing operations to create the SHA family [49]:

a. SHA-512 (512 bit)
b. SHA-384 (384 bit)
c. SHA-256 (256 bit)
d. SHA-224 (224 bit)

Since SHA-2 is a more robust algorithm, MD5 is being phased out. The newest algorithms are SHA-256, SHA-384, and SHA-512.

3.6 PROPOSED ENCRYPTION ALGORITHM

A new encryption methods are proposed called *Lightweight Security and Integrity Data – Internet of Thing* ***LSID-IoT***, the main advantages of the proposed methods that they are easy and quick to implement. It contains an implicit key (extracted from plain text) so it is classified as substitution encryption algorithms because they replace the plain text characters by another's to produce the cipher text, encrypt both text (English and Arabic) and number, strong against to crypto analyser. In addition, End-to-End encryption.

3.6.1 Encryption Text methods

The encryption process starts by converting the plain text to a set of characters. Then, encrypted each character by finding the encryption alphabetical from shifting the original alphabets base on the sequence of the character in the plain text. Later, taking the corresponding encryption character to the plain character. Encryption process applied according to the Eq. (1). Figure 3.14 shows the flowchart of the encryption process.

$$C[I] = P[I] + (I \bmod 26) \quad (3.1)$$

Where C is an array holds Cipher text, and P is an array holds Plain text and I is character index in plain text or cipher text.

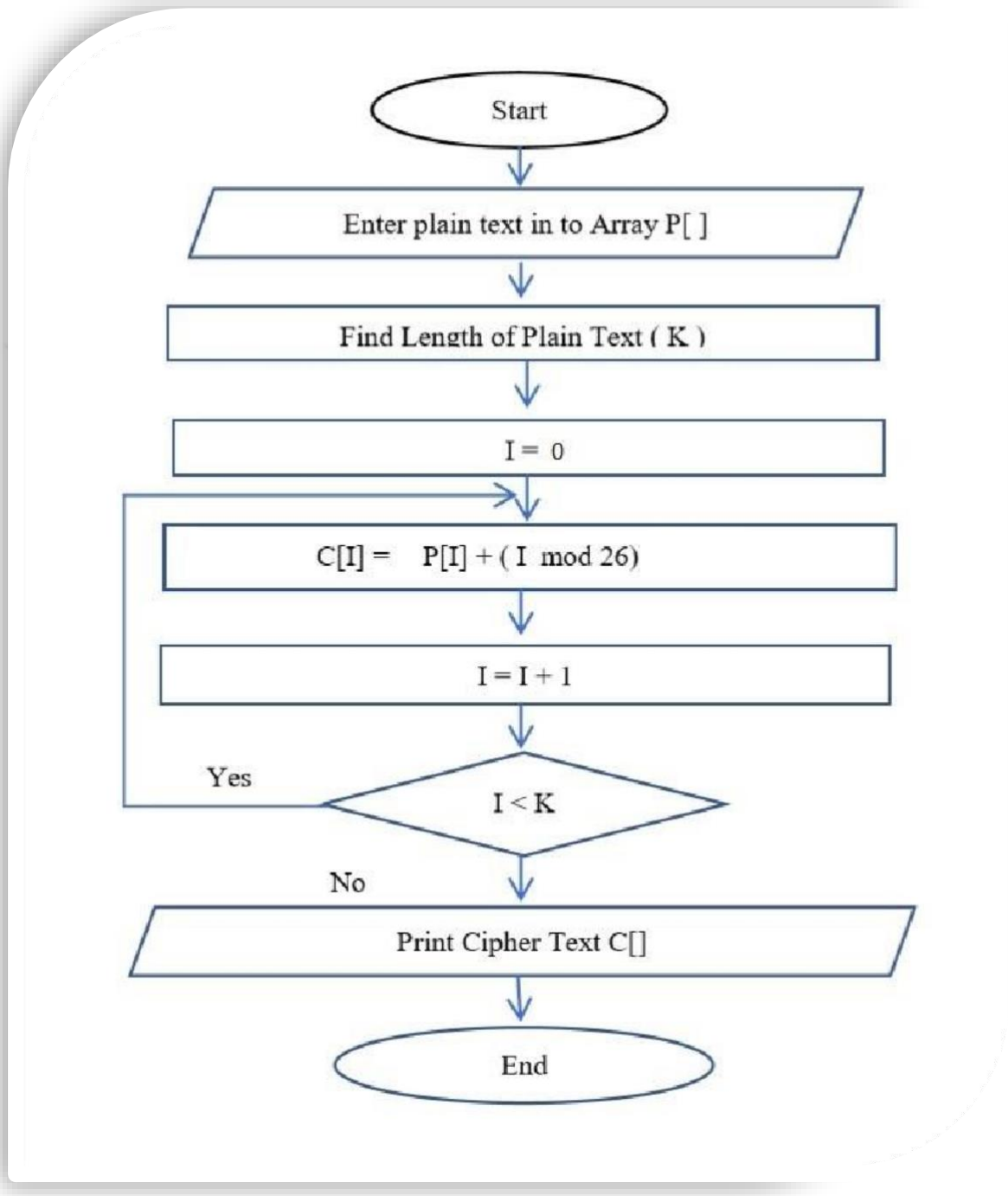


Figure 3.12: Flowchart of the Encryption Process.

Below is Figure 3.13 an example for the Text Encryption.

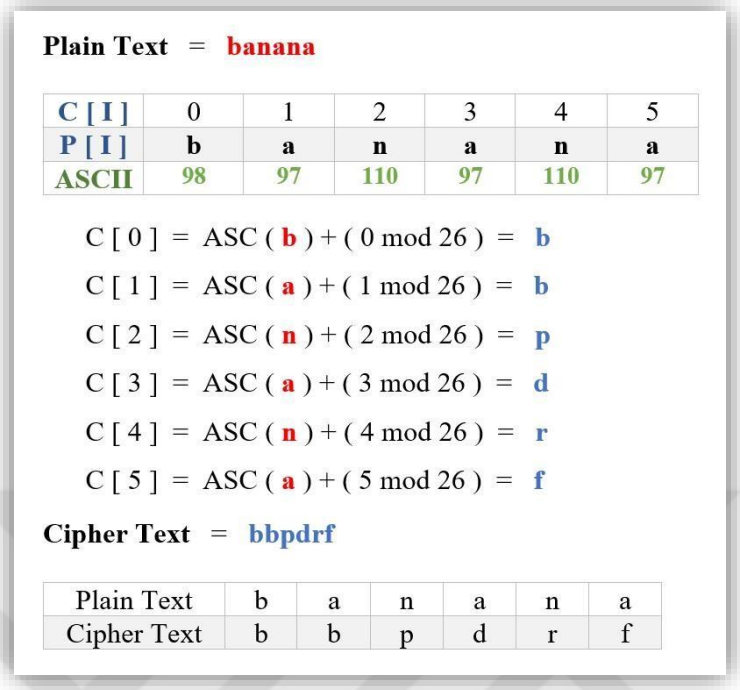


Figure 3.13: Example of Text Encryption.

3.6.2 Decryption Text methods

The decryption process starts in an opposite way to the encryption process by taking the cipher text then converting it to a set of characters. After that, decrypt the character by finding the decryption alphabetical from shifting the original alphabets base on the sequence of the character in the cipher text. The last step is taking the corresponding plain character to the cipher character. Decryption process is applied according to the Eq. (2). Figure 3.14 shows the flowchart of decryption process.

$$P[I] = C[I] - (I \bmod 26) \tag{3.2}$$

where C is an array holds Cipher text, and P is an array holds Plain text and I is character index in plain text or cipher text.

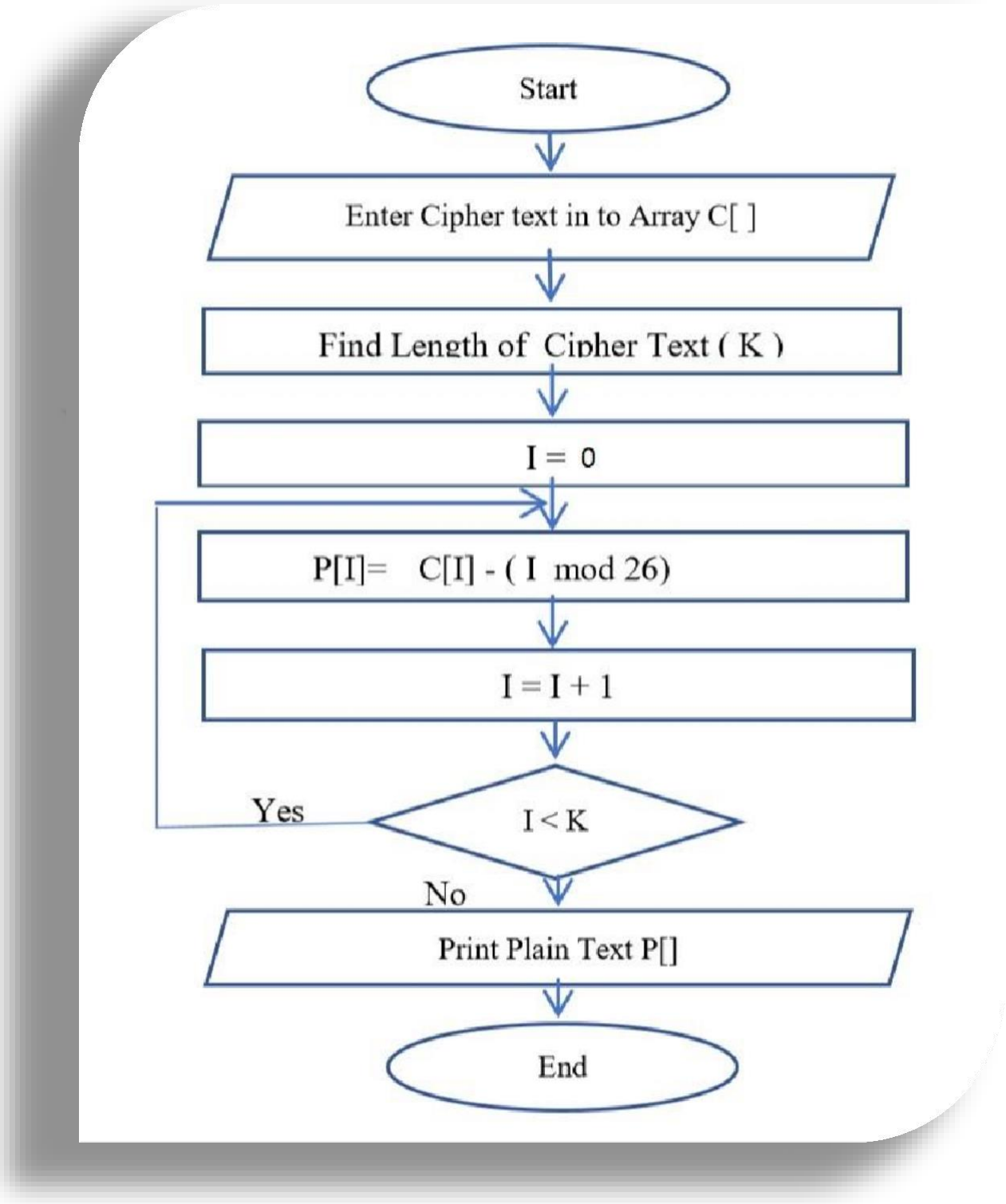
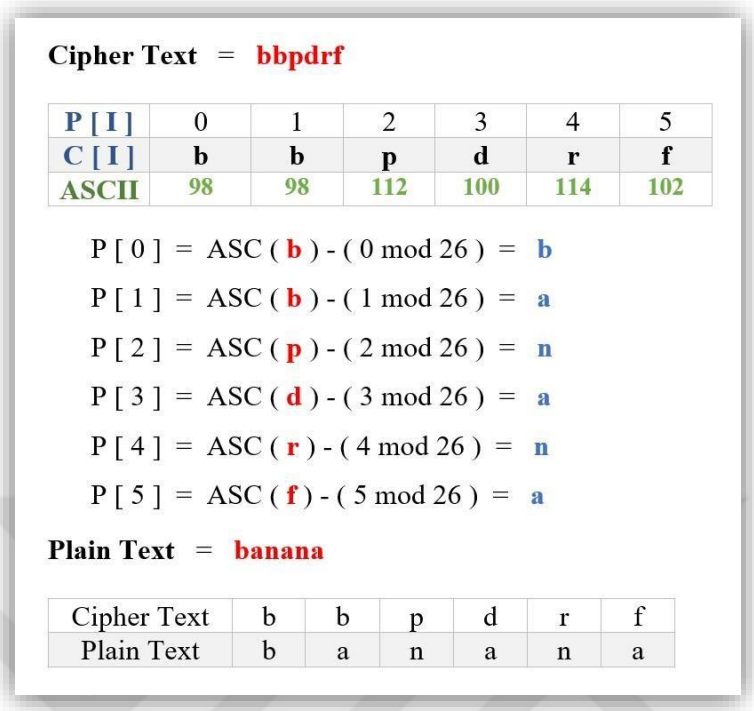


Figure 3.14: Flowchart of the Decryption Process.

Below is Figure 3.15 an example for the Text Decryption.



Cipher Text = bbpdrf

P [I]	0	1	2	3	4	5
C [I]	b	b	p	d	r	f
ASCII	98	98	112	100	114	102

P [0] = ASC (b) - (0 mod 26) = b

P [1] = ASC (b) - (1 mod 26) = a

P [2] = ASC (p) - (2 mod 26) = n

P [3] = ASC (d) - (3 mod 26) = a

P [4] = ASC (r) - (4 mod 26) = n

P [5] = ASC (f) - (5 mod 26) = a

Plain Text = banana

Cipher Text	b	b	p	d	r	f
Plain Text	b	a	n	a	n	a

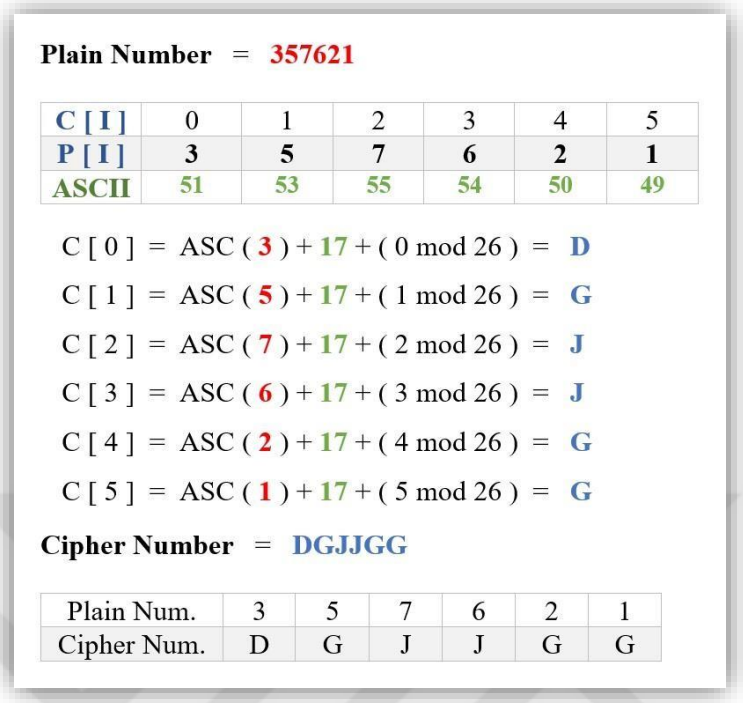
Figure 3.15: Example of Text Decryption.

3.6.3 Encryption and Decryption Numbers Methods

In addition, the same encryption algorithm was used to encrypt the numbers with simple changes, translate the number digits to char by adding (17) to number ASCII and encrypt the character. For example, to encrypt any number Eq. (3) is used.

$$C[I] = (P[I] + 17) + (I \bmod 26) \qquad (3.3)$$

Below is Figure 3.16 an example to Encrypt a Number.



Plain Number = 357621

C [I]	0	1	2	3	4	5
P [I]	**3**	**5**	**7**	**6**	**2**	**1**
ASCII	51	53	55	54	50	49

C [0] = ASC (**3**) + 17 + (0 mod 26) = **D**

C [1] = ASC (**5**) + 17 + (1 mod 26) = **G**

C [2] = ASC (**7**) + 17 + (2 mod 26) = **J**

C [3] = ASC (**6**) + 17 + (3 mod 26) = **J**

C [4] = ASC (**2**) + 17 + (4 mod 26) = **G**

C [5] = ASC (**1**) + 17 + (5 mod 26) = **G**

Cipher Number = DGJJGG

Plain Num.	3	5	7	6	2	1
Cipher Num.	D	G	J	J	G	G

Figure 3.16: Example of a Number Encryption.

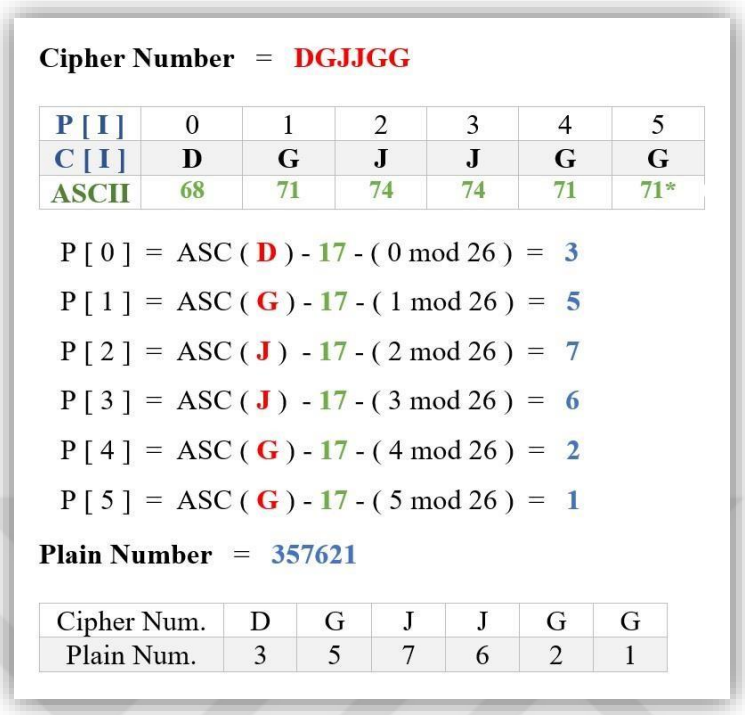
And to Decrypt any number Eq. (4) is used.

$$P[I] = (C[I] - 17) - (I \bmod 26) \qquad (3.4)$$

Where C is an array holds Cipher number, and P is an array holds Plain number and I is a number index in plain number or cipher number.

The proposed methods could be also deal with the Real numbers.

Below is Figure 3.17 an example to Decrypt a Number.



Cipher Number = DGJJGG

P [I]	0	1	2	3	4	5
C [I]	D	G	J	J	G	G
ASCII	68	71	74	74	71	71*

P [0] = ASC (D) - 17 - (0 mod 26) = 3

P [1] = ASC (G) - 17 - (1 mod 26) = 5

P [2] = ASC (J) - 17 - (2 mod 26) = 7

P [3] = ASC (J) - 17 - (3 mod 26) = 6

P [4] = ASC (G) - 17 - (4 mod 26) = 2

P [5] = ASC (G) - 17 - (5 mod 26) = 1

Plain Number = 357621

Cipher Num.	D	G	J	J	G	G
Plain Num.	3	5	7	6	2	1

Figure 3.17: Example of a Number Decryption.

3.6.4 Integrity

To achieving the integrity service in the proposed system, the SHA-256 (256 bit) algorithm was used as shown in Figure 3.18, where the first and last 2 hexa-decimal numbers of the output of the above algorithm was used to reducing the size of the sent message and achieving lightweight in addition to using low network bandwidth.

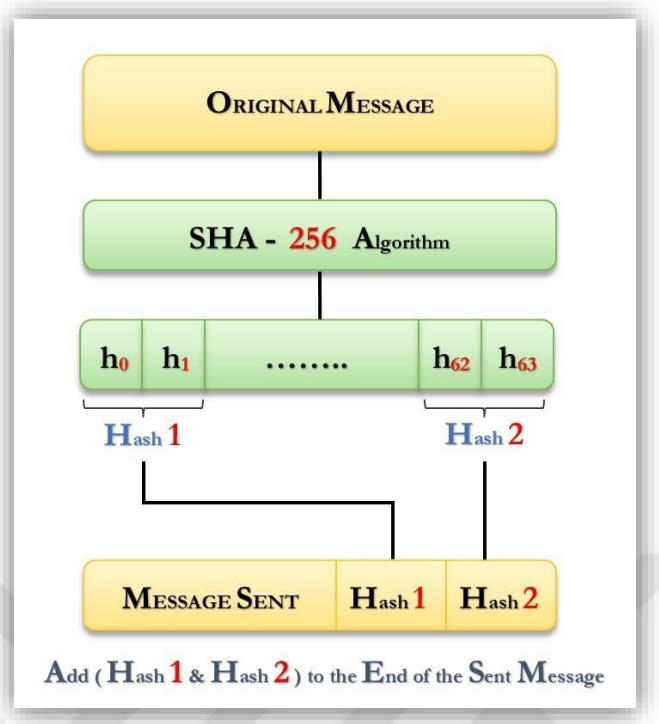


Figure 3.18: Flowchart of the Integrity service for proposed System.

3.7 DESIGN SUGGESTED SYSTEM

The proposed system was designed within the IoT environment in two stages:

i. Software Design Stage

At this stage, the proposed system was designed into two main parts:

a. The First Part, includes encryption and integrity of the send message as illustrated flowchart in Figure 3.19.

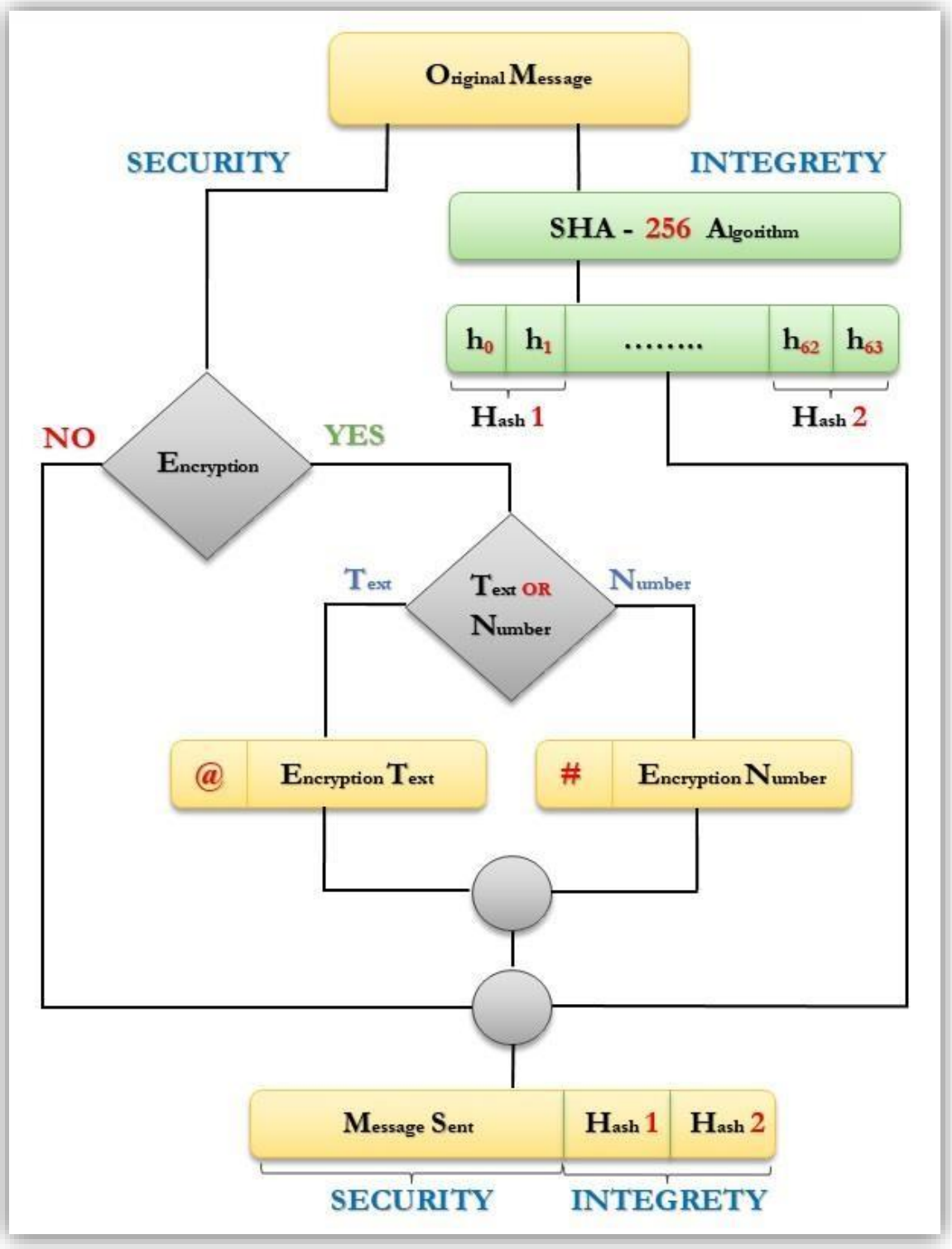


Figure 3.19: Encryption Flowchart.

b. The Second Part, includes decrypting and integrity of the received message as illustrated flowchart in Figure 3.20.

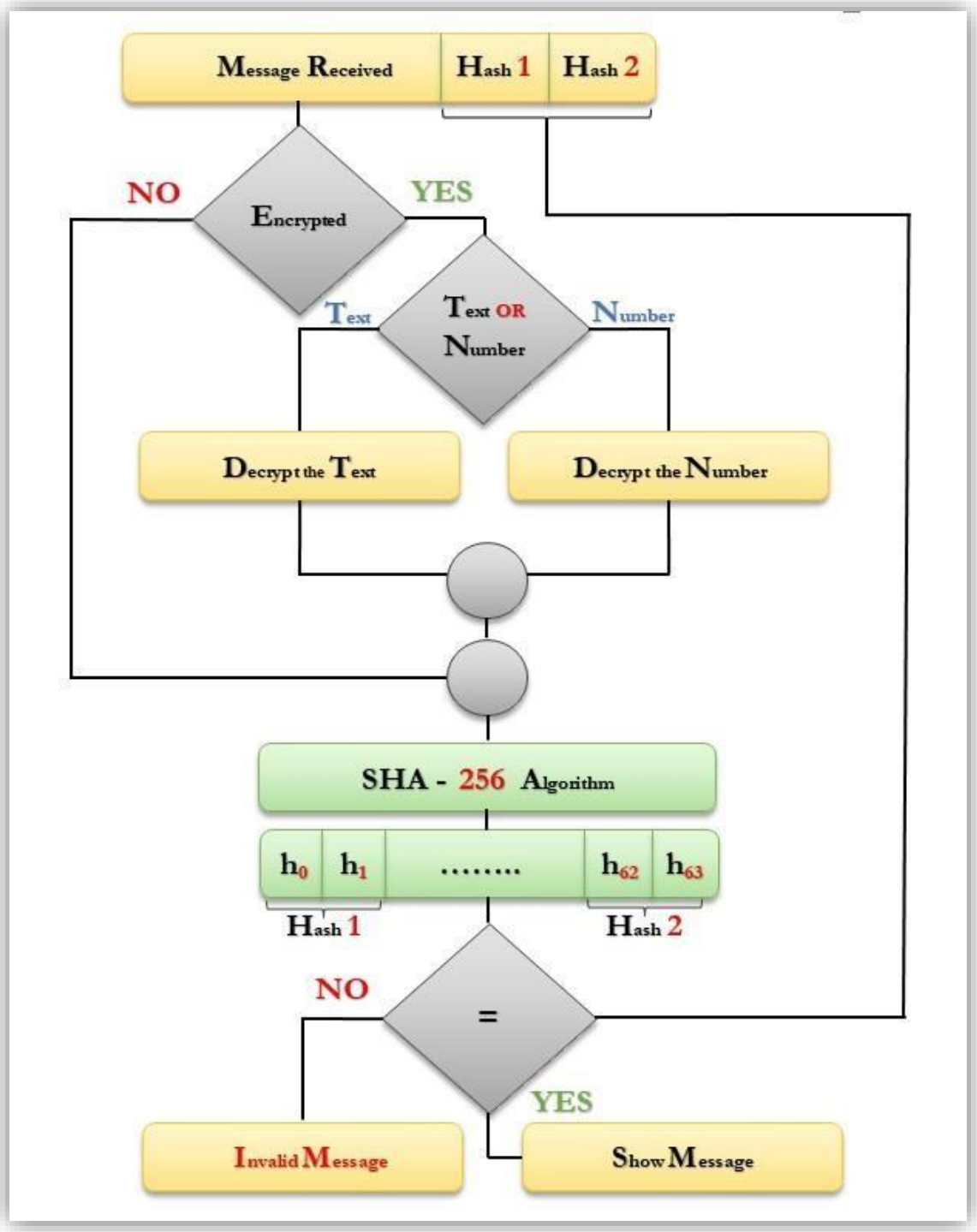


Figure 3.20: Decryption Flowchart.

ii. Hardware Design Stage

A Raspberry Pi4 device was used, containing Mosquitto Broker that acts as a coordinator for MQTT messages, in addition to using two computers. The first computer, PC1, is used

to send MQTT messages (Publisher Device), and the second computer, PC2, is used to receive MQTT messages (Subscriber device), as shown in the Figure 3.21.

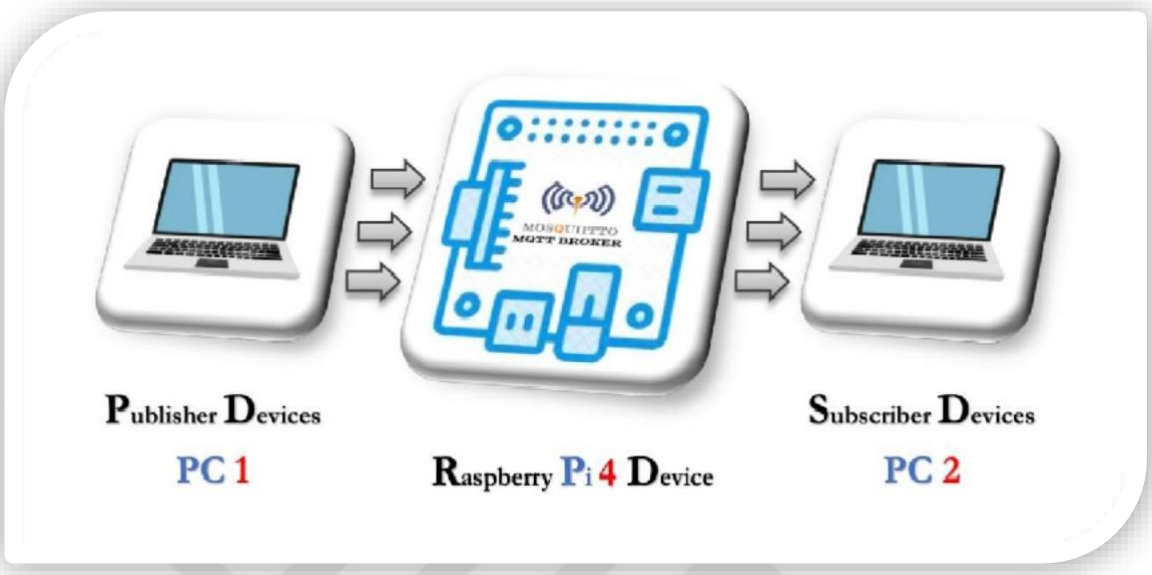


Figure 3.21: General IoT Network for the Proposed System.

3.8 TOOLS AND PROGRAMMING LIBRARIES FOR IMPLEMENTATION

The open-source Anaconda distribution platform is regarded as the most well-known worldwide[51]. It has the following features: 'Open Source,' i.e., access to any open-source software needed for developing projects in any area is allowed. 'User-friendly,' i.e., the quick search and install packages are very easy and provide the ability to create, load, and switch between environments. 'Trusted,' i.e., the carefully tested and consistently updated packages and artefacts are hosted securely. With Anaconda Distribution, users have access to a massive library of community material and support as well as hundreds of Python and R packages.

To implement and evaluate the experiments of the proposed system, a Core i7 processor (8GB of RAM) with the Anaconda 3.0 distribution was used, which includes the following libraries and editors:

a. An interactive Python-based (IDE) called (**Jupyter**) Notebook version 6.4.3 was utilized as a software tool to implement and evaluate proposed experiments.

b. The powerful python IDE with advanced editing, interactive testing, and debugging called (**Spyder**) version 4.2.5 was used to implement the applications.

3.9 TESTING STAGE DESIGN

Many Hardware and Software tools are needed to investigate the performance of the proposed system in the real environment.

3.9.1 Hardware Tools

The hardware tools are PCs, and Raspberry Pi.

a. PC devices

These devices are used to generate normal and encrypted traffics. Two python applications were developed to generate normal or encrypted traffic, which can work as sensors and actours. One generates normal or encrypted then sends it to the specific topic in the Mosquitto broker using the MQTT protocol (Publish Message). The second application receives a message (Normal or Encrypted Traffic) from the same topic in the Mosquitto broker using the MQTT protocol (Subscribe Message).

b. Raspberry Pi 4 Model B

The most recent model of the well-known Raspberry PI family of computers is the Raspberry PI4 Model B (PI 4B). Compared to the Raspberry PI 3 Model B, it offers significant speed improvements for the CPU, memory, multimedia performance, and connectivity while maintaining backward compatibility and similar power consumption[52][53][54]. Figure 3.22 shows the main components of the Raspberry PI 4B.

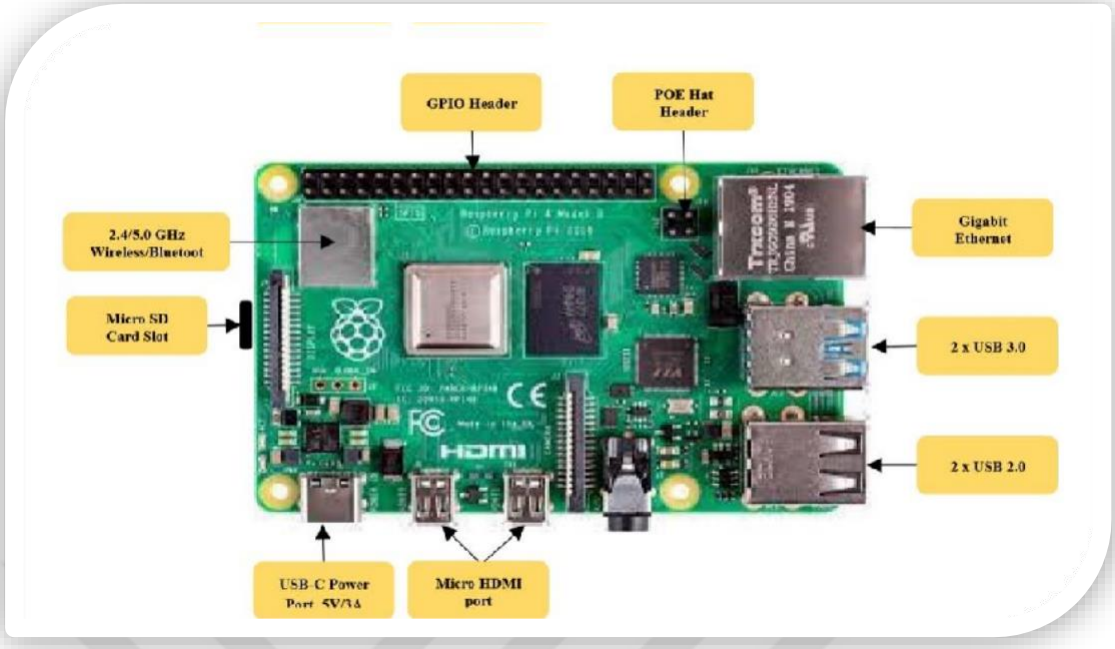


Figure 3.22: Main Components of the Raspberry PI 4B[55].

3.9.2 Installing the Operating System and Configuring the Raspberry PI 4B

To use the Raspberry PI 4B as a microprocessor, an operating system must be installed on it, which can be done by the following steps:

a. Connect the SD card to the PC or laptop.

b. Whether you use a Mac, Linux, or Windows computer, you must download Raspberry Pi Imager.

c. Select and install the Raspberry Pi 4 OS (32-bit) on the SD card.

d. When finished, remove the SD card from the laptop and insert it in the SD slot on the Raspberry.

e. The Raspberry can now be powered on by connecting a power supply via the USB-C connector.

f. After this is completed, it can take a few seconds or a minute, but then the Raspberry Pi OS Home screen should be displayed on the monitor.

g. Finally, some pop-up window setup appears for timing, language, and password Wi-Fi setup.

3.9.3 Installation and Configuration of Mosquitto Broker on the Raspberry PI 4B

Eclipse Mosquitto Broker is an open source (EPL (Eclipse Public License) / EDL (Enhanced Driver's License) licensed) message broker that implements the MQTT protocol versions 5.0, 3.1.1, and 3.1. Mosquitto is lightweight and suitable for all devices, from low-power single-board computers to full servers . The following steps are used to install and configure the mosquitto broker v. 2.0.6 on the Raspberry Pi 4B.

i. Install Mosquitto on the Raspberry Pi 4B. Use the following commands:

```
# sudo apt-get install  mosquitto –y
# sudo apt-get install  mosquitto-clients
```

ii. Configure Mosquitto using the following steps:

a. Create a new file for configuration

```
# sudo nano /etc/ mosquitto/ Mosquitto.conf
```

b. Put the following command in the new configuration file:

```
pid_file /var/run/mosquitto.pid
persistence true
persistence_location /var/lib/mosquitto/
log_dest file /var/log/mosquitto/mosquitto.log
allow_anoymous false
password_file /etc/mosquittto/pwfile
listener 1883
```

c. Save the file.

d. Setup mosquitto credentials by the following command:

```
#sudo mosquitto_passwd –a /etc/mosquitto/pwfile (username) (password)
```

e. To make the mosquitto a startup application, use the following command:

```
#sudo systemctl enable mosquitto service
```

3.9.4 Software Tools

Virtual Network Computing (VNC) Viewer and PuTTY are software tools:

a. VNC Viewer

A VNC Viewer is a software tool that controls local computers and mobile devices remotely. For example, a device such as a computer, tablet, or smartphone with VNC Viewer software can access and take control of a computer in another location[56]. VNC provides a deceptively straightforward solution allowing monitoring and managing a remote system. VNC automatically and dynamically adapts to a dynamic environment, such as different network bandwidths and screen contents. A Windows PC can easily control a Linux server because of VNC's platform independence and vice versa. When using VNC, the viewer receives fragments of the server screen image that have been segmented, as shown in Figure 3.23. In addition, the viewer's key presses and mouse movements are transmitted to the server.

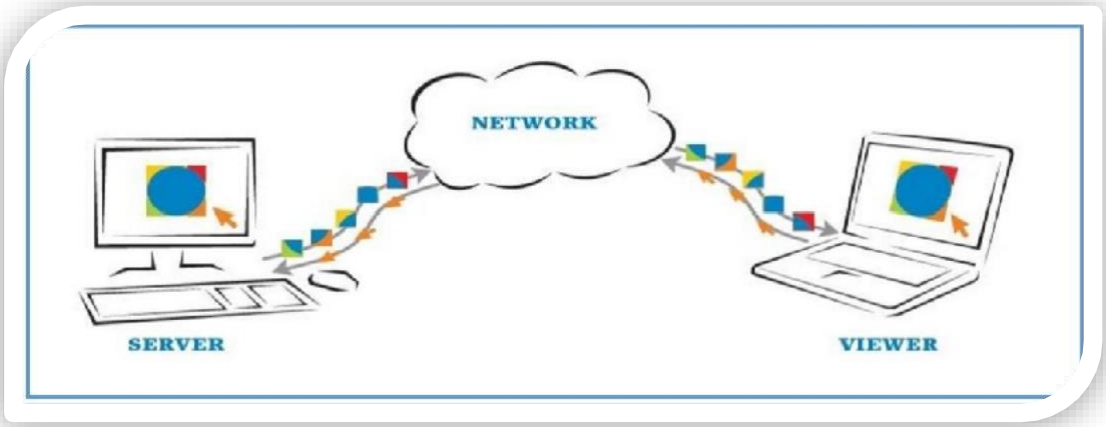


Figure 3.23: Using VNC Viewer (Fragment of the Server Screen Image).

VNC provides secure password protection. However, VNC servers prove their authenticity to defeat server spoofers by providing a special identification code before disclosing any viewing information. In order to create a significant barrier for attackers,

these characteristics are paired with the new high-strength connection encryption. So, the VNC tool has been used in this thesis to access and manage Raspberry Pi 4B from a laptop.

b. PuTTY Tool

PuTTY is an open-source, free network file sharing, serial console, and terminal emulator. Some of the network protocols it supports are SCP, SSH, Telnet, rlogin, and raw socket connections. Moreover, it can attach to a serial port[57].

Additionally, Users can modify the protocol version and SSH encryption key with PuTTY, in addition to controlling various ciphers including DES, Blowfish, AES, 3DES, and public-key authentication. SSH can also be used for dynamic, remote, or local port forwarding[57].

3.10 TESTING PROPOSED SYSTEM ON RASPBERRY PI

The testing phase of the proposed system has been done after finishing the entire configuration through implementing the tenser flow (64-bit) environment on Linux OS (32-bit) and completing the appropriate environment for executing the proposed system on Raspberry Pi devices, the proposed system consist of the following steps:

a. Defined the used libraries such as (**PYSHARK**, **asyncio**, **numpy**).

b. Determined if the proposed system show the MQTT Message on the command windows

c. Capture operation starts on the WLAN interface and captures only the MQTT messages using the PYSHARK library.

d. Extract the message filed from Publish MQTT messages.

e. Show the Extract message depended on step 2.

3.11 TESTING PROPOSED SYSTEM ON PCS

At this stage, Two Applications were developed using the Python language within the Anaconda environment.

i. The First Application

Sends MQTT messages to the broker inside the Raspberry device using the MQTT protocol. The development include the following steps:

a. Create the program interface using the standard GUI library for Python (Tkinter).

b. Develop a Integrity function to compute the HASH value.

c. Giving the user the ability to specify whether he wants to encrypt the message or not. If the user chooses the encrypt message, the proposed encryption algorithm is called to carry out the process of encrypting the message.

d. Connect to the Broker by providing the username and password

e. Sending the message, whether encrypted or unencrypted, to the broker by using the Publish function of the (**Paho library**), specifying the Topic within the broker that the message is stored in it.

ii. The Second Application

Receives MQTT messages from the broker inside the Raspberry device using the MQTT protocol, The development include the following steps:

a. Create the program interface using the standard GUI library for Python (**Tkinter**).

b. Develop a special function to provide the integrity.

c. Connect to the Broker by providing the username and password

d. listening to the port no. (**1883**) and determine the topic to received the MQTT message

e. received the MQTT message from the Broker and check it encrypted or not, if encrypted the application call the decrypt function in the proposed algorithm to decrypt the message.

f. Compute the HASH value of the MQTT message (after decrypt message) and compare it with the received HASH value. If equal the application will display the

MQTT message in the list box, otherwise the application will display invalid string message in the list box.

4. RESULTES

Many different experiments were conducted for verifying the performance of the proposed system by performed a set of scenarios that included encrypting and decrypting texts in Arabic and English language, in addition to numbers, and merging them. To realize from the efficiency of the proposed system, the time difference between processing the received of the original message (without encryption) and decrypt the encrypted message was calculated.

4.1 SCENARIO 1

This scenario include sending a message (**English Text only**) once without encryption and with encryption in addition to calculate the received time as follows:

c. Sending the message (**without Encryption**), as shown in Figure 4.1 .

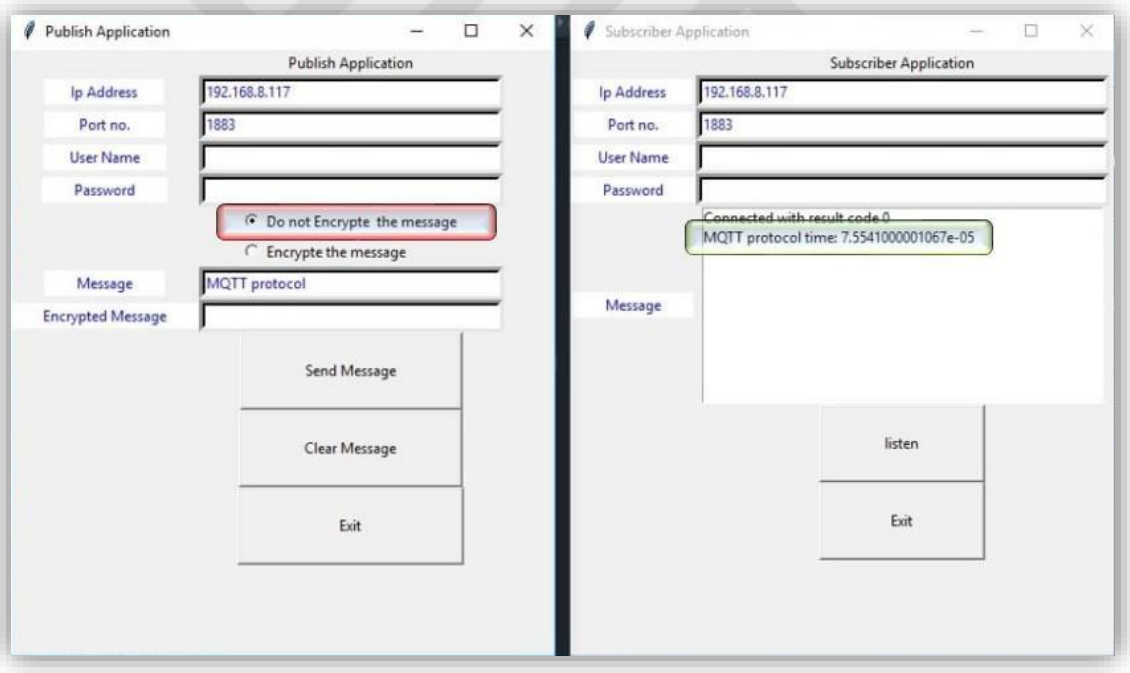


Figure 4.1: The Interface of the Pr0p0sed System to Sending Message (without Encryption) with Calculate the Received Time in Scenario (1).

d. Sending the message (**with Encryption**), as shown in Figure 4.2.

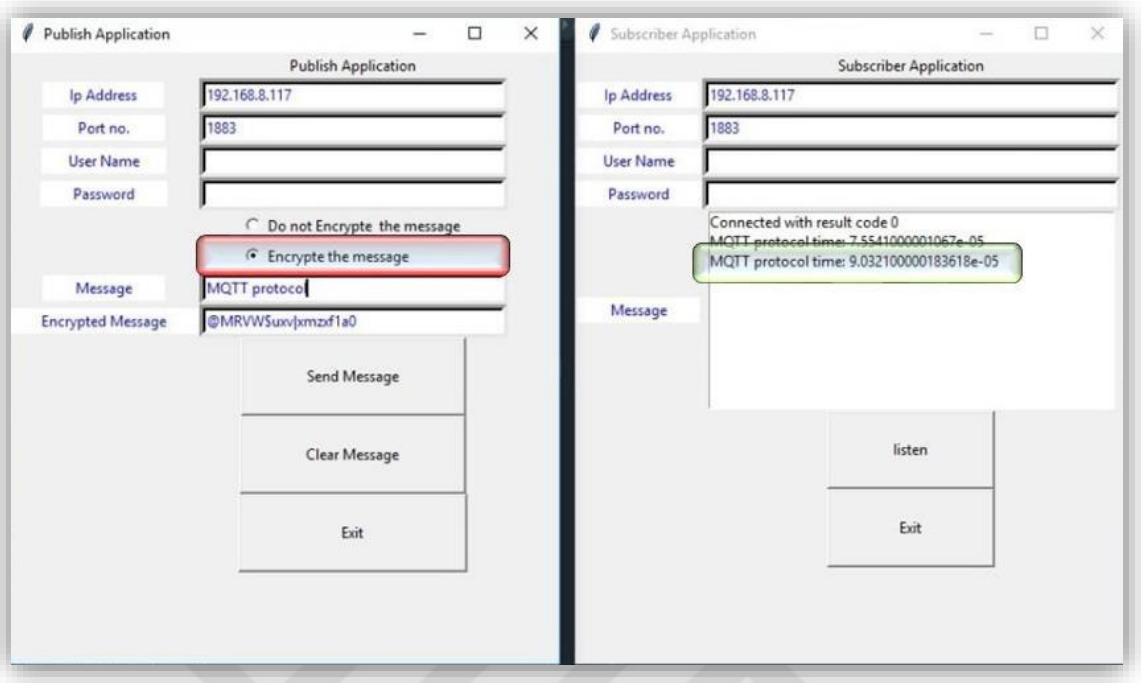


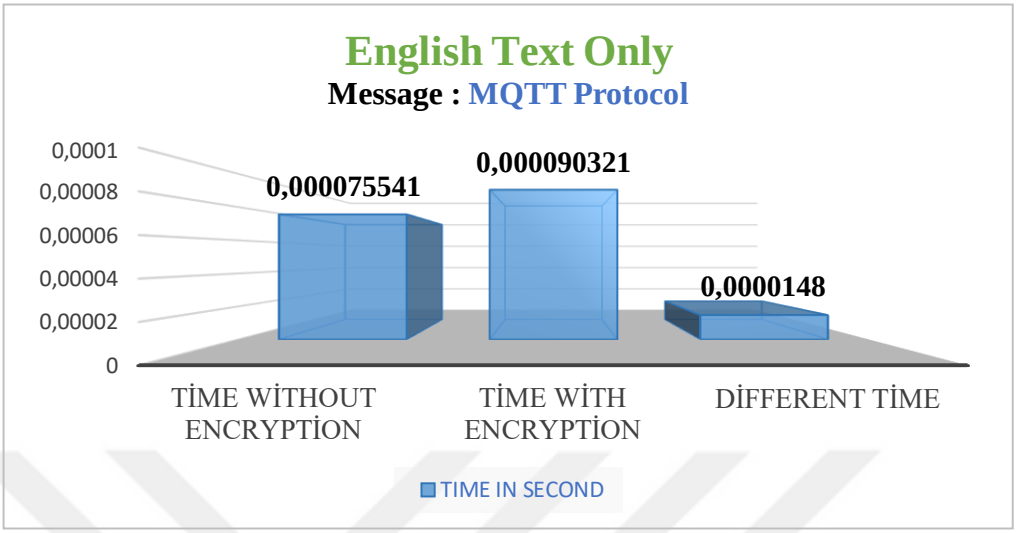
Figure 4.2: The Interface of the Proposed System to Sending Message (with Encryption) with Calculate the Received Time in Scenario (1).

Then Calculate the difference between the two times of the two cases above, as shown in Table 4.1 and Chart 4.1 .

Table 4.1: The Required Time in Second to Receive a Message (English Text only) in Scenario (1).

Message Type	Message	Time without encryption in second	Time with encryption in second	Different Time in second
English Text Only	**MQTT Protocol**	0.000075541	0.000090321	0.0000148000

Chart 4.1: The Required Time in Second to Receive a Message (English Text only) in Scenario (1).



4.2 SCENARIO 2

This scenario include sending a message (**Number only**) once without encryption and with encryption in addition to calculate the received time as follows:

a. Sending the message (**without Encryption**), as shown in Figure 4.3 .

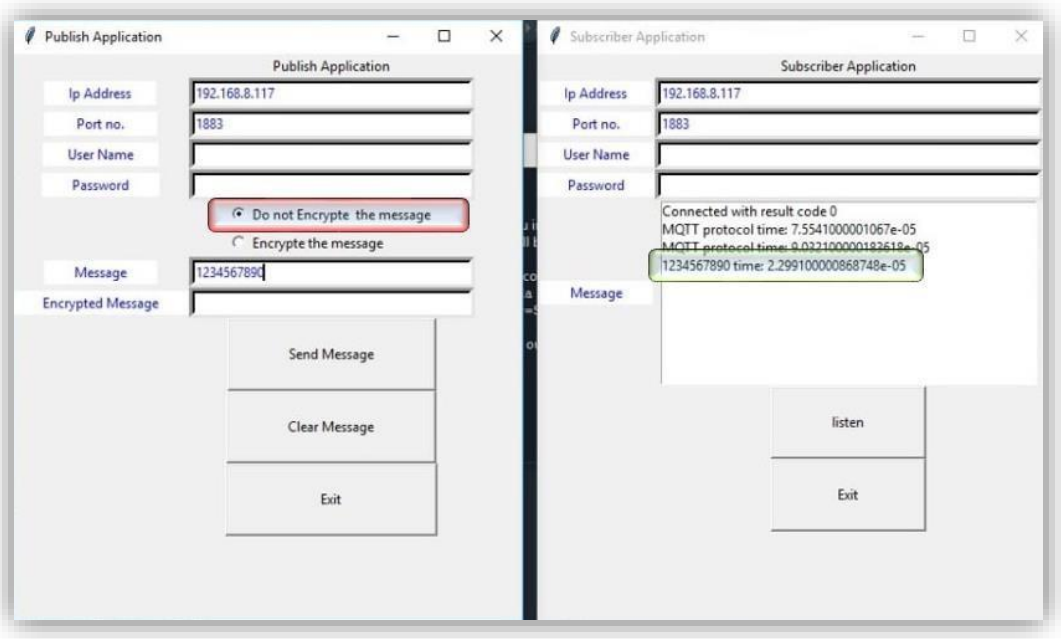


Figure 4.3: The Interface of the Proposed System to Sending Message (without Encryption) with Calculate the Received Time in Scenario (2).

b. Sending the message (**with Encryption**), as shown in Figure 4.4.

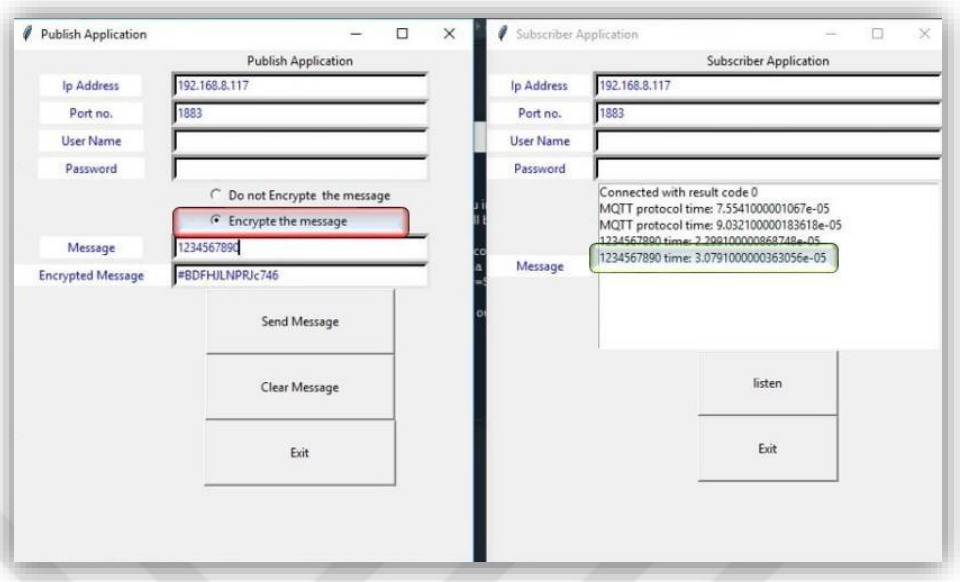


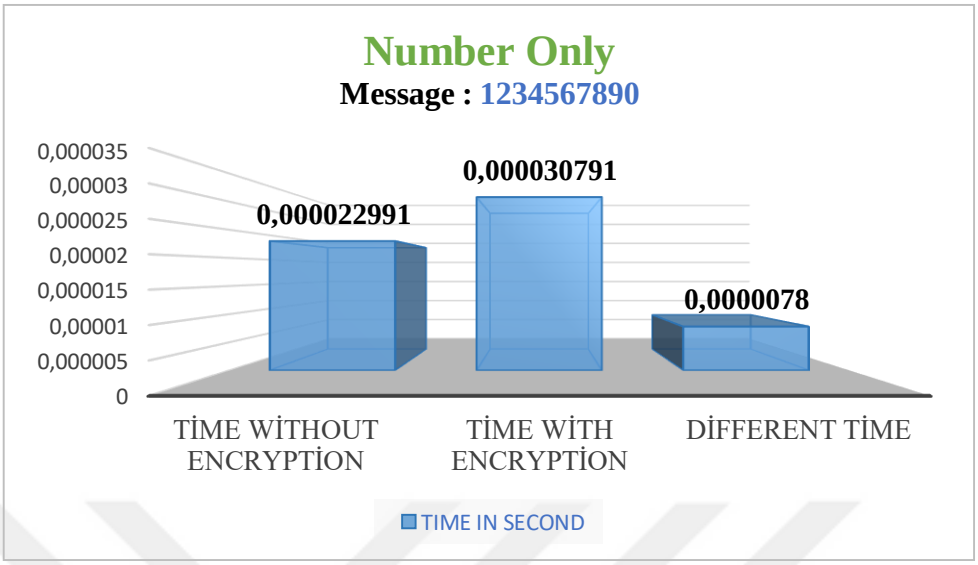
Figure 4.4: The Interface of the Proposed System to Sending Message (with Encryption) with Calculate the Received Time in Scenario (2).

Then Calculate the difference between the two times of the two cases above, as shown in Table 4.2 and Chart 4.2 .

Table 4.2: The Required Time in Second to Receive a Message (Number Only) in Scenario (2).

Message Type	Message	Time without encryption in second	Time with encryption in second	Different Time in second
Number Only	1234567890	0.000022991	0.000030791	0.0000078000

Chart 4.2: The Required Time in Second to Receive a Message (Number Only) in Scenario (2).



4.3 SCENARIO 3

This scenario include sending a message (**English Text with Number**) once without encryption and with encryption in addition to calculate the received time as follows:

a. Sending the message (**without Encryption**), as shown in Figure 4.5 .

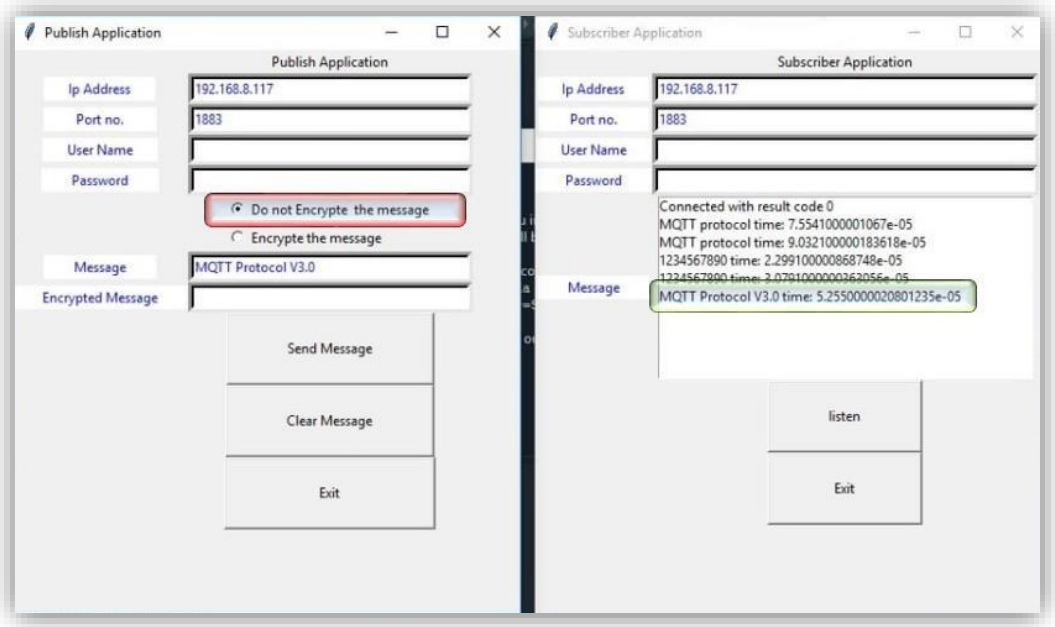


Figure 4.5: The Interface of the Proposed System to Sending Message (without Encryption) with Calculate the Received Time in Scenario (3).

b. Sending the Message (**with Encryption**), as Shown in Figure 4.6.

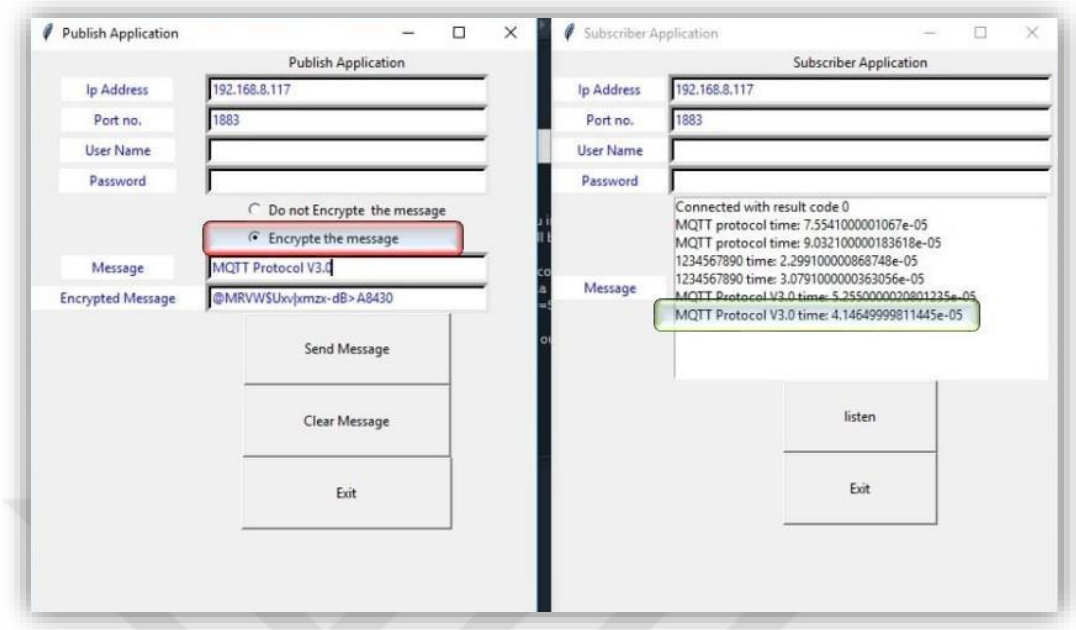


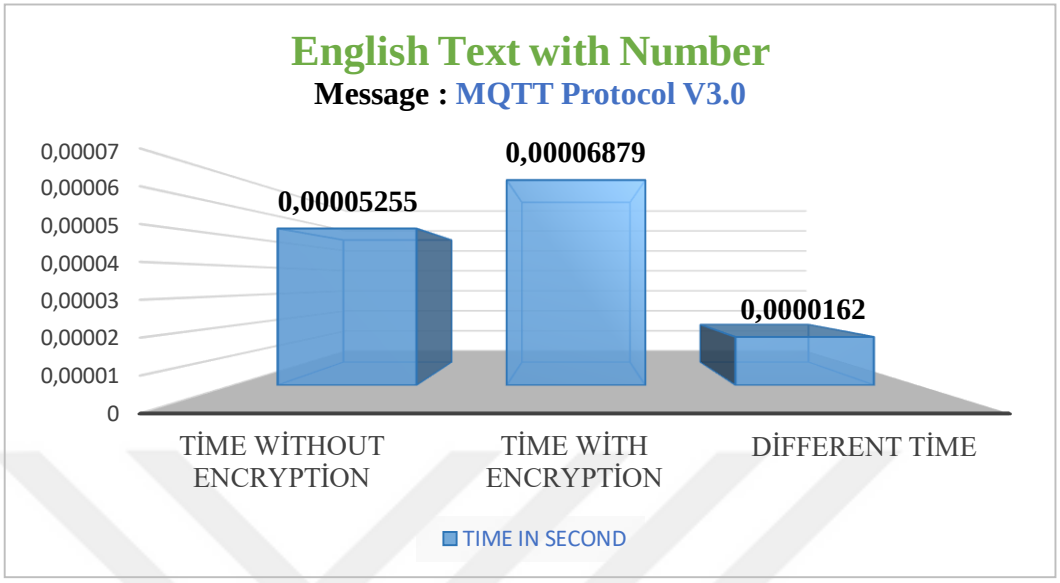
Figure 4.6: The Interface of the Proposed System to Sending Message (with Encryption) with Calculate the Received Time in Scenario (3).

Then Calculate the difference between the two times of the two cases above, as shown in Table 4.3 and Chart 4.3 .

Table 4.3: The Required Time in Second to Receive a Message (English Text with Number) in Scenario (3).

Message Type	Message	Time without encryption in second	Time with encryption in second	Different Time in second
English Text with Number	**MQTT Protocol V3.0**	0.000052550	0.000068790	0.0000162000

Chart 4.3: The Required Time in Second to Receive a Message (English Text with Number) in Scenario (3).



4.4 SCENARIO 4

This scenario include sending a message (**Arabic Text only**) once without encryption and with encryption in addition to calculate the received time as follows:

a. Sending the message (**without Encryption**), as shown in Figure 4.7 .

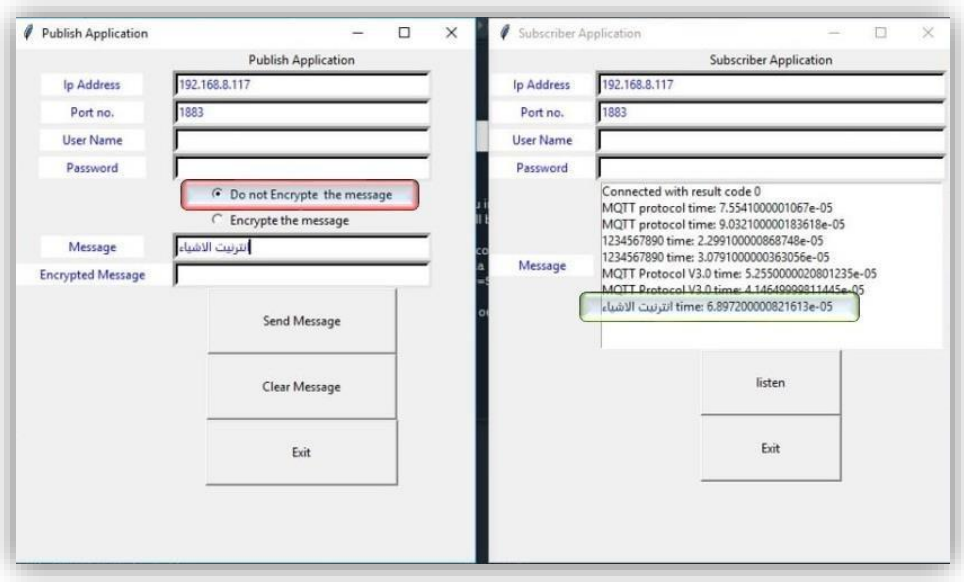


Figure 4.7: The Interface of the Proposed System to Sending Message (without Encryption) with Calculate the Received Time in Scenario (4).

b. Sending the message (**with Encryption**), as shown in Figure 4.8.

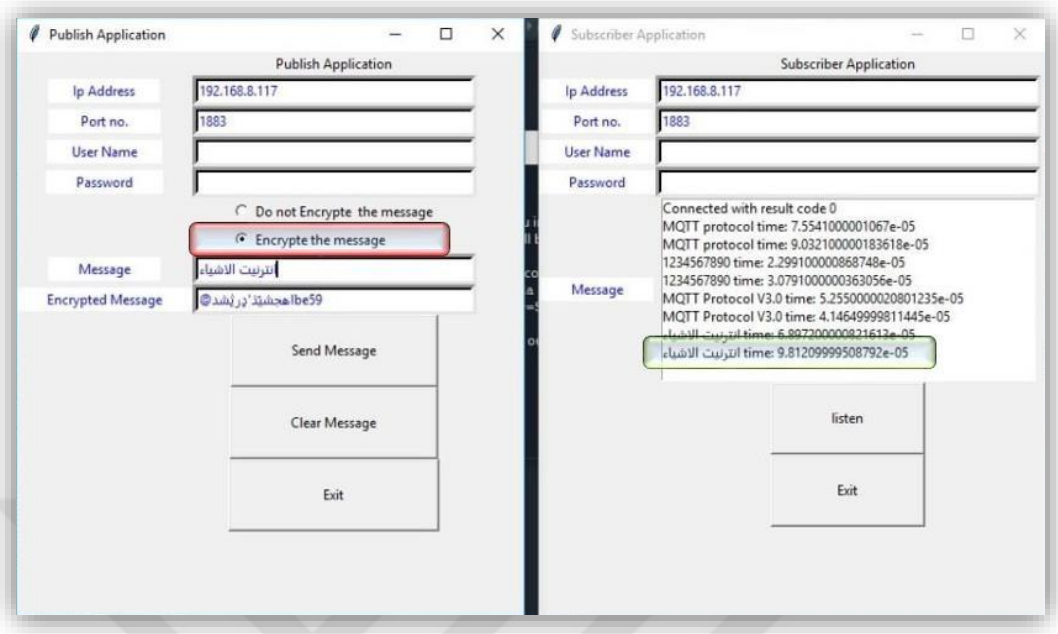


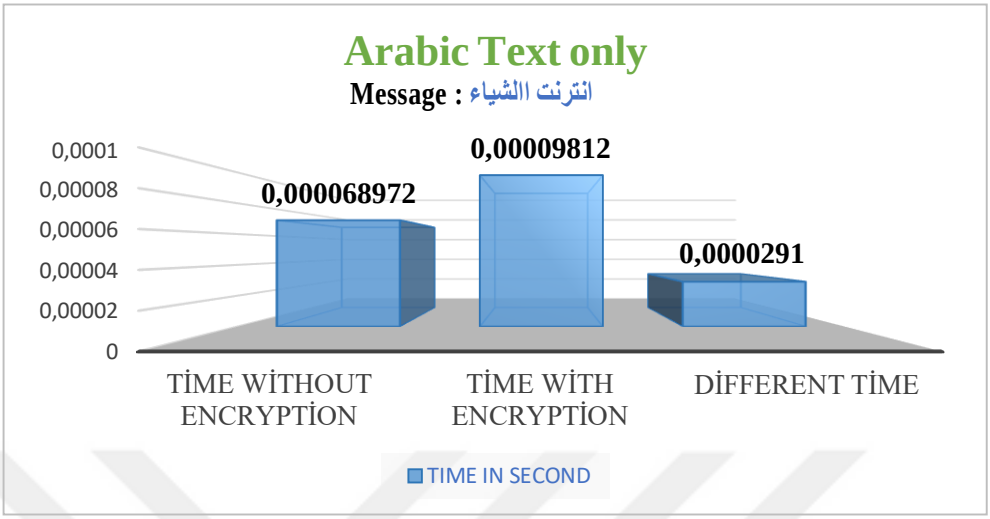
Figure 4.8: The Interface of the Proposed System to Sending Message (with Encryption) with Calculate the Received Time in Scenario (4).

Then Calculate the difference between the two times of the two cases above, as shown in Table 4.4 and Chart 4.4 .

Table 4.4: The Required Time in Second to Receive a Message (Arabic Text only) in Scenario (4).

Message Type	Message	Time without encryption in second	Time with encryption in second	Different Time in second
Arabic Text only	انترنيت الاشياء	0.000068972	0.000098120	0.000029100

Chart 4.4: The Required Time in Second to Receive a Message (Arabic Text only) in Scenario (4).



4.5 SCENARIO 5

This scenario include sending a message (**Arabic with English Text**) once without encryption and with encryption in addition to calculate the received time as follows:

1. Sending the message **(without Encryption), as sh**own in Figure 4.9 .

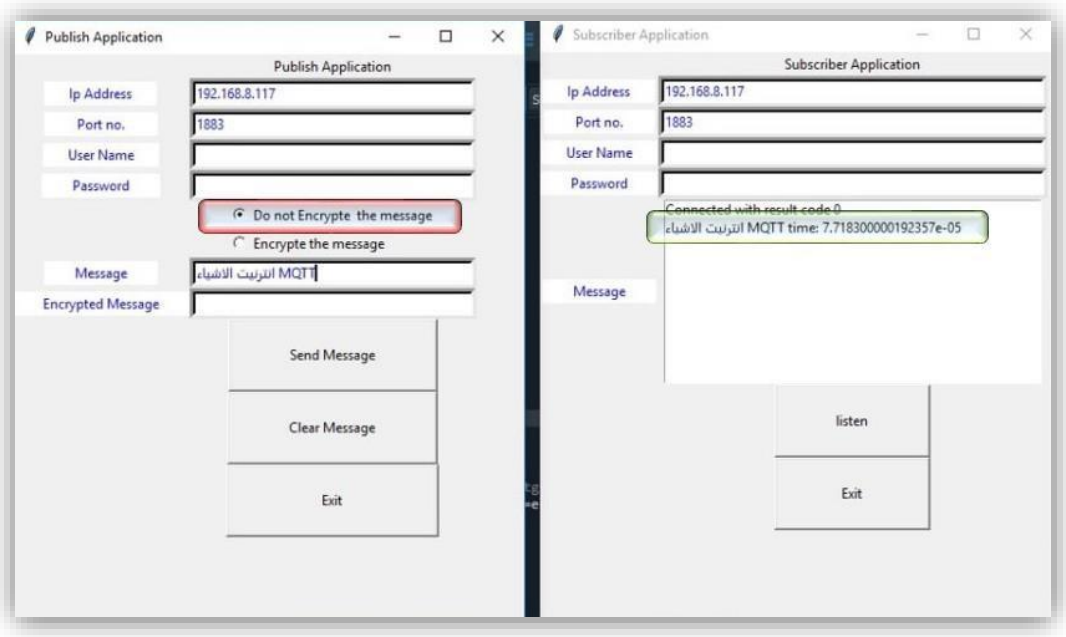


Figure 4.9: The Interface of the Proposed System to Sending Message (without Encryption) with Calculate the Received Time in Scenario (5).

2. Sending the message (**with Encryption**), as shown in Figure 4.10.

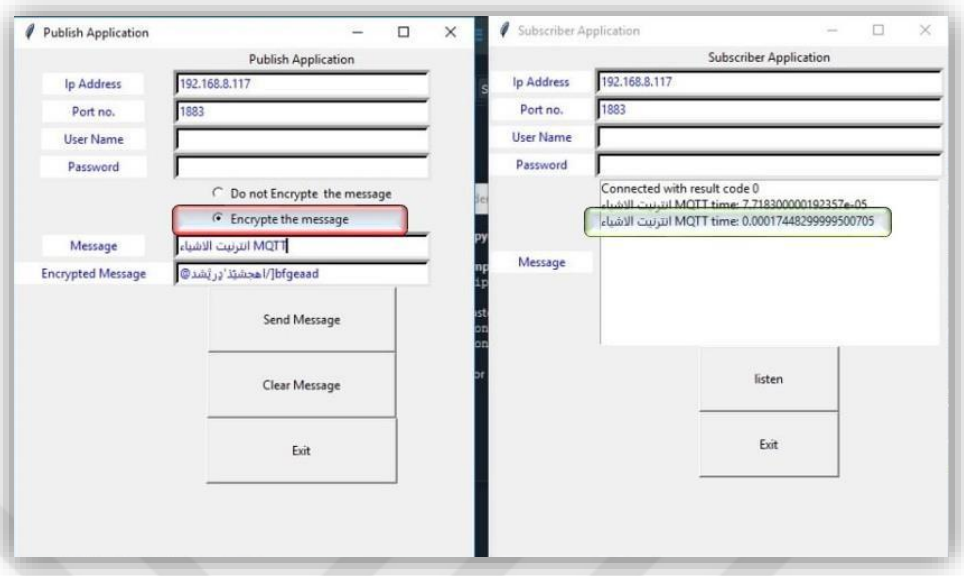


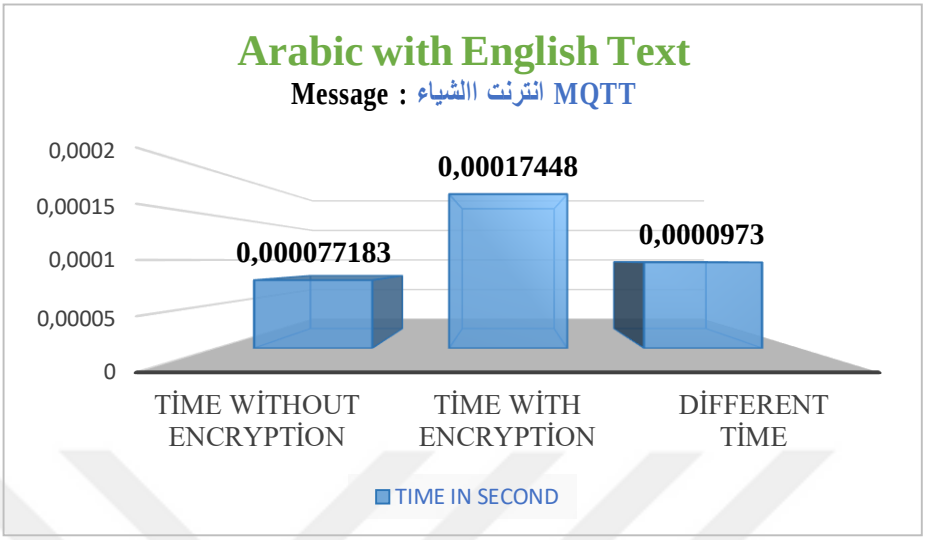
Figure 4.10: The Interface of the Proposed System to Sending Message (with Encryption) with Calculate the Received Time in Scenario (5).

Then Calculate the difference between the two times of the two cases above, as shown in Table 4.5 and Chart 4.5 .

Table 4.5: The Required Time in Second to Receive a Message (Arabic with English Text) in Scenario (5).

Message Type	Message	Time without encryption in second	Time with encryption in second	Different Time in second
Arabic with English Text	انترنيت االشياء MQTT	0.000077183	0.00017448	0.000097300

Chart 4.5: The Required Time in Second to Receive a Message (Arabic with English Text) in Scenario (5).



4.6 SCENARIO 6

This scenario include sending a message (**Arabic with English Text and Number**) once without encryption and with encryption in addition to calculate the received time as follows:

a. Sending the message (**without Encryption**), as shown in Figure 4.11 .

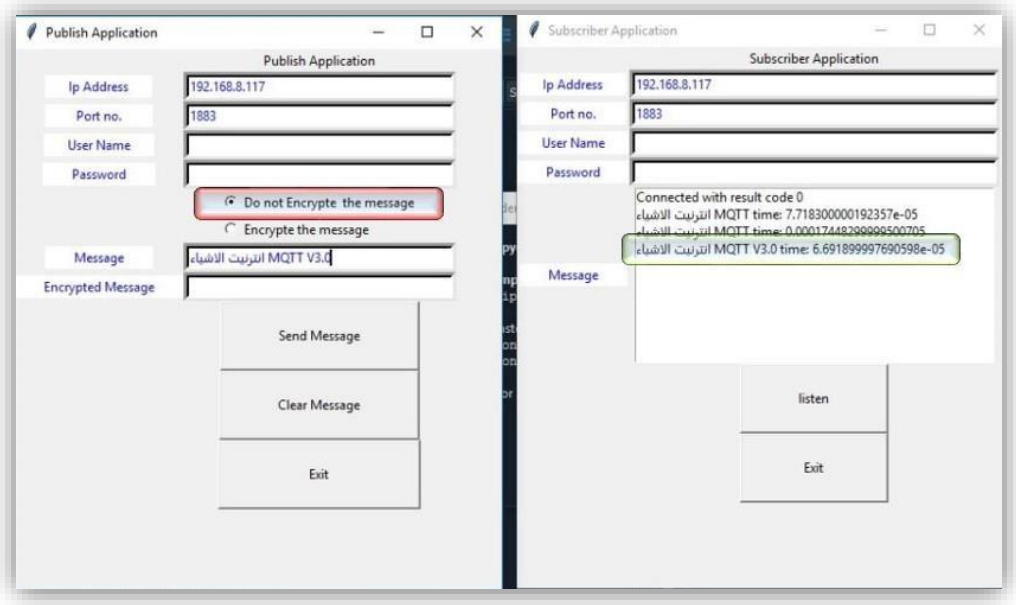


Figure 4.11: The Interface of the Proposed System to Sending Message (without Encryption) with Calculate the Received Time in Scenario (6).

b. Sending the message (**with Encryption**), as shown in Figure 4.12.

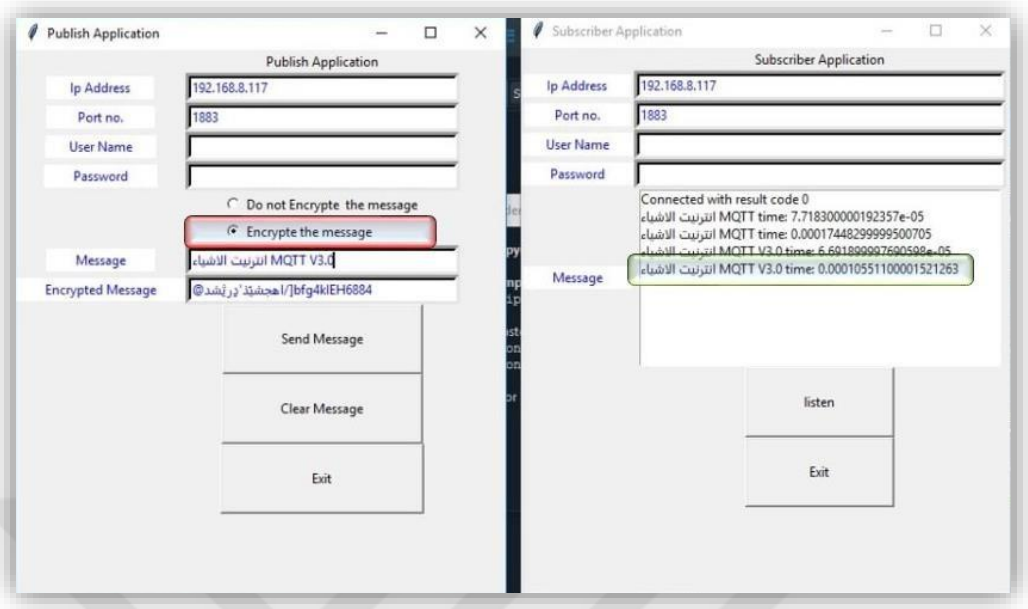


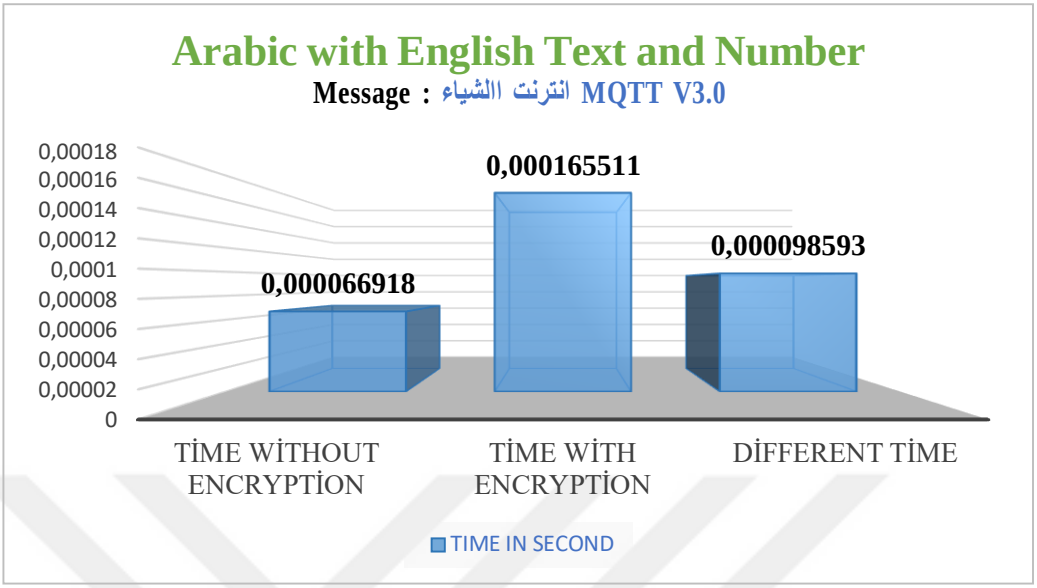
Figure 4.12: The Interface of the Proposed System to Sending Message (with Encryption) with Calculate the Received Time in Scenario (6).

Then Calculate the difference between the two times of the two cases above, as shown in Table 4.6 and Chart 4.6 .

Table 4.6: The Required Time in Second to Receive a Message (Arabic with English Text and Number) in Scenario (6).

Message Type	Message	Time without encryption in second	Time with encryption in second	Different Time in second
Arabic with English Text and Number	انترنيت االشياء MQTT V3.0	0.000066918	0.000165511	0.000098593

Chart 4.6: The Required Time in Second to Receive a Message (**Arabic with English Text and Number**) in Scenario (6).



All results' Scenarios of this thesis are summaries and show in the table 4.7 and chart 4.7.

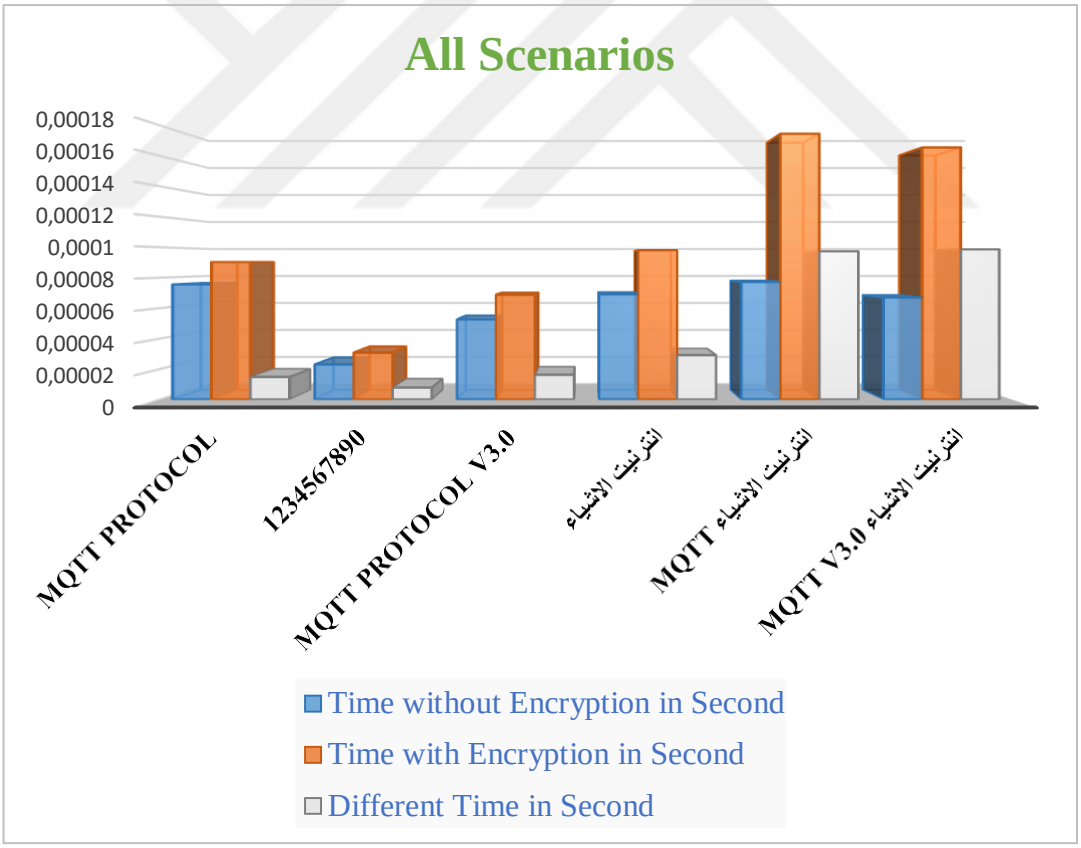
Table 4.7: The Results of the Proposed System in our Scenarios.

Scenario	Message Type	Message	Time without encryption in second	Time with encryption in second	Different Time in second
1	English Text Only	**MQTT Protocol**	0.000075541	0.000090321	0.0000148000
2	Number Only	**1234567890**	0.000022991	0.000030791	0.0000078000
3	English Text with Number	**MQTT Protocol V3.0**	0.000052550	0.000068790	0.0000162000

Table 4.7: The Results of the Proposed System in our Scenarios “Table Continued”.

4	**Arabic Text only**	ازترزيت االشرياء	**0.000068972**	**0.000098120**	**0.000029100**
5	Arabic with English Text	انترنيت الشياء MQTT	0.000077183	0.00017448	0.000097300
6	Arabic with English Text and Number	انترنيت الشياء MQTT V3.0	0.000066918	0.000165511	0.000098593

Chart 4.7: The Results of the Proposed System in our Scenarios.



5. DISCUSSIONS

As demonstrates in our Scenarios, we note that the pr0p0sed system does n0t significantly affect the perf0rmance 0f the Raspberry device operating in the Internet of Things environment. It does not need to distribute keys that used in encryption and decryption operations because it relies on the implicit key principle, which adds strength to its security, taking into account the optimal use of Internet of Things resources (Bandwidth, Memory, CPU, Energy).

In addition, the proposed system is strong against attempts to Crypto Analyzer attack, so that repeated letters in the plaintext do not take the same letters encrypted in the ciphertext, such as the encryption of the word (banana), which after encryption will became (bbpdrf).

Moreover, the ability of the proposed system to provide End-to-End Encryption (i.e. the process of encryption and integrity will protect data from the sending side to the receiving side).

Furthermore, the results in table 4.7 sh0w that pr0p0sed system in our Scenarios it is proven that time difference is very small between the message in its two states (with encrypti0n and with0ut encrypti0n), this indicates the possibility 0f applying the pr0p0sed system within the Internet 0f Things environment, which pr0vides security and data integrity services while maintaining accuracy and speed, in addition t0 the possibility 0f dealing with all types of texts and numbers.

It should be noted that the time calculation will vary depending on several Factors including:

a. The length of message.

b. Specifications of the sending and receiving devices and the Raspberry Pi.

c. Network speed.

6. CONCLUSION

Nowadays, the development and design of effective and lightweight encryption algorithms and techniques while taking into account the resource constraints of connected devices is now the primary security challenge in the IoT environment.

When utilizing the traditional security mechanisms in an IoT network, such as conventional Intrusion Detection System (IDS), encryption, access control, authentication, network protection, and application control, the processing of these mechanisms will be time-consuming and could cause problems for an extensive system with many connected heterogeneous devices.

As a conclusion, all of these works may offer an effective way to secure communication. They are not appropriate for restricted IoT devices due to several ongoing intrinsic restrictions in terms of scalability, efficiency and access control. As a result, there is an increasing need for effective lightweight encryption techniques that integrate both lightweight symmetric and asymmetric algorithm features. Additionally, in an IoT environment, these techniques ought to be able to offer mutual lightweight authentication for secure access control and authentication between users and devices. In this thesis, we will design and implement a lightweight encryption method which has the following characteristics (Key less, Encryption & Integrity, Text & Number End-to-End Encryption, Reduce Traffic and processing overhead).

The time taken for the encryption and decryption processes for the proposed encryption algorithm is very short compared to the time taken for classical encryption algorithms, in addition to the lack of need to distribute keys as they operate with an implicit key, which makes them ideal for working within the Internet of Things environment.

In order to provide the best Network Bandwidth used, 4 hexa decimal digit from HASH value were used instead of original 64 hexa decimal digit HASH value, which provide the data integration service.

REFERENCES

[1] L. Farhan, R. Kharel, O. Kaiwartya, M. Quiroz-Castellanos, A. Alissa, and M. Abdulsalam, "A Concise Review on Internet of Things (IoT) -Problems, Challenges and Opportunities," in *2018 11th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP)*, Jul. 2018, vol. 18-20 July, pp. 1–6, doi: 10.1109/CSNDSP.2018.8471762.

[2] D. Clark, "Characterizing Cyberspace: Past, Present, and Future," *ECIR Working Paper*, vol. 1, no. 2010. Massachusetts Institute of Technology, pp. 1–18, 2010, [Online]. Available: https://projects.csail.mit.edu/ecir/wiki/images/7/77/Clark_Characterizing_cyberspace_1-2r.pdf.

[3] N. Shetty, "Vulnerability Assessment for Cybersecurity using Machine learning," *SSRN Electron. J.*, 2021, doi: 10.2139/ssrn.3884044.

[4] D. Smith, T. Leonis, and S. Anandavalli, "Belonging and loneliness in cyberspace: impacts of social media on adolescents' well-being," *Aust. J. Psychol.*, vol. 73, no. 1, pp. 12–23, Jan. 2021, doi: 10.1080/00049530.2021.1898914.

[5] A. Zwitter and J. Hazenberg, "Cyberspace, Blockchain, Governance: How Technology Implies Normative Power and Regulation," in *Blockchain, Law and Governance*, Cham: Springer International Publishing, 2021, pp. 87–97.

[6] C. Venish Raja, K. Chitra, and M. Jonafark, "A Survey on Mobile Cloud Computing," *IJSRCSEIT - Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 3, no. 3, pp. 2096–2100, 2018, [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/wcm.1203.

[7] M. Ammar, G. Russello, and B. Crispo, "Internet of Things: A survey on the security of IoT frameworks," *J. Inf. Secur. Appl.*, vol. 38, pp. 8–27, Feb. 2018, doi: 10.1016/j.jisa.2017.11.002.

[8] Dongpo Zhang, "Big Data Security and Privacy Protection," in *2019 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)*, Sep. 2019, pp. 87–

89, doi: 10.1109/ICVRIS.2019.00030.

[9] Mahmood Subhy Mahmood Saeed, “Lightweight Cyber Attack Intelligent Detection Model Based on Blockchain in IoT,” University of Mosul in iraq, 2023.

[10] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, “Machine Learning in IoT Security: Current Solutions and Future Challenges,” *IEEE Commun. Surv. Tutorials*, vol. 22, no. 3, pp. 1686–1721, Mar. 2019, doi: 10.1109/COMST.2020.2986444.

[11] I. Cvitić, D. Peraković, M. Periša, and M. Botica, “Smart Home IoT Traffic Characteristics as a Basis for DDoS Traffic Detection,” in *Proceedings of the 3rd EAI International Conference on Management of Manufacturing Systems*, 2018, 3rd ed., doi: 10.4108/eai.6-11-2018.2279336.

[12] I. Vaccari, M. Aiello, and E. Cambiaso, “SlowTT: A Slow Denial of Service against IoT Networks,” *Information*, vol. 11, no. 9, p. 452, Sep. 2020, doi: 10.3390/info11090452.

[13] M. A. Khan *et al.*, “A Deep Learning-Based Intrusion Detection System for MQTT Enabled IoT,” *Sensors*, vol. 21, no. 21, p. 7016, Oct. 2021, doi: 10.3390/s21217016.

[14] F. Carpio, A. Jukan, and X. Masip-bruin, “A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration,” *ACM journals - ACM Comput. Surv.*, vol. 27 Feb 201, no. 1, pp. 1–30, 2019, [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3292674.

[15] P. Tuwanut and S. Kraijak, “A survey on IoT architectures, protocols, applications, security, privacy, real-world implementation and future trends,” in *11th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2015)*, 2015, pp. 6 .-6 ., doi: 10.1049/cp.2015.0714.

[16] V. Dankan Gowda, S. B. Sridhara, K. B. Naveen, M. Ramesha, and G. Naveena Pai, “INTERNET OF THINGS: INTERNET REVOLUTION, IMPACT, TECHNOLOGY ROAD MAP AND FEATURES,” *Adv. Math. Sci. J.*, vol. 9, no. 7, pp. 4405–4414, Jul. 2020, doi: 10.37418/amsj.9.7.11.

[17] S. Fenanir, F. Semchedine, and A. Baadache, “A Machine Learning-Based

Lightweight Intrusion Detection System for the Internet of Things," *Rev. d'Intelligence Artif.*, vol. 33, no. 3, pp. 203–211, Oct. 2019, doi: 10.18280/ria.330306.

[18] E. Al-Masri *et al.*, "Investigating Messaging Protocols for the Internet of Things (IoT)," *IEEE Access*, vol. 8, pp. 94880–94911, 2020, doi: 10.1109/ACCESS.2020.2993363.

[19] A. Cornel - Cristian, T. Gabriel, M. Arhip-Calin, and A. Zamfirescu, "Smart home automation with MQTT," in *2019 54th International Universities Power Engineering Conference (UPEC)*, Sep. 2019, vol. 19, pp. 1–5, doi: 10.1109/UPEC.2019.8893617.

[20] S. V. Mukherji, R. Sinha, S. Basak, and S. P. Kar, "Smart Agriculture using Internet of Things and MQTT Protocol," in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, Feb. 2019, pp. 14–16, doi: 10.1109/COMITCon.2019.8862233.

[21] R. A. Atmoko and D. Yang, "Online Monitoring & Controlling Industrial Arm Robot Using MQTT Protocol," in *2018 IEEE International Conference on Robotics, Biomimetics, and Intelligent Computational Systems (Robionetics)*, Aug. 2018, pp. 12–16, doi: 10.1109/ROBIONETICS.2018.8674672.

[22] A. J. Hintaw, S. Manickam, S. Karuppayah, M. A. Aladaileh, M. F. Aboalmaaly, and S. U. A. Laghari, "A Robust Security Scheme Based on Enhanced Symmetric Algorithm for MQTT in the Internet of Things," *IEEE Access*, vol. 11, pp. 43019–43040, 2023, doi: 10.1109/ACCESS.2023.3267718.

[23] A. Ghannadrad, "Machine leaning-based DoS attacks detection for MQTT sensor networks," Politecnico di Milano, 2021.

[24] S. Medileh *et al.*, "A flexible encryption technique for the internet of things environment," *Ad Hoc Networks*, vol. 106, p. 102240, Sep. 2020, doi: 10.1016/j.adhoc.2020.102240.

[25] X. Yao, Z. Chen, and Y. Tian, "A lightweight attribute-based encryption scheme for the Internet of Things," *Futur. Gener. Comput. Syst.*, vol. 49, pp. 104–112, Aug. 2015, doi: 10.1016/j.future.2014.10.010.

[26] M. Usman, I. Ahmed, M. Imran, S. Khan, and U. Ali, “SIT: A Lightweight Encryption Algorithm for Secure Internet of Things,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 1, pp. 1–10, 2017, doi: 10.14569/IJACSA.2017.080151.

[27] B.-Y. Sung, K.-B. Kim, and K.-W. Shin, “An AES-GCM authenticated encryption crypto-core for IoT security,” in *2018 International Conference on Electronics, Information, and Communication (ICEIC)*, Jan. 2018, vol. 2018-Janua, pp. 1–3, doi: 10.23919/ELINFOCOM.2018.8330586.

[28] F. Hazzaa, “New Security Scheme and Lightweight Encryption Algorithm for Voice Over Wireless Networks Connectivity To Internet,” in *Anglia Ruskin University*, 2019, no. May, pp. 1–248, [Online]. Available: https://arro.anglia.ac.uk/id/eprint/705272.

[29] T. Z. Ananth Vishnu Bhaskar, Ankit Baingane, Ryan Jahnige, Qingquan Zhang, “A Secured Protocol for IoT Networks,” in *Cornell University*, 2020, pp. 1–5, doi: 10.48550/arXiv.2012.11072.

[30] Y. M. Khattabi, M. M. Matalgah, and M. M. Olama, “Revisiting Lightweight Encryption for IoT Applications: Error Performance and Throughput in Wireless Fading Channels with and without Coding,” *IEEE Access*, vol. 8, pp. 13429–13443, 2020, doi: 10.1109/ACCESS.2020.2966596.

[31] M. H. Saracevic *et al.*, “Data Encryption for Internet of Things Applications Based on Catalan Objects and Two Combinatorial Structures,” *IEEE Trans. Reliab.*, vol. 70, no. 2, pp. 819–830, 2021, doi: 10.1109/TR.2020.3010973.

[32] B. Mishra and A. Kertesz, “The Use of MQTT in M2M and IoT Systems: A Survey,” *IEEE Access*, vol. 8, pp. 201071–201086, 2020, doi: 10.1109/ACCESS.2020.3035849.

[33] P. Fehrenbach, “Messaging Queues in the IoT under pressure,” *Comput. Sci. Its Appl. ICCSA*, pp. 1–9, 2018, [Online]. Available: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://blog.it-securityguard.com/wp-content/uploads/2017/10/IOT_Mosquitto_Pfehrenbach.pdf.

[34] S. Bandyopadhyay and A. Bhattacharyya, “Lightweight Internet protocols for web enablement of sensors using constrained gateway devices,” in *2013 International Conference on Computing, Networking and Communications (ICNC)*, Jan. 2013, pp. 334–340, doi: 10.1109/ICCNC.2013.6504105.

[35] L. Anne, T. Nandan, M. Kunj, S. A. Kumar, G. Mahesh, and R. Sangeetha, “MQTT-Based Android Chat Application for IoT,” *SN Comput. Sci.*, vol. 3, no. 5, p. 402, Jul. 2022, doi: 10.1007/s42979-022-01278-8.

[36] D. Soni and A. Makwana, “A survey on mqtt: a protocol of internet of things(IoT),” *International Conference on Telecommunication, Power Analysis and Computing Techniques (Ictpact - 2017)*, no. April. pp. 0–5, 2017, [Online]. Available: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.researchgate.net/profile/Dipa-Soni/publication/316018571_A_SURVEY_ON_MQTT_A_PROTOCOL_OF_INTERNET_OF_THINGSIOT/links/58edafd4aca2724f0a26e0bf/A-SURVEY-ON-MQTT-A-PROTOCOL-OF-INTERNET-OF-THINGSI.

[37] S. M. Sakina ELHADI, Abdelaziz Marzak, Nawal SAEL, “Comparative Study of IoT Protocols,” *SSRN Electron. J.*, vol. february 2, 2018, [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3186315.

[38] N. S. Han, “Semantic service provisioning for 6LoWPAN : powering internet of things applications on Web,” ET L’UNIVERSIT´E PIERRE ET MARIE CURIE - PARIS, 2015.

[39] R. A. Light, “Mosquitto: server and client implementation of the MQTT protocol,” *J. Open Source Softw.*, vol. 2, no. 13, p. 265, May 2017, doi: 10.21105/joss.00265.

[40] F. Ouakasse and S. Rakrak, “A comparative study of MQTT and COAP application layer protocols via. performances evaluation,” *Journal of Engineering and Applied Sciences*, vol. 13, no. 15. pp. 6053–6061, 2018, doi: 10.3923/jeasci.2018.6053.6061.

[41] D. Dinculeană and X. Cheng, “Vulnerabilities and Limitations of MQTT Protocol Used between IoT Devices,” *Appl. Sci.*, vol. 9, no. 5, p. 848, Feb. 2019, doi: 10.3390/app9050848.

[42] M. Yue, Y. Ruiyang, S. Jianwei, and Y. Kaifeng, “A MQTT Protocol Message Push Server Based on RocketMQ,” in *2017 10th International Conference on Intelligent Computation Technology and Automation (ICICTA)*, Oct. 2017, vol. 2017-Octob, pp. 295–298, doi: 10.1109/ICICTA.2017.72.

[43] D. Matabuena, F. J. Bellido-Outeirino, A. Moreno-Munoz, A. Gil-de-Castro, and J. M. Flores-Arias, “Educational platform for communications using the MQTT protocol,” in *2018 XIII Technologies Applied to Electronics Teaching Conference (TAEE)*, Jun. 2018, pp. 1–6, doi: 10.1109/TAEE.2018.8475972.

[44] R. K. Kodali, B. Yatish Krishna Yogi, G. N. Sharan Sai, and J. Honey Domma, “Implementation of Home Automation Using CoAP,” in *TENCON 2018 - 2018 IEEE Region 10 Conference*, Oct. 2018, vol. 2018-Octob, pp. 1214–1218, doi: 10.1109/TENCON.2018.8650135.

[45] B. Oryema, H.-S. Kim, W. Li, and J. T. Park, “Design and implementation of an interoperable messaging system for IoT healthcare services,” in *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Jan. 2017, pp. 45–52, doi: 10.1109/CCNC.2017.7983080.

[46] N. Naik, “Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP,” in *2017 IEEE International Systems Engineering Symposium (ISSE)*, Oct. 2017, pp. 1–7, doi: 10.1109/SysEng.2017.8088251.

[47] E. Bertrand-Martinez, P. Dias Feio, V. de Brito Nascimento, F. Kon, and A. Abelém, “Classification and evaluation of IoT brokers: A methodology,” *Int. J. Netw. Manag.*, vol. 31, no. 3, May 2021, doi: 10.1002/nem.2115.

[48] M. S. Turan, E. Barker, W. Burr, and L. Chen, “Recommendation for password-based key derivation: part 1: storage applications,” *NIST Special Publication*, no. December. pp. 800–132, 2010, [Online]. Available: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf.

[49] M. N. Alenezi, H. Alabdulrazzaq, and N. Q. Mohammad, “Symmetric encryption algorithms: Review and evaluation study,” *International Journal of Communication Networks and Information Security*, vol. 12, no. 2. IJCNIS : International Journal of

Communication Networks and Information Security, pp. 256–272, 2020, [Online]. Available: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.researchgate.net/profile/Haneen-Alabdulrazzaq/publication/349324592_Symmetric_Encryption_Algorithms_Review_and_Evaluation_study/links/602acfa7a6fdcc37a82c0189/Symmetric-Encryption-Algorithms-.

[50] D. Kurniawan, Suparti, and Sugito, "Cryptography and Network Security Principles and Practices," *J. Phys. Conf. Ser.*, vol. 1025, no. 1, p. 592, May 2018, doi: 10.1088/1742-6596/1025/1/012111.

[51] D. Rolon-Mérette, M. Ross, T. Rolon-Mérette, and K. Church, "Introduction to Anaconda and Python: Installation and setup," *Quant. Methods Psychol.*, vol. 16, no. 5, pp. S3–S11, May 2020, doi: 10.20982/tqmp.16.5.S003.

[52] S. Kumar P., R. Kumudham, D. R. Kumar, M. Dhamodharan, and S. Vetrivel, "Smart home automation using Raspberry PI 4," *AIP Conference Proceedings*, vol. 2463. 2022, doi: 10.1063/5.0080751.

[53] David Crookes *et al.*, *the Official Raspberry Pi Handbook 2021*, 1st ed. Russell Barnes, 2021.

[54] D. Crookes *et al.*, *the Official Raspberry Pi Handbook 2022*. Russell Barnes, 2022.

[55] W. Donat, *Learn Raspberry Pi Programming with Python*, Second. Berkeley, CA: Apress Berkeley, CA, 2014.

[56] J. Ridler, "VNC Viewer (Virtual Network Computing)," *Food Manufacture*, vol. 2019, no. August. 2019, Accessed: Oct. 13, 2023. [Online]. Available: https://discover.realvnc.com/what-is-vnc-remote-access-technology.

[57] Putty, "PuTTY FAQ," 2016. https://www.chiark.greenend.org.uk/~sgtatham/putty/faq.html#faq-pronounce (accessed Oct. 03, 2023).