Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Information Technology
Department of Information Network

Improving IoT Security using Data Compression and Lightweight Cryptographic Technique

A Thesis

Submitted to the Council of the College of Information Technology for Postgraduate Studies of University of Babylon in Partial Fulfillment of the Requirements for the Degree of Master in Information Technology / Information Networks.

By

Ahmed Najah Kadhim Kareem

Supervised by

Asst. Prof.Dr. Mehdi Ebady Manaa Mehdi

2022 A.D. **1444 A.H**

يَرفعُ الله الذينَ امنوا مِنكُم والذينَ
اوتوا الِعلم دَرجات

صدق اللَّهُ العظيم

سورة المجادلة /الآية 11

Supervisor Certification

I certify that this thesis entitled (**IMPROVING IOT SECURITY USING DATA COMPRESSION AND LIGHTWEIGHT CRYPTOGRAPHIC TECHNIQUE**) was prepared under my supervision at the department of Information Networks / College of Information Technology / University of Babylon as partial fulfillment of the requirements for the degree of Master in Information Technology-Information Networks.

Signature:

Supervisor Name: **Asst. Prof.Dr. Mehdi Ebady Manaa**

Date: / /2022

The Head of the Department Certification

In view of the available recommendations, I forward the thesis entitled "**IMPROVING IOT SECURITY USING DATA COMPRESSION AND LIGHTWEIGHT CRYPTOGRAPHIC TECHNIQUE"** for debate by the examination committee.

Signature:

Prof. Dr. **Saad Talib Hasson Aljebori**

Head of Information Networks Department

Date: / /2022

Declaration

I hereby declare that this Dissertation, submitted to the University of Babylon in partial fulfillment of the requirements for the degree of Master of Information Technology-Information Networks has not been submitted as an exercise for a similar degree at any other university. I also certify that this work described here is entirely my own except for experts and summaries whose sources are appropriately cited in the references.

Signature:

Name: **Ahmed Najah Kadhim**

Date: **/ / 2022**

Dedication

I dedicate this thesis

To the big heart, my father

To the fountain of patience my mother

To my greatest lovely, my wife

To my sweetest sons

To my brother and my sisters

To my supervisor

To my family

To my friends

Researcher

ACKNOWLEDGMENT

In the name of God, Most Gracious, Most Merciful, at first, Praise be to God and thanks to God and the satisfaction of parents and conciliation only from God greatest praise is to **Allah** for His assistance in facing the difficulty that I met in my study, and for always helping me to achieve my aims, also for His great graces and boons all the time.

I would like to express my deepest thanks to my supervisor **Asst. Prof.Dr. Mehdi Ebady Manaa** for his valuable advice, motivation, guidance, and so many fruitful discussions throughout the preparation of this thesis.

I would like to extend my respect and deepest gratitude to the College of Information Technology.

Sincere appreciation and love go to my family they provide me with optimism and pure affection and they give me great hope, encouragement and they have stood with me in every step of this research. I dedicate this work and give special thanks to those who encouraged me to continue my scientific career, my wife, and my children.

Finally, Sincere thanks and appreciation to all friends, colleagues, and loved ones

Researcher

ABSTRACT

The Internet of Things (IoT) can be considered as a physical object that connects and exchanges data with other objects over networks. The recent emergence of the IoT has led to a lot of attention being focused on it. The IoT device generates a huge amount of stream data which affects the connection bandwidth rate. Due to the massive volume of IoT data, requires techniques to deal with this limitation, such as compression algorithms to decrease data size, save network bandwidth, and increase the speed of IoT data transmission.

The IoT environment transmits different types of personal healthcare information, critical data, and other items via the Internet. This data becomes a target of security issues. Therefore, the security of this data is vital to maintain the confidentiality and integrity of the IoT data. Due to the limited resources of IoT devices, they require special encryption algorithms such as lightweight cryptography algorithms (LWC).

The proposed system aims to overcome the massive volume of data and security problems in the Internet of Things. By combining a data compression algorithm and lightweight cryptographic techniques called the Compcrypt approach. It used the Zstandard algorithm for the compression of IoT sensor data, while it used the Tiny Encryption Algorithm (TEA) for the encryption of compressed data with a shared secret key that was generated by using the Elliptic Curve Diffie Hellman protocol on the Client side. The decompression and decryption processes are done on the server side with the same algorithms.

The proposed system is based on hardware devices, such as the temperature sensor, SPO_2 (heart rate and oxygen level sensor), and the Raspberry Pi model 4. The results of the proposed Compcrypt system showed a better in terms of compression ratio is 81%, encryption time is 108.5 ms, and throughput is 1099.82

Kbps for a data size is 118.78 Kb. The security strength evaluation of the proposed Compcrypt system is based on entropy as 7.998, and the avalanche effect as 94% for the 128-byte size of cipher text.

List of Thesis-Related Publications

(First Paper)

- **Name of Journal: Bulletin of Electrical Engineering and Informatics.**
- **Paper Title: Design an efficient internet of things data compression for healthcare applications.**
- **Authors:**
  - Ahmed Najah Kahdim
  - Asst. Prof.Dr. Mehdi Ebady Manaa

Information Network Department, Information Technology College,
Babylon University.

DOI: https://doi.org/10.11591/eei.v11i3.3758

Email: tech.ahmed2012@gamil.com

(Second Paper)

- **Name of Conference: Fifth International Iraqi Conference on Engineering Technology and its Applications (5th IICETA).**
- **Paper Title: Improving IoT data Security Using Compression and Lightweight Encryption Technique.**
- **Authors:**
  - Ahmed Najah Kahdim
  - Asst. Prof.Dr. Mehdi Ebady Manaa

Information Network Department, Information Technology College,
Babylon University.

DOI: 10.1109/IICETA54559.2022.9888376

Email: ahmedn.net.msc@student.uobabylon.edu.iq

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

List of Abbreviations

Abbreviation	Description
AE	Avalanche Effect
AES	Advanced Encryption Standard
CPU	Central Processing Unit
DCR	Data Compression Ratio
DES	Data Encryption Standard
DWT	Discrete Wavelet Transform
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie Hellman
FNS	Feistel Network Structure
GE	Gate Equivalent
IoT	Internet of Things
LAES	Lightweight AES
LSC	Lightweight stream cipher
LWC	Lightweight Cryptography Algorithms
MTU	Maximum Transmission Units
NFC	Near Field Communication
NIST	National Institute of Standards and Technology
PGP	Pretty Good Privacy
RAM	Random Access Memory
RFID	Radio Frequency Identification
ROM	Read Only Memory
RSA	Rivest Shamir Adleman
SHA	Secure Hash Algorithm
TEA	Tiny Encryption Algorithm
WSN	Wireless Sensor Network
AE	Avalanche Effect
CPU	central processing unit

Chapter One

Introduction

CHAPTER ONE
INTRODUCTION

1.1 Introduction

The rapid developments in communication, computing, and wireless network have led to the emergence of the Internet of Things (IoT). Each physical item may be connected to a virtual one using IoT technology. Different technologies such as Wireless Sensor Networks (WSN), Radio Frequency Identification (RFID), and other machine-to-machine interfaces, cloud services, and so on, may be used to create this relationship, such as a mix of these technologies [1]. The IoT may be found in a variety of applications in everyday life, including smart home equipment (smart lock, indoor security camera, smart thermostat) as well as health monitoring items (Personal EKG, Portable Gluten Tester, Health-Oriented Smart Watch, Blood Pressure Monitor), among others. These intelligent gadgets integrate several applications (often without direct human intervention). The Internet of Things devices are referred to as resource-constrained devices due to their restricted resources (processor, memory, and storage) and power consumption, especially when powered by a battery [2].

The main components of the IoT healthcare architecture include a body sensor network, gateways, and data centers as shown in figure (1.1) [3].

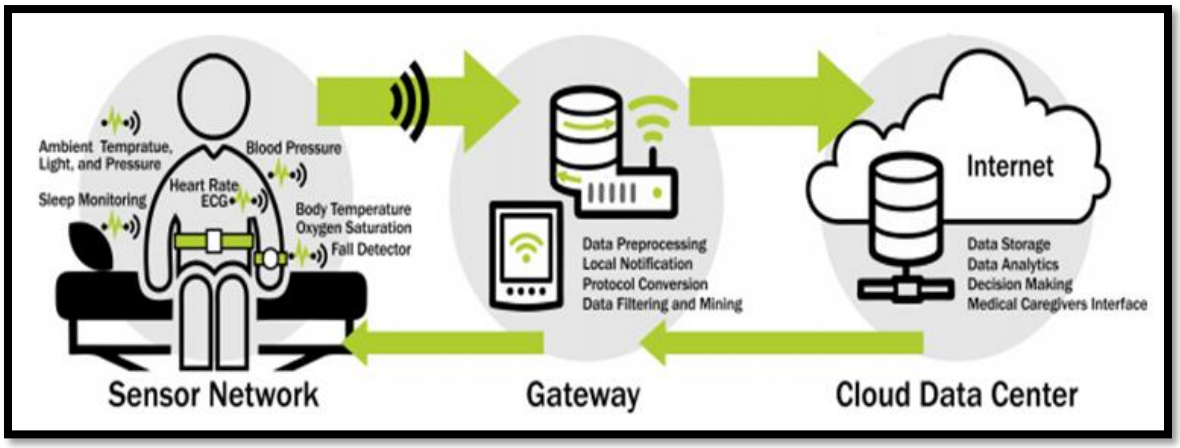

Figure 1.1: General IoT healthcare architecture [3].

According to several studies, the IoTs have been produced 4.4 trillion gigabytes of data in 2020, and this number is projected to grow in the next years. Despite the IoT's intriguing expansion and the introduction of possible services that will be provided to enhance human life, it faces several challenges. Additionally, billions of smart gadgets running on several platforms are linked to the Internet. As a result, the Internet will contain an ever-increasing quantity of data [4]. This data may confront several new and growing difficulties, including administrative complexity, privacy, sending, lifetime, support technologies, and security assaults such as eavesdropping and modification. To design a good system, there are several needs and traits that must be met, such as data compression, security, privacy, and integrity [5].

Data compression is a crucial part of digital communication today. It is defined as reducing the amount of data, which makes it faster to send the data. Also, it is encoding the data by using fewer bits than the original data which achieves a higher level of security. In the decompression process, the original data are put back together with as few steps as possible. Different

compression algorithms can be used to get different compression ratios. Lossless compression and lossy compression are two different classifications of data compression. When we compress data without losing any information in the process, we refer to this as lossless compression. This allows us to get almost the same amount of data after decompressing it as we did before we compressed it. When data is compressed using a technique known as loss compression, some (often less important) of the data's characteristics may be lost as a result of the compression process [6].

The Internet of Things has become an essential aspect of today's digital environment. It handles a huge amount of information. In the case of the Internet of Things, it has been found that some data is redundant, while others are not. Data that cannot be tampered with in any way must be protected using proper measures so that it can be duplicated at the sink end without any alterations. Lossless compression is the best option in these situations. The Internet of Things networks is resource constrained and do not have sufficient bandwidth. Since of this restriction, data compression is required in many situations because the data rates and bandwidth that are available for IoTs are limited [7].

IoT devices and smart things are appealing targets for attackers because they have direct links to the real world. An attacker may spy on a large enterprise by hacking into any of the smart items that are being utilized there. A real and important example of direct communication with the actual world, the risk that is faced by healthcare organizations such as hospitals. These are frequently associated with a patient records and various pieces of medical equipment, such as MRI, X-ray machines, Thermometers, oxygen,

and heart rate, and are normally acquired from third-party sources. Because the security of these medical devices was not a primary design consideration during their construction, they are susceptible to cyberattacks that aim to obtain the sensitive data stored on them [8].

Cryptography is an essential part of Information Security; it is protecting the data by encryption. Encryption is the act of encoding a message or piece of information by transforming it from plaintext to ciphertext in such a manner that only allowed parties can access it, and those who are not permitted cannot. This is necessary to ensure the data's secrecy and integrity [9].

As mentioned before, IoT devices are resource constrained and can't rely on the traditional encryption algorithm and need a special encryption algorithm that requires low resources without complex operations, the National Institute of Standards and Technology (NIST) recommended favoring lightweight cryptography algorithms (LWC) that provide the same level of security [10].

Unlike some other cryptographic methods, lightweight cryptography doesn't require a specific device to work. It works well with devices that have a lot of resources to share (such as cellphones, tablets, PCs, servers, etc.). The main goal of lightweight cryptography is to reduce the overall costs of software and hardware implementation. Software has an implementation size, RAM usage, and throughput (bytes per cycle). On the other hand, hardware has code size, memory consumption (RAM), energy consumption, and Gate Equivalence (GE) [11].

The security management requires from two parties each using a symmetric encryption algorithm to generate or agree to a shared secret key in an efficient way and without enter any additional party in this process. It is possible for two parties each with an elliptic curve public–private key pair to create a shared secret across an unsecured channel using the EllipticCurveDiffieHellman protocol, as showed in figure (1.2). Using this shared secret as a key or generating another key from it is possible. A symmetric-key cipher may be used to encrypt future communications using the original key or a derived key [12].

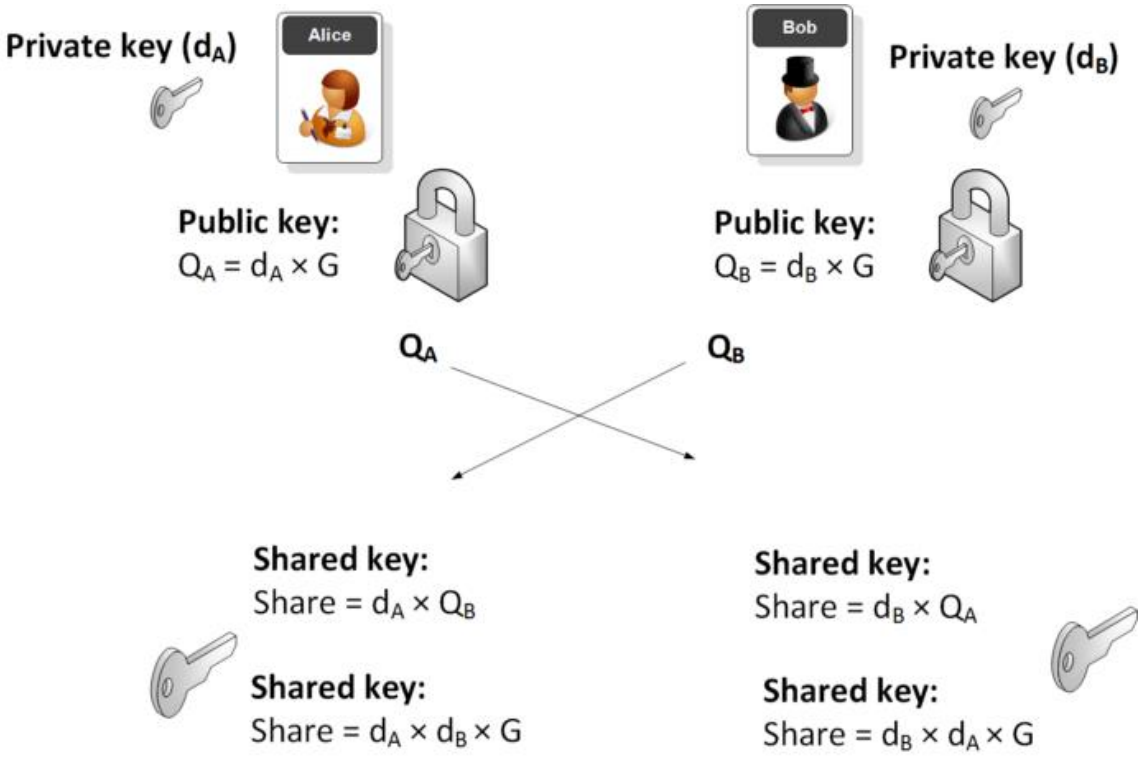

Figure 1.2: EllipticCurveDiffieHellman (ECDH) work [12].

1.2 Related Work

In this section, some related works in term of enhancing IoT data Security using data compression and lightweight encryption techniques have been discussed and overviewed as follow:

A combination of the AES encryption technique and compression with LZ4 has been proposed by (Malekzadeh et al., 2018). Combining these two approaches will hopefully result in an enhancement in data security as well as a speedup in the transfer of data. They focused primarily on the method of packing, compacting, and safeguarding data using parallel approaches to maximize both the speed at which data can be processed and its level of safety [14].

A method for reducing the storage and transmission bandwidth needs has been proposed by (Abboud et al., 2018) with high data rate satellite applications. In the first option, compression is done using a lossless algorithm (Huffman algorithm). One of three cryptography algorithms that will be used (RC4, Blowfish or AES). Even though the second choice Uses a lossy compression algorithm (the SPIHT algorithm) and the same algorithm for encrypting and decrypting. Lossy SPIHT compression or lossless Huffman compression may be used depending on the entropy level, storage and communication resources available for each block. Each block of a satellite image's compressed bit stream is encrypted or decrypted using a different encryption/decryption technique dependent on the entropy level of that block [15].

A lightweight stream cipher (LSC) algorithm approach has been proposed by (Noura et al., 2019) to achieve top-tier security. The overhead of

this system has been minimized by the use of a few simple operations. The proposed encryption has been shown effective and resilient in security and performance analyses, making it an excellent choice for low-power Internet of Things (IoT) gadgets. Besides being immune to algebraic, statistical, and brute-force assault, this method shows good timing and reduced energy consumption and minimal processing power [16].

In (Jawad et al., 2019) proposed a hybrid method by merging two types of cryptography algorithms (Salsa20 and PRESENT). The primary objective of the hybrid algorithm is to improve the complexity of the PRESENT method while maintaining the nominal performance of computational operations. The proposed approach provided a high degree of randomness, and performs well with fast execution times, but adds complexity while keeping computational speed constant [17].

A novel hybrid technique that used a mix of LED and PRESENT Cipher has been proposed by (Prakash et al., 2019) with a compact key approach. This system was made both quicker and more reliable by the use of the RECTANGLE S-Box. The LED, PRESENT, and RECTANGLE S-Box are the tools that are used throughout the encryption process while the SPECK method is used for key scheduling. The proposed method encrypts 64 bits of plain data and then performs an XOR operation on that encrypted data using a schedule key that is 128 bits long. This cutting-edge system used a 64-bit block plain text XOR with a 128-bit key, which enabled it to keep its resiliency. This technique is very lightweight yet provides security against a variety of key threats [18].

AES and Huffman coding methods are combined to make file encryption more secure and reduce file size by (Ashila et al., 2019). The goal is to reduce the amount of space used for storage and speed up file transmissions. First, the encryption is done, and then the compression. Entropy and Avalanche Effects (AE) was used to figure out how safe a file was after it had been encrypted and compressed. Based on the results of the tests, AES encryption makes files about 25% bigger than they were before. The encrypted file dropped by about 30% after using Huffman algorithm compression, since the compressed file was less than the original file's size. On the other hand, the Huffman file is good for security because it makes AE and entropy values much higher [19].

A new, lightweight variant of the PRESENT cipher has been proposed by (Chatterjee et al., 2020) to improves upon the original by cutting down on encryption rounds and reworking the Key Register. With this extra protection, we've been able to lower the PRESENT round from 31 down to the least secure level of 25. Encrypting the key register improves the proposed algorithm's performance. The gate value performance of this algorithm is superior. The power consumption of this approach, however, has not been verified [20].

A new cryptographic algorithm has been proposed by (Mousavi et al., 2020) that combines RC4, ECC, and SHA-256 to safeguard sensitive data in smart irrigation systems based on the Internet of Things. The ECC algorithm is used to encrypt the RC4 key, and the result of this encryption is changed to SHA-256 so that it can be hashed and used to make mysterious data. The SHA-256 algorithm encrypts RC4 cipher text to make sure that the

data is correct. The results showed the proposed scheme works well in terms of performance, time to encrypt and decrypt, throughput, and security. Also, showed it is strong enough to protect privacy based on an analysis of secrecy [21].

Combining the Zstandard algorithm and the AES algorithm proposed by (Bhargavi et al., 2020) to get both security and performance. The Zstandard compression algorithm used for data compression because has high compression and it come first in this approach. The AES algorithm is then used to change the format of the data so that it can't be read. The keys are made up of random numbers, and they are even safer because they are made in a two-step process. The encrypted data is kept in the cloud, where only authorized users can get to it. If an application or user can prove that they are whom they say they are, they can use their private key to decrypt the data. The results show that the suggested method is good because it makes security better while also making storage and computing simpler. Compared to the single-stage compression method, the system does a better job of sending data faster [22].

In standard Pretty Good Privacy (PGP), compression is done after encryption, and since encrypted data is random, it can't be compressed. So, data that has been properly encrypted can't be compressed. The lightweight PGP encryption has been proposed by (Khan et al., 2020), the compression comes first and then encryption. The proposed method used RSA 4096 bit for asymmetric encryption, SHA-512 for hashing, DES for symmetric encryption, and ZIP for compression. After using the proposed encryption method, the results show that the time and size of the encrypted data are both better [23].

Encryption of data collected by IoT sensors has been proposed by (Fadhil et al., 2021) using a low-overhead, safe, and quick symmetric encryption technique called (LAES). The LAES algorithm is a variation of the original AES algorithm that makes use of chaos theory. The proposed IoT security system has high throughput and low time consumption for encryption and decryption. The results showed throughput is greater by roughly 19.24 %. The hardware implementation shows that this method uses very little processing power (about 1.2 percent) and very little memory (about 26 %) [24].

A lightweight and quick technique for securing all types of IoT data has been proposed by (Mohammad et al., 2022) to improve and enhanced the advanced encryption standard (AES) algorithm by decreasing algorithm computing power and boosting cryptography performance in resource-constrained devices. The proposed technique eliminates the need for mixed column overhead, and the ciphertext is treated by a mathematical function (continued fraction) to compress the ciphertext and make it more perplexing, as well as to increase data transmission speed. The results are shown to be superior in terms of time execution or latency, with short execution time and less compute power as the memory central processing unit (CPU) utilization [25].

Table 1.1: A Summarizing of the Relevant Works

Ref.	Years	algorithms	Evaluation Criteria	Goals
[14]	2018	AES with LZ4	Time, Throughput	enhancement in data security as well as a speedup in the transfer of data.

[15]	2018	(Huffman, SPIHT) (RC4, Blowfish or AES)	Entropy, Peak-to-signal-ratio, Structural similarity index, and Saved time	Reducing the storage and transmission bandwidth needs with high data rate satellite applications
[16]	2019	LSC	Security analysis, Encryption time, and Delay	-Achieve top-tier security. -The overhead minimized by the use simple operations
[17]	2019	PRESENT, Salsa20	Randomness, and execution times	Improve the complexity of the PRESENT method while maintaining nominal performance of computational operations
[18]	2019	LED, PRESENT	Time, Throughput	Achieve high level security for IoT data
[19]	2019	AES and Huffman	Data size, compression ratio, Entropy and Avalanche Effects	reduce the amount of space used for storage and speed up file transmissions and protected
[20]	2020	PRESENT	Non-homogeneity Test, N-gram analysis, Histogram, and Floating Frequency	Improve original PRESENT encryption algorithm
[21]	2020	RC4, ECC, and SHA-256	Cipher text Size, time of encrypt/decrypt,	Safeguard sensitive data in smart irrigation

			throughput of encrypt/decrypt, and security	systems based on the Internet of Things
[22]	2020	Zstandard with AES	Data size, Time of encrypt/ decrypt, and Time of compression/ decompression	Provided high level of security as well as fast transmission
[23]	2020	LPGP (RSA, SHA-512, DES, ZIP form compression)	Total time, size of data	Improve standard Pretty Good Privacy protocol
[24]	2021	LAES	avalanche Criteria, Balance Criteria, completeness Criteria, strict avalanche Criteria, invertibility Criteria, Time, Throughput	Design a secure IoT system to protect data
[25]	2022	LAES (AES with Continued Fraction)	Time of encrypt/decrypt, Cipher size, RAM, Throughput, Avalanche effect, and Cipher-text expansion	Improve the AES algorithm by reduce delay and computation power

1.3 Problem Statement

The Internet of Things has a large amount of sensitive data that needs to be sent over the network and protected from attackers to ensure the information accessible to the final decision maker. In this case, there are three main problems with IoT data, which are as follows:

- The Internet of Things devices generate a huge amount stream of real-time data, which can effect on the connection bandwidth [26].

Many applications, such as healthcare data, need real-time connection and monitoring with high throughput [27].

- The main problem of IoT is its openness and wide distribution, thus it is difficult to provide security services [28]. So, Traditional encryption methods are completely unsuitable for the type of devices supported by the Internet of Things [29].
- A major drawback of traditional symmetric encryption was the need that a key be generated by one of the parties and then sent to the other. An unauthorized person may steal or copy the key, allowing them to decipher any ciphertext they'd previously encrypted with it. Key exchange and distribution are an extra problem in the IoT environment [30].

To address these problems, a Compcrypt approach which is a combination data compression algorithms and lightweight cryptography with the Elliptic Curve Diffie-Hellman (ECDH) key exchange protocol to agree on a shared key for use in the encryption\ decryption data have been designed and implemented. The Compcrypt system provide the main objectives of increasing data transfer speed, saving network bandwidth by decreasing the amount of data sent, and protecting data.

1.4 Aim and Objective

The main aim of this thesis is improving IoT security and the objectives can be summarized as follows:

1) A data compression with Zstandard algorithm is being used to increase the speed of data transmission and decrease the amount of data while saving bandwidth on the Internet of Things network.

2) Increase the level of security needed to protect data generated by sensors of the Internet of Things by using TEA lightweight encryption algorithm.
3) Creating a strong shared key that is hard to break in the system by using an Elliptic curve Diffie Hellman (ECDH) key exchange protocol.
4) Comparing the obtained results with most related work for security and data compression.

1.5 Thesis Contribution

1) Implementing and configuring the secure IoT environment based on devices such as raspberry pi and two sensors (Temperature and oximeter sensor) in real-time.
2) Save IoT network bandwidth and improve throughput due to decreasing the amount size of data, and increasing speed of data transmission based on the Zstandard compression algorithm.
3) A robust encryption method based on TEA lightweight algorithm and shared key generated by using Elliptic Curve Diffie-Hellman protocol that helps in securely distributing this key.
4) Enhancement the IoT data security by using TEA lightweight encryption algorithm and Zstandard compression algorithm due to minimizing size of plain text and increased randomness of data.

1.6 Thesis Outline

In addition to chapter one, this thesis contains four chapters:

Chapter Two: Introduce a theoretical part of proposed work, which includes a general view of the Internet of Things (architecture, application, and its

challenges), Data compression algorithm, Lightweight cryptographic algorithm, key management, and the standard evaluation performance.

Chapter Three: Presents the proposed system and illustrates the practical stages of the system and explains the proposed key method and pseudo code for each algorithm based on several parameters to explain each step in this work.

Chapter Four: Presents the test performed, followed by analysis and evaluation obtained results during this work.

Chapter Five: Shows the main conclusions of this work and it also offers suggestions for future works.

Chapter Two

Theoretical Background

CHAPTER TWO
THEORETICAL BACKGROUND

2.1 Introduction

In this chapter, basic background information about the Internet of things and its architecture is provided. Next, it deals with the application of the Internet of things and its challenges. Then it focuses on the main types of data compression methods and the lightweight cryptographic algorithms. This chapter discussed the important way to increase the strength of the encryption algorithm is to find a robust way to generate a secret key. The Elliptic-curve Diffie-Hellman (ECDH) is used as a promising way to generate the secret key. Then explain the implementation tools used in the thesis. Finally, this chapter ends with the performance criteria such as throughput, encryption time, memory used, avalanche effect, entropy, and data compression in the evaluation. Figure (2.1) shows the main steps of the research landscape.

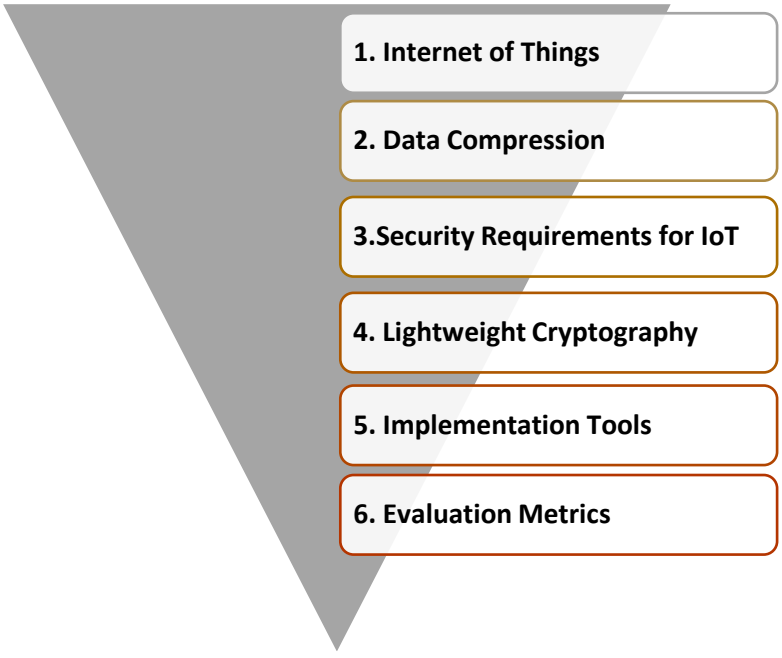

Figure 2.1: Pyramid of the Research Landscape

2.2 Internet of Things (IoT)

As the Internet has become more integral to our everyday lives, a new frontier has opened up for its use as a medium via which machines, smart devices, and electronic objects may interact with one another (thereby making the possibility of a better life for humans a reality) and with which people can have conversations with and use tools independently of one another. The name given to this system is the Internet of Things (IoT) [31].

It is a network of physical items that have been combined with sensers, programmers, and different technologies so that they may communicate with each other and share data through the internet. The Internet of Things depends on embedded sensors, actuators, processors, and transceivers to give its many parts the intelligence and connectivity they need. It's not just one technology; It is comprised of several diverse technologies that collaborate with one another [31].

Sensors and actuators are both kinds of devices that help us communicate with the real world. Sensing information must first be saved and then processed in an intelligent way to draw meaningful conclusions from the data gathered by the sensors. If an object can offer inputs on its present state (internal state plus surroundings), then we may consider it a sensor. An actuator is a device that changes the environment, as in an air conditioner's temperature controller [32].

Different technologies such as Wireless Sensor Networks (WSN), Radio Frequency Identification (RFID), and other machine-to-machine interfaces, cloud services, and so on, may be used to create this relationship, such as a mix of these technologies are shown in figure (2.2). The IoT may be found in a variety of applications in everyday life, including smart homes, health care, smart cities, smart agriculture, smart manufacturing, and other similar applications. The Internet of Things devices are referred to as resource constrained devices due to their restricted

resources (processor, memory, and storage) and power consumption, especially when powered by a battery [33].

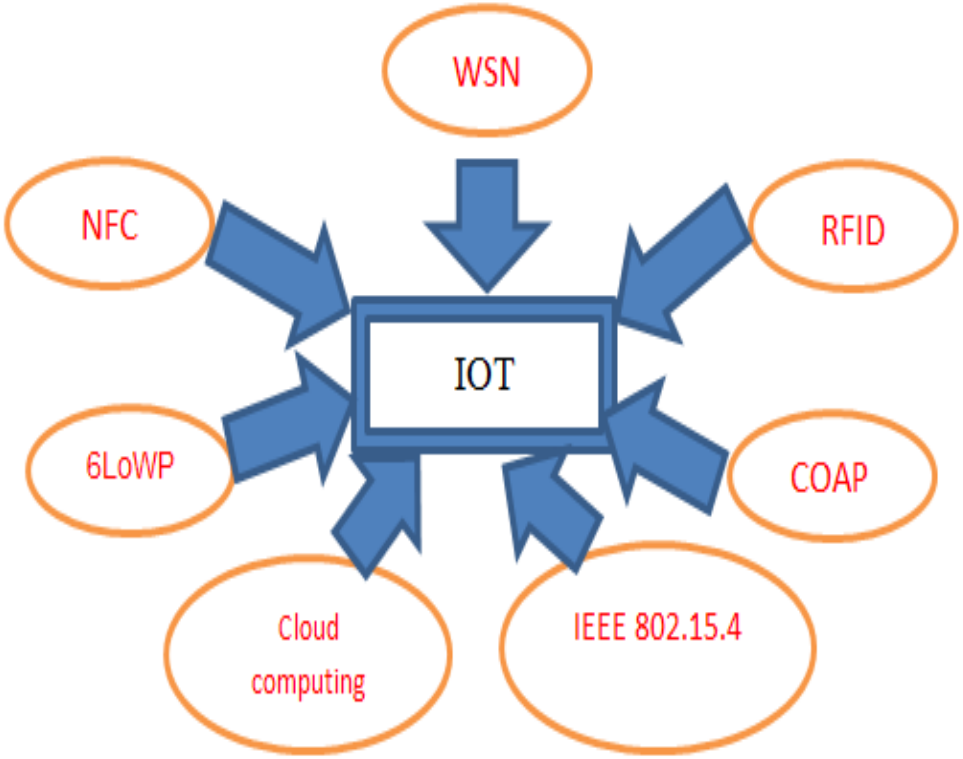

Figure 2.2: The Technologies of Internet of Things [33]

2.2.1 Architecture of IoT

The architecture of the Internet of Things is not standardized. Researches have proposed a variety of architecture; typical has three-layers architecture (perception, network, and application) as shown in figure (2.3) [34].

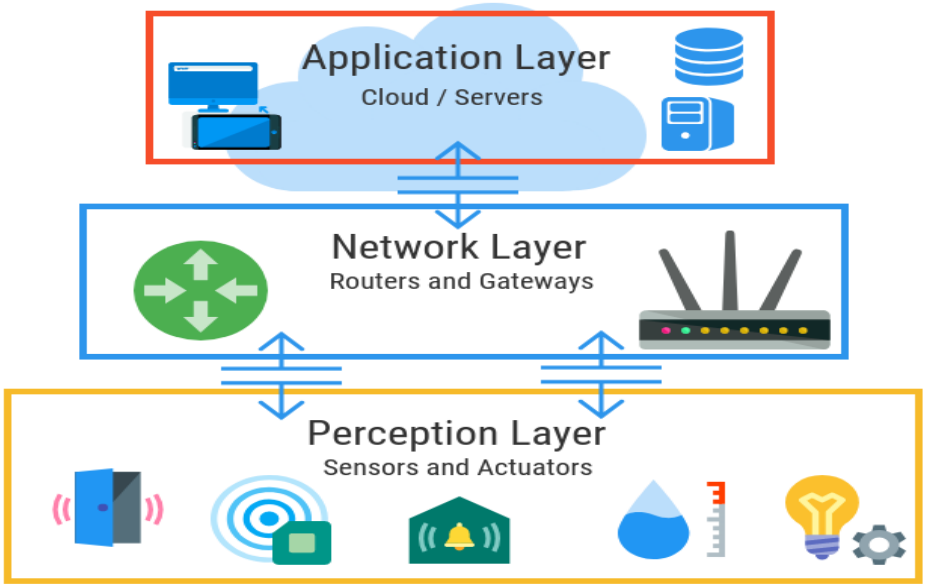

Figure 2.3: Three layers of the IoT Architecture [34]

i. **The Perception Layer** is comprised of a variety of physical Internet of Things devices, and it is accountable for the interaction that occurs between devices as well as the collecting of IoT data. The collection of data is carried out with the assistance of intelligent equipment like radio frequency identification (RFID) tags and sensors [35].

ii. **The Network Layer** evaluates the acquired data from the perception layer and stores or transmits it to the application layer. There are a number of communication methods on this layer that enable IoT devices to interact with one another [35].

iii. **The Application Layer** takes data from the network layer and offers IoT users with the necessary services. It is compatible with a wide range of applications, including those for the smart home, smart retail, healthcare, and other similar uses [35].

The Internet of things is defined by a 3 layer, yet this is insufficient for research since it focuses on minute details. The literature, on the other hand, suggests a plethora of multilayer architectures [36].

One is the five-layer architecture as shown in Figure (2.4), which incorporates the processing and business levels in addition to the other layers. Perception layer, transmission layer, middleware (processing) layer, application layer, and business layer are the five layers. The work of the perception layer and application layer is identical to that of a three-layer architecture [37].

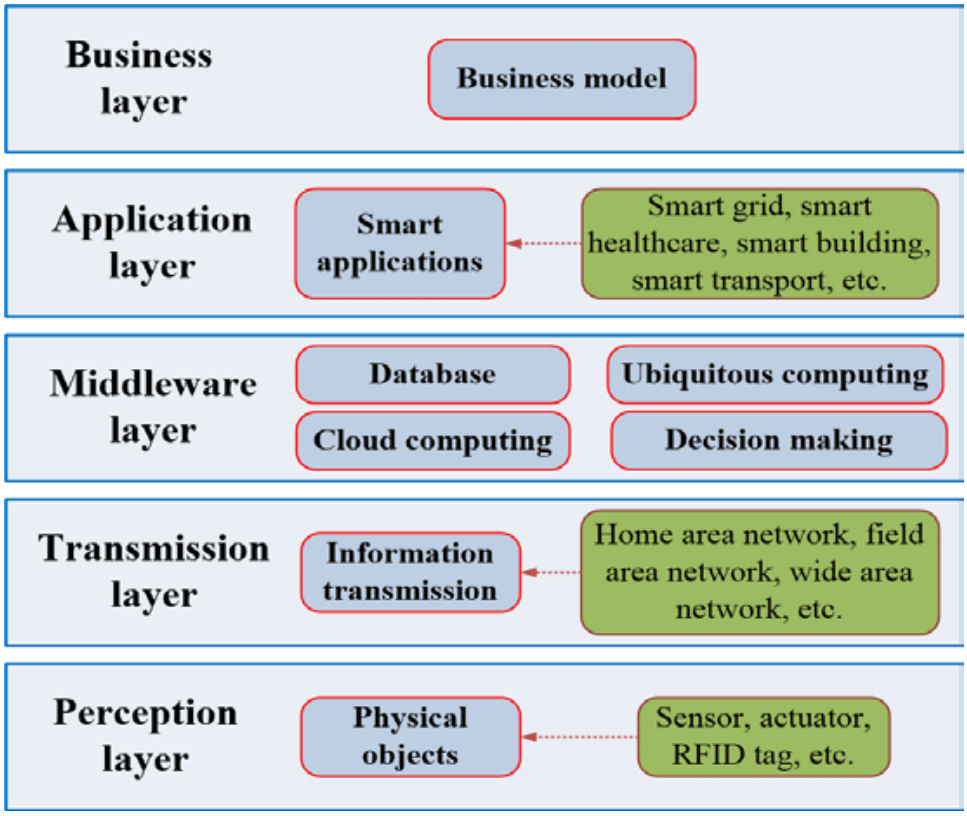

Figure 2.4: Five layers of IoT Architecture [37].

i. **The Transmission Layer** transports sensor data across layers, from the perception layer to the middleware layer and back again via various transmission networks [38].

ii. **The Middleware (processing) layer** Massive amounts of data from the transport layer are stored, analyzed, and processed at the processing layer. It is able to exert command on the lower tiers and provide them with a broad range of support services. Technologies such as database management systems, cloud computing, and big data processing modules are all included [38].

iii. **The business layer** controls the whole of the IoT system, including all of its applications, business and profit models, and consumers' privacy settings [38].

2.2.2 Applications of IoT

Intelligent applications have been made for many different kinds of things. All of these uses are not yet widely available, but early research shows that IoT in has the potential to make life better for people in our society. IoT applications can be used to smart home, smart cities, natural disaster monitoring, health monitoring, industrial IoT, and etc. [39]. Figure (2.5) shows potential examples of IoT applications.

Figure 2.5: Some of IoT Application [39]

A. Smart Home

Home automation, also called smart homes, is the automation of a home's building systems. A home automation system will keep an eye on and/or control things like the lighting, temperature, devices, and entertainment systems. It could also include things like door locks and alarm systems for home security [39].

B. Smart Cities

A smart city is an advanced metropolitan region that makes use of various electronic means, voice activation techniques, and sensors to gather specialized data. By better managing assets, resources, and services, the city as a whole benefits from the information provided by this data [40].

C. Natural Disaster Monitoring

Using wireless sensors, natural calamities including earthquakes, landslides, forest fires and volcanoes may be predicted in advance. These signs warn the proper authorities so that they may prepare for an oncoming catastrophe [41].

D. Industrial Application in IoT

The industrial internet of things (IIoT) is used to describe the networking of sensors, instruments, and other devices with computers for use in industrial applications including production and energy management [42].

E. Healthcare Monitoring

The Internet of Things is used in the healthcare sector to enhance human lives by making routine chores easier. Patients' health monitoring devices may be outfitted with sensors. To enhance therapy and response, the data gathered by these sensors are made accessible online to clinicians, family members, and anyone with an interest. IoT devices may also track a patient's existing prescriptions and assess

the potential for bad drug responses or drug interactions with new meds. Using the sensors and technologies, we can monitor the subject's vital signs in real time, including their core temperature, heart rate, blood pressure, and so on. All the information gathered by the sensors will be sent to the individual and their personal doctor in the event of an emergency. Independent seniors and those with disabilities will benefit greatly from using this method. Figure (2.6) shows the components of the Patient monitor System [43].

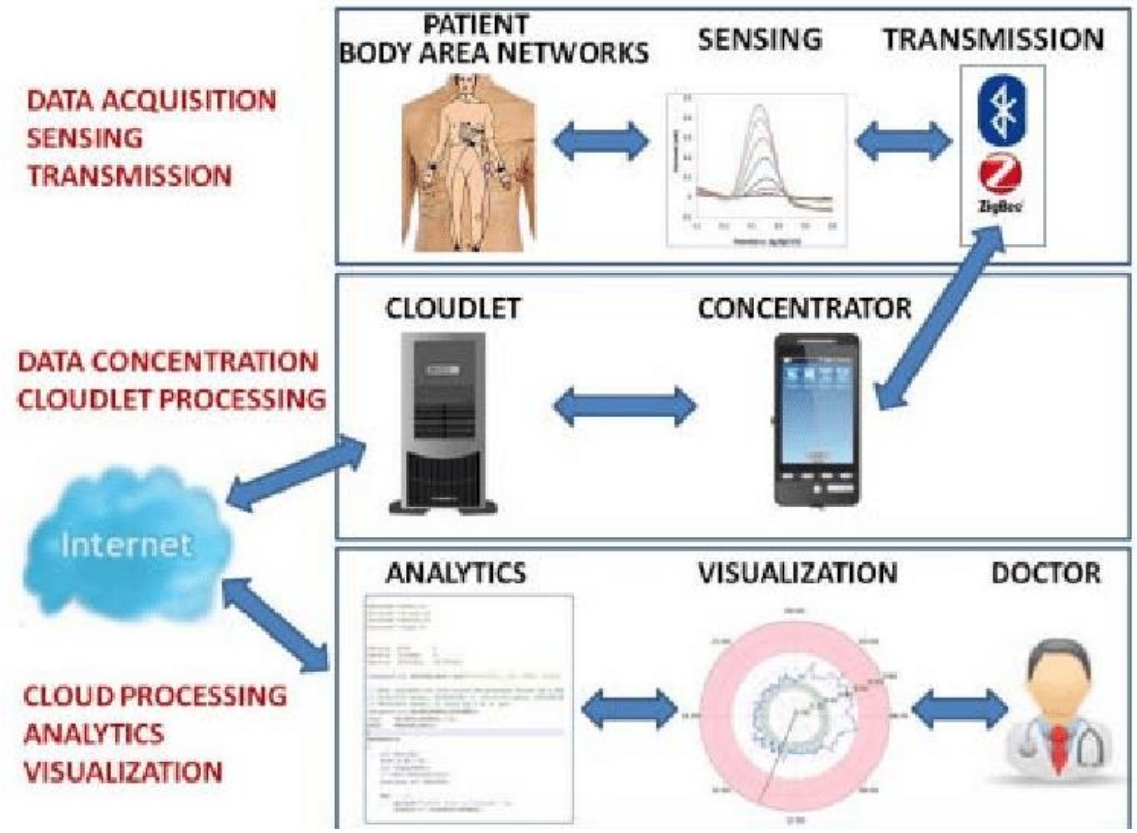

Figure 2.6: Components of Patient monitor System [43].

2.2.3 Challenges of IoT

The Internet of Things as a concept has the potential to significantly alter many aspects of modern society. Challenges develop when billions of items and devices join the internet, causing information convergence, from a wide range of

IoT's implications. As a result of the IoT's complexity, there are sevral issues that need to be resolved before the full potential of the system can be realized [44]. Figure (2.7) summarize the 6 main challenges of the IoT system.

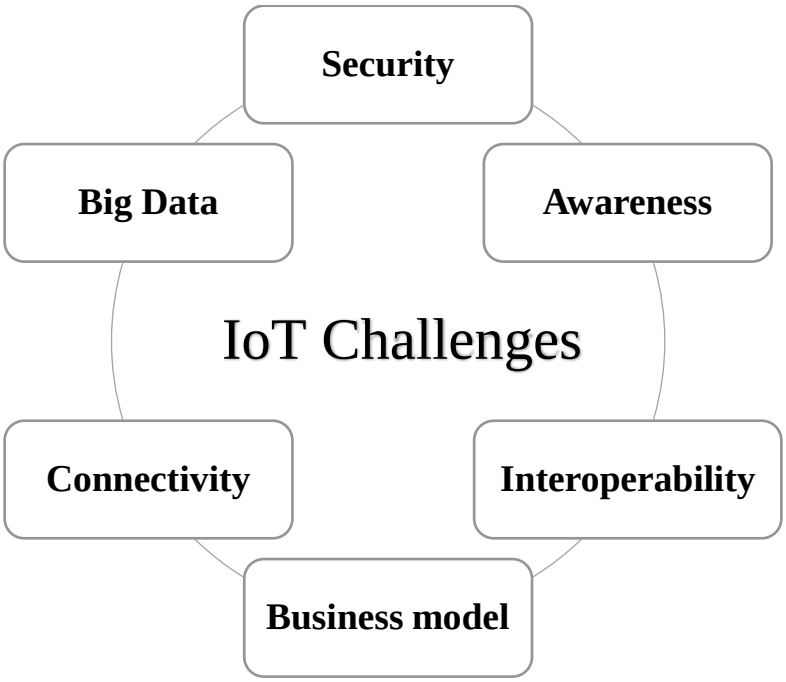

Figure 2.7: The main challenges of the IoT system

A. Security

Security is the main challenge to fully implementing the Internet of Things. Any internet-connected device may be attacked. Thousands of IoT devices were compromised because the attack exploited of their security flaws. Several security necessities in the IoT that must be met. These include things like authorization, authentication, confidentiality, trust, and data security. Things' data must be gathered, analyzed, stored, transferred, and displayed in a fashion that is safe against malicious software assaults at all times [45].

B. Awareness

The Internet of Things still receives little attention from the community. Although the Internet of Things has the potential to improve many areas of human

existence. The awareness issue is the major hurdle to the growth of the IoT sector [46].

C. Interoperability

Interoperability becomes one of a primary problem for the development of IoT system since all items being connected and integrated were diverse. The "things" may be created by multiple producers that may not necessarily meet a single standard. It may function utilizing a diversity of communication technologies and required interoperability like a simple bridge between multiple protocols. Achieving interoperability is crucial for joining many items together over various communication networks [47].

D. Business model

The Internet of Things have far-reaching effects on the economy because it will help many businesses become digital. The issue that still occurs today is how businesses may make money off of their IoT service [48].

E. Connectivity

The growing number of heterogeneous IoT devices places a heavy burden on the internet's capacity and introduces new connection issues. Transferring sensor data, storing it in the Internet of Things backend, and automating the process of feeding that data back to the actuator or application are all dependent on connectivity. Wi-Fi, Bluetooth, Zigbee, LoRa, 6LoWPAN, LTE, and many more all fit the bill as communication mediums. Different technologies have different features, such as signal strength, data transfer rate, and power consumption, as well as different demands on the communication device's battery life. For example, in a smart home, the constant transmission of signals and data by various IoT devices may significantly strain the available power supply [49].

F. Big Data

Big Data is enormous volumes of data generated simultaneously by the sensors of various devices. Due to large amounts of data, IoT devices generate different data types. These massive amounts of data need some technology to the gathering, cleaning, and analyze it. For this reason, deploying an IoT system with Big Data is quite difficult [50].

2.3 Data Compression

Devices and sensors linked to the Internet of Things provide real-time monitoring and measurement of data. To gather data, IoT sensor nodes continuously take measurements and transmit them back to the cloud. The collection of IoT data often necessitates the collection of massive amounts of data streams [51].

Sensors that are close to each other often produce data that is highly related and duplicated. This is because sensors that are placed in close proximity to each other usually pick up on the same things, which would cause them to produce a lot of duplicate data. Because this data will be duplicated, it will take up too much bandwidth and time. Also, the station will have to deal with a huge amount of data from a large network of sensors [52]. This leads us to the conclusion that we should look for ways to eliminate redundancy and consolidate data at IoT sensor nodes to reduce the number of packets that will sent and so conserve network capacity. Data compression may help with this.

A data compression process is one in which data is encoded, restructured, or otherwise modified in order to minimize its size. As a basic principle, it includes reencoding data in a way that uses fewer bits than the original. Data compression represents one of the procedures that conserves the secrecy of the data by encoding it. Data compression in IoT devices is a great way to save the limited resources available on these devices. By using data compression techniques, it is possible to

reduce the quantity of data transfers and save bandwidth in the network, which will also reduce the size of transmitted data and increase the speed of data transfer in the network [53].

To reduce the size of a text file, it is usual practice to remove any unnecessary characters, add a single character to serve as a reference for a long string of similar characters, and then swap out the larger bit string with a smaller bit string. When done correctly, data compression may reduce the size of a text file by as much as 50% or more [53]. Figure (2.8) explained the text file compression.

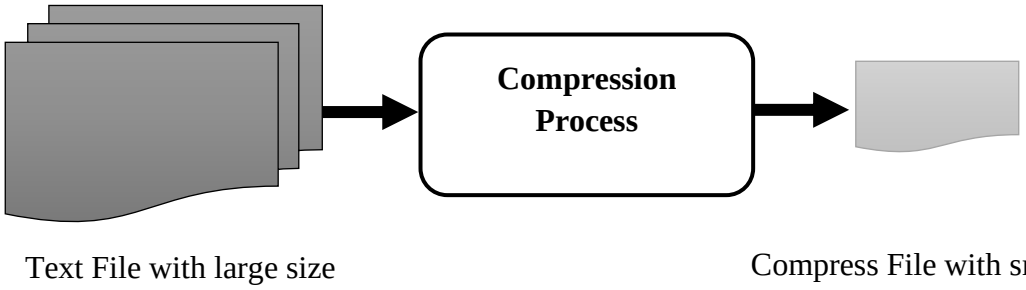

Figure 2.8: The Process of File Compression [53]

The main benefits of compression are reduced size in storage hardware, less time spent sending data, and save communication bandwidth. This can lead to big savings on costs. Compressed files take up much less space than uncompressed files, which means that the cost of storage goes down by a lot. A compressed file takes less time to send over the network and uses less bandwidth. This can also save money and make people more productive [54].

The biggest problem with data compression is that it uses more computer resources to compress the data. Because of this, compression manufacturers prioritize speed and resource efficiency enhancements at the top of their lists in order to minimize the impact of their most demanding compression activities [54].

Compression may be lossy or lossless, as seen in the following figure (2.9). Lossless compression reduces the quantity of data transferred by finding and

deleting statistical redundancy. In lossy compression, information that isn't absolutely essential is omitted from the compressed file, resulting in smaller file sizes [55].

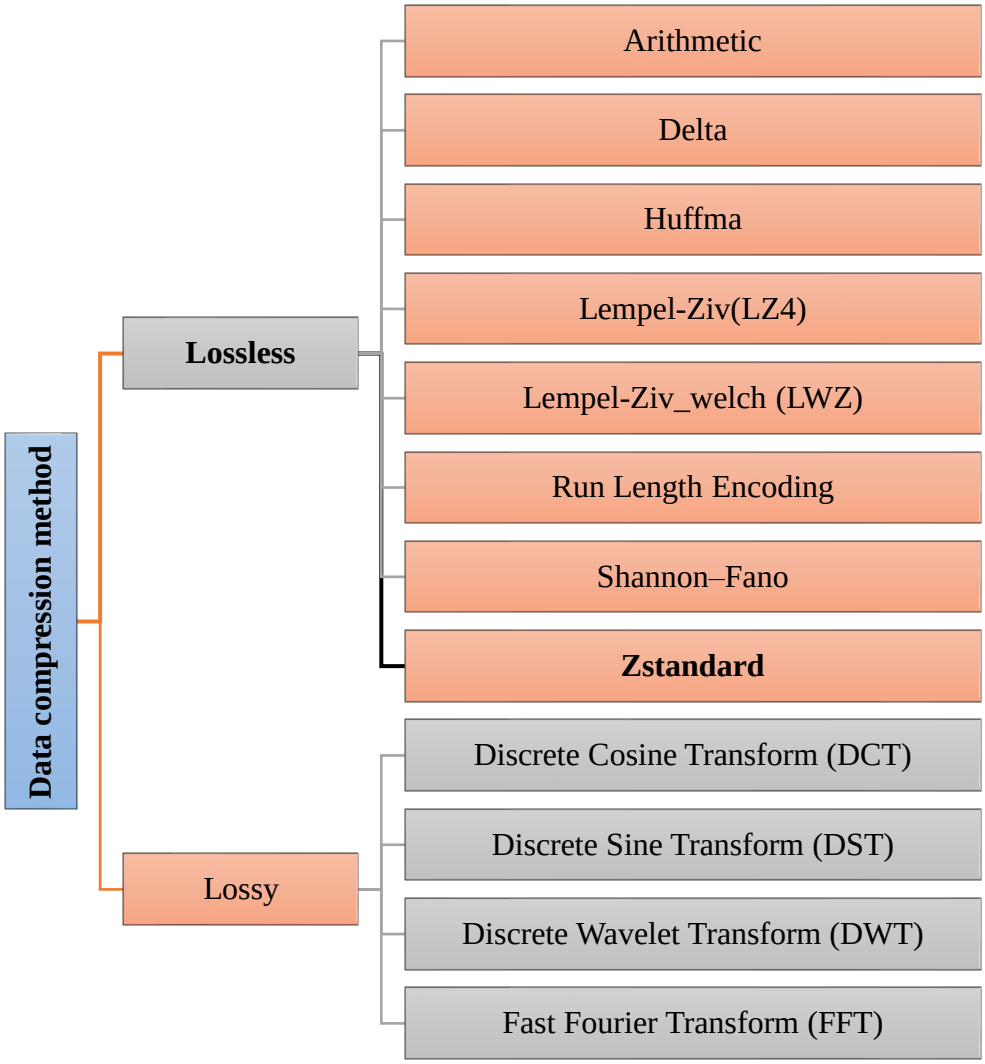

Figure 2.9: some important algorithms for Data Compression.

2.3.1 Lossy Algorithm

Lossy compression methods lose some information, and most of the time, data that was compressed using a lossy method can't be correctly reconstructed. Usually, we can get much better compression ratios than with lossless compression by letting the retrieval process cause some distortion. In lossy algorithms, the data that is reconstructed is not the same as the original data. So, when a lossy

compression algorithm is used, the compressed data can't be used to get back to the original data. This doesn't prove that the information being retrieved is bad. A good loss algorithm keeps the important information and gets rid of the less important information [56].

Lossy compression is based on the idea that modern data files store more information than people can understand. As a result, the Lossy data compression is most used in MP3 audio, MPEG video, and JPEG image formats [57]. Figure (2.10) showed the idea of lossy compression.

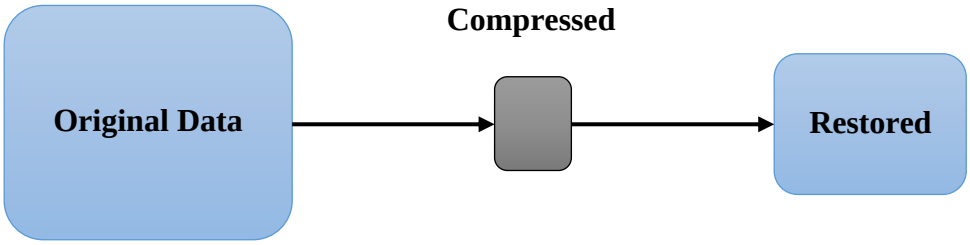

Figure 2.10: The Lossy Data compression [57]

2.3.2 Lossless Algorithm

A lossless data compression algorithm compresses data without losing any information in the compression process. In addition, the decompression data process by a lossless algorithm results in the same original data exactly. Lossless methods are often the methodology of choice in situations when complete reconstruction is more essential than compression ratio, such as when text file compression, picture compression, and numeric data storage [58]. Figure (2.11) showed the idea of lossless data compression.

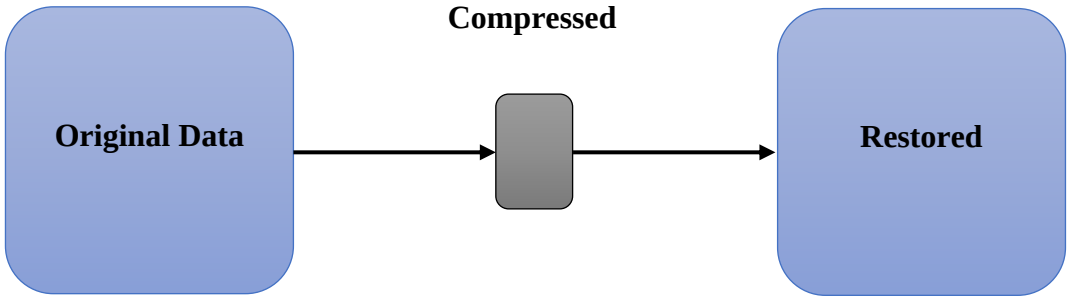

Figure 2.11: The lossless Data compression [58]

In all lossless algorithms, the goal is to find a pattern in the data and compress it more when it shows up. This way, the total size of the data can be reduced without any information being lost. So, all of the original data can be put back together again. In text compression, the algorithms figure out how often each string of characters shows up. But it's hard to use a textual compression algorithm when the object being compressed is a series of numbers that change over time. In this case, data difference, which is a more common way to compress data, is used [59].

For instance, in the method known as delta encoding, the initial data value is maintained but only the differences between the subsequent nearby data samples are encoded. This is done in place of recording the initial values of the data samples themselves [59].

In this thesis, we only focus on the lossless compression method because the model of the system deals with healthcare applications and the sensor data is sensitive.

2.3.3 Zstandard compression algorithm

Zstandard, sometimes referred to as Zstd, is an algorithm for real-time compression that is free, open-source, and was invented by Yann Collect, an

employee at Facebook. Zstd is incredibly quick and provides excellent compression ratios at the same time. It is a lossless compression technique that was designed in C language, but it has API implementations in other prominent programming languages such as Python, Java, C#, JavaScript, and a lot of others languages. On the 31st of August, 2016, Version 1 of this implementation was made available to the public as open-source software. It is reasonable to anticipate that Zstd will reduce the size of text, executable code, and associated data files to around fifty percent of their initial dimensions [60].

When compared to the DEFLATE method, Zstandard compression ratio is similar, but it is quicker, particularly during decompression. Finite State Entropy and performance-first design are combined in Zstandard, which then optimizes the implementation for modern CPUs. Consequently, it improves on the tradeoffs of previous compression algorithms and offers a broad variety of applications with extremely fast decompression [61].

Zstandard utilizes Finite State Entropy and Huffman coding to achieve high compression rates while also including a dictionary-matching stage (LZ77) with a huge search window. Because Finite State Entropy shares information across symbols, decompression requires processing the symbols in the Sequences subsection of each block in the opposite direction (from last to first) [62].

The Zstandard algorithm is one of the most popular lossless compression algorithms, in which the dictionary is created dynamically to encode new strings based upon strings previously encountered where an initiated dictionary contains the strings of a single character corresponding for to potential input characters. For instance, when using the “American standard code for information interchange (ASCII)”, the dictionary will include 256 initial entries. The Zstandard algorithm then searches each character of the incoming stream of data until a substring that

was not in the dictionary can be found. When it detected such a string, the longest identical substring index in the dictionary sends to the output stream of data, while adding the new string into the dictionary with the next obtainable code. Then, the Zstandard algorithm continues checking the input stream of data, it starts with the last character of the preceding string [63].

2.4 Security Requirements for IoT

IoT devices must meet fundamental security criteria such as authorization and authentication as well as confidentiality, availability, integrity, and non-repudiation which showed in figure (2.12).

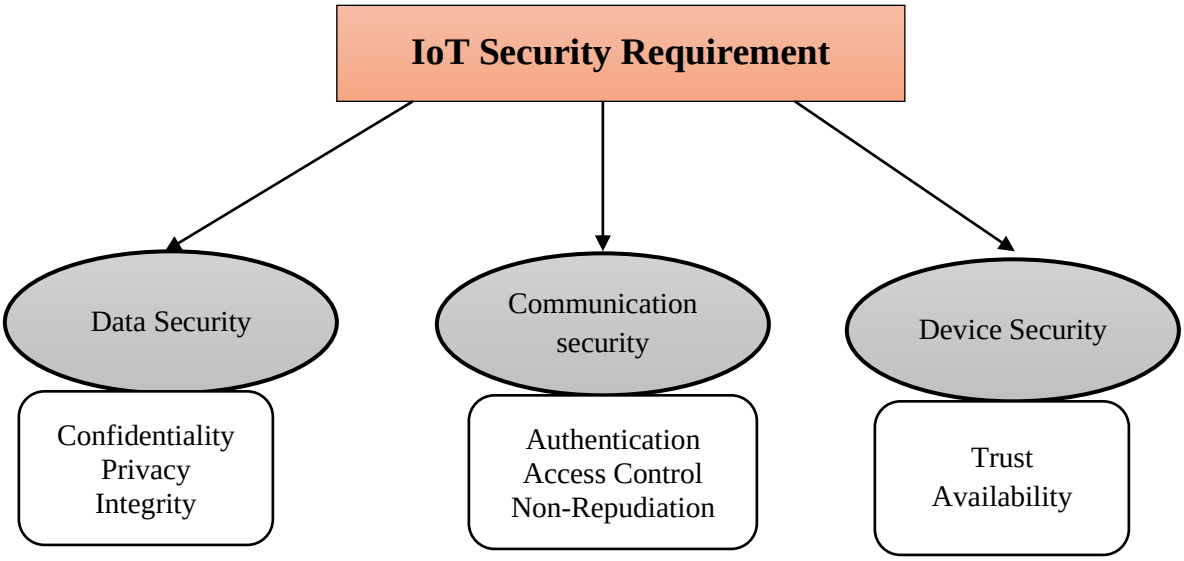

Figure 2.12: The Security Requirement for Internet of Things.

2.4.1 Confidentiality

Confidentiality makes sure that sensitive information can't be seen by unauthorized people. Therefore, it assures and pledges that only authorized parties may access, change, or delete sensitive information. In an IoT environment, when sensors collect personal information, encryption technologies may be used to protect the information and prevent it from being accessed or discovered by a third person.

Therefore, to achieve the confidentiality goal data sent between two devices must thus be encrypted before it is sent [64].

2.4.2 Integrity

Integrity means that data can't be changed by unauthorized people during the transmission sessions. The system gives ways to find out if data has been tampered with or changed without permission. Data integrity is critical to the safety of smart devices in an IoT network. Also, cryptography can be used such as HMAC-SHA 256 algorithm for achieving data integrity [65].

2.4.3 Availability

Availability IoT security also needs to make sure that resources are always available to the right people, no matter where or when they are. Availability means that a legitimate user should be able to get to the resources and information when they want it. A sensor is also considered to be available in the architecture of the IoT if it can to communicate the data that it has measured in real time. Also, if an actuator is present, it can do what the user tells it to do right away, without any significant delay. On the other hand, attackers can use three main types of malicious attacks to hurt availability: DOS, flooding, or black hole attacks [66].

2.4.4 Authentication

The authentication service is regarded as the greatest obstacle in the IoT network. Verification of identification is part of the authentication process. The devices can verify the authenticity and legality of remotely used devices on a public network throughout the authentication process. The good achievements of authentication process by applying multifactor methods such as (password, PIN, One Time Password, SMS text, biometrics). Figure (2.13) showed a good example of the authentication process [67].

Figure 2.13: The example of authentication process [67].

2.4.5 Authorization

Authorization is becoming an increasingly important concern in the IoT system as the number of items that are linked to the network continues to rise. Actually, it is a reference to the security service that is in charge of deciding the rights and privileges of users. It also outlines the access control rules that must be followed to grant or revoke rights for IoT devices. Therefore, the problem is to stop individuals with restricted rights from gaining more permissions so that they may get unlawful access to devices and the data stored [68]. Figure (2.14) showed the difference between the authentication and authorization processes.

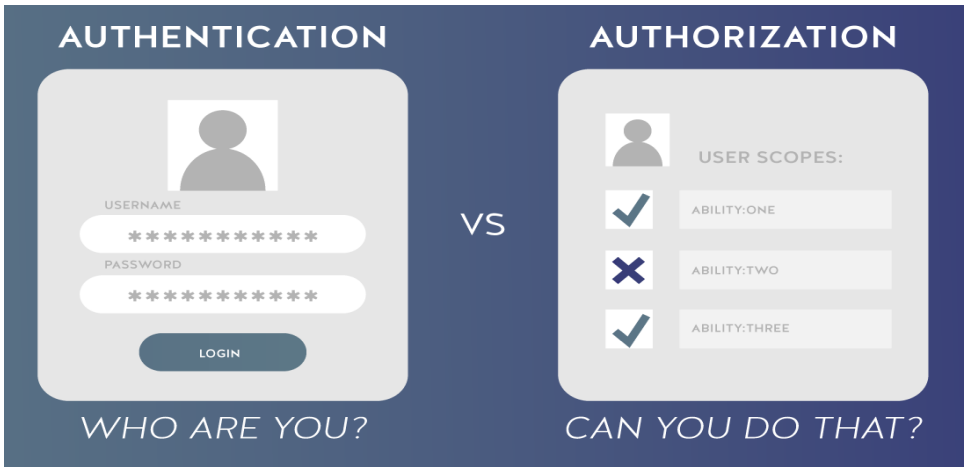

Figure 2.14: The difference between authentication and authorization process [68].

2.4.6 Non-Repudiation

Non-repudiation ensures that the sensor node can exchange integrality data. Additionally, it ensures the authenticity of the transmission of data or identifications between two IoT, which cannot be disputed. Non-repudiation provides promises to a source node that it will communicate its data and also provides guarantees to a receiving node that it will check that the data it has received matches the data's source [69].

2.5 Data Security of the Internet of Things

Many new issues arise when billions of smart devices are functioning on a variety of platforms, particularly when they are moving from servers to sensors. Among these are issues of security and privacy, interoperability, long-term viability, and many more. Due to their direct connection to the actual world, IoT devices are particularly sensitive to a broad range of security concerns, including unauthorized access and the modification of the variables in the physical environment. Because of this, fraudsters see IoT devices as attractive targets. IoT device security is challenging to achieve due to these and other considerations, including the need to meet high criteria for availability, authentication and authorization, as well as accessibility, privacy and regulatory regulations. The IoT security challenges and requirements of IoT showed in figure (2.15) [70].

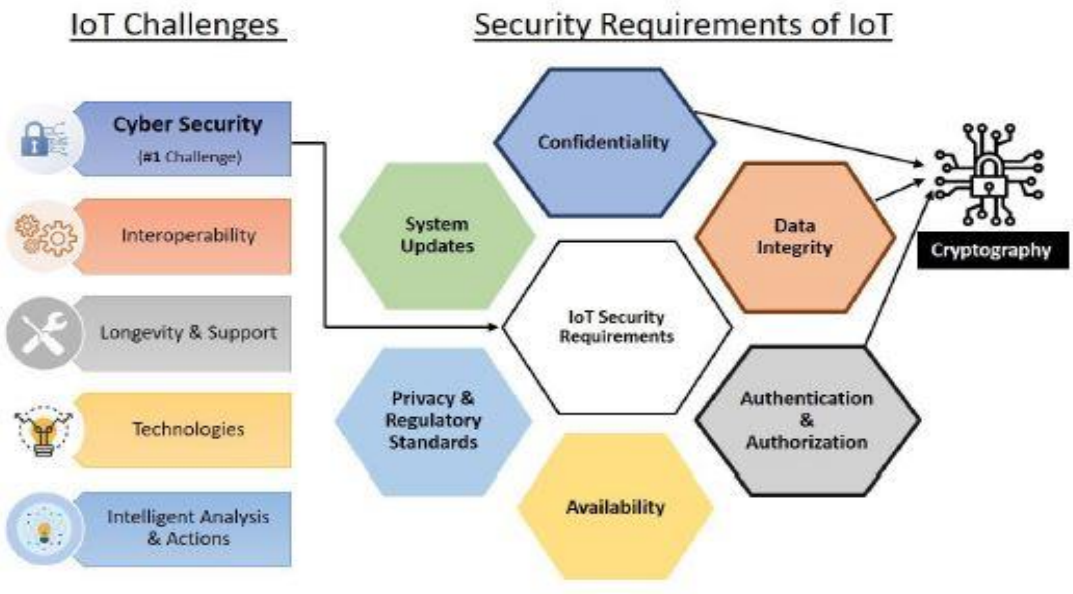

Figure 2.15: The IoT Security Challenges and its requirements [70].

In this case, cryptography may be one of the effective steps that can be taken to ensure the confidentiality, integrity, as well as authentication and authorization of the data that is travelling via IoT devices. However, traditional encryption algorithms do not work in resource-constrained IoT devices since they need a lot of resources. The National Institute of Standards and Technology (NIST) recommended favoring lightweight cryptography algorithms (LWC) that provide the same level of security as traditional algorithms, and their performance is also acceptable on these resource-limited [71].

The key challenges while implementing conventional cryptography in IoT devices as shown in figure (2.16) are as follows [72]:

- Limited memory (registers, RAM, ROM).
- Reduced computing power.
- Small physical area to implement the assembly.
- Low battery power (or no battery).
- Real-time response.

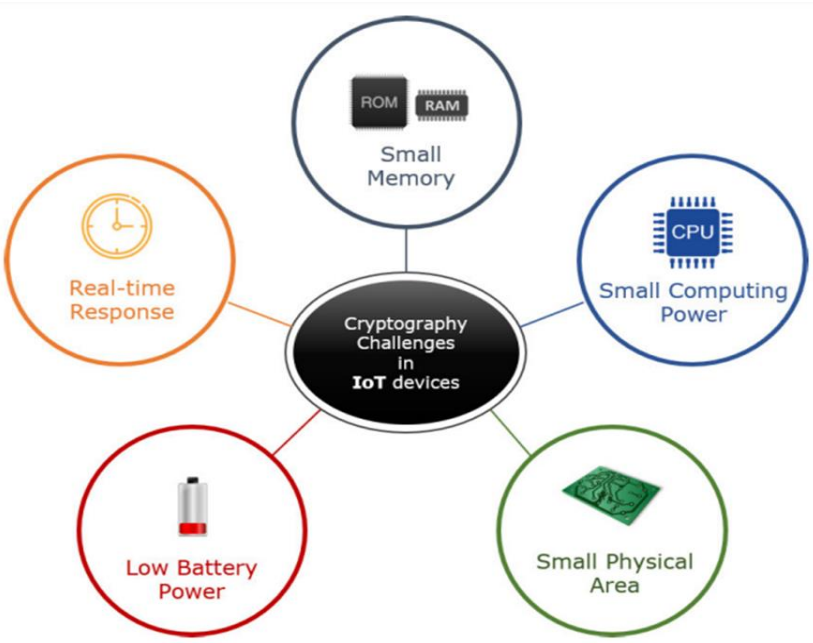

Figure 2.16: Key Challenges with Conventional Cryptography [72].

The low computing power (RAM, ROM, CPU, and battery), and physical space available is small in the IoT devices as showed in figure (2.17), making it difficult to integrate them into larger systems. Real-time applications are a difficult task for most IoT devices due to the limited resources available, and this is especially true when it comes to security. IoT device designers face a multitude of risks and concerns, including energy capacity and data security [73].

Figure 2.17: The IoT devices [73].

Traditional encryption standards may not function well on IoT devices. Lightweight cryptography is particularly adept at resolving these issues. By incorporating lightweight characteristics such as low memory, low processing power, low power consumption, and real-time responsiveness even on restricted resources, it does this. One of the advantages of lightweight cryptography is that it may be used not just on devices that have few resources, but also on devices that it interacts directly or indirectly with (such as servers, PCs, tablets, smartphones, etc.) [74].

2.5.1 Lightweight Cryptography for Internet of Things

Lightweight cryptography (LWC) is a cryptographic algorithm that is designed to work in places with limited resources, such as RFID tags, sensors, contactless smart cards, medical devices, and so on. LWC is made for applications that are made quickly and use a lot of smart devices with limited resources. To create the best environment for the Internet of Things, you need to protect sensitive data and make sure it's easy to manage and flexible enough to deal with the packets that move around on these networks [75], [76].

LWC is a sub-discipline of cryptography that was developed to meet similar difficulties. Lightweight cryptography provides qualities such as speed even on resource-constrained devices, such as little memory and low computing power. Table (2.1) is a listing of the three primary qualities that distinguish lightweight cryptography methods and their respective products [77]:

Table 2.1: LWC Characteristics.

Characteristics		**What LWC can offer**
Physical	Physical Area (GEs, logic blocks)	-Tiny key & block -Simple rounds with simple computation
	Memory (registers, RAM, ROM)	
	Battery power (energy consumption)	

Performance	Computing power (latency, throughput)	-Simple key generation
Security	Minimum security strength (bit)	-Strong internal structure
	Attack models (related key, multi-keys)	
	Side-channel and Fault-injection attacks	

2.5.2 Classification of LWC

Lightweight encryption process is encoding information. The original representation of the information, known as plaintext, is transformed into ciphertext throughout this procedure. Deciphering a ciphertext back to plaintext and gaining access to the original data must be restricted to only authorized persons [78].

Cryptography can be classified into two main branches, symmetric cipher and asymmetric cipher as shown in figure (2.18). Symmetric cipher uses single key for both encryption and decryption of the data, whereas asymmetric cipher uses two keys, public key for encryption and private key for decryption. A symmetric cipher protects the data's confidentiality and integrity, but not its legitimacy. The Asymmetric cipher safeguards both privacy and the integrity of data; it also ensures authentication by the digital signature. The drawback of asymmetric ciphers is the large key, for which the procedure takes longer and is more difficult [79].

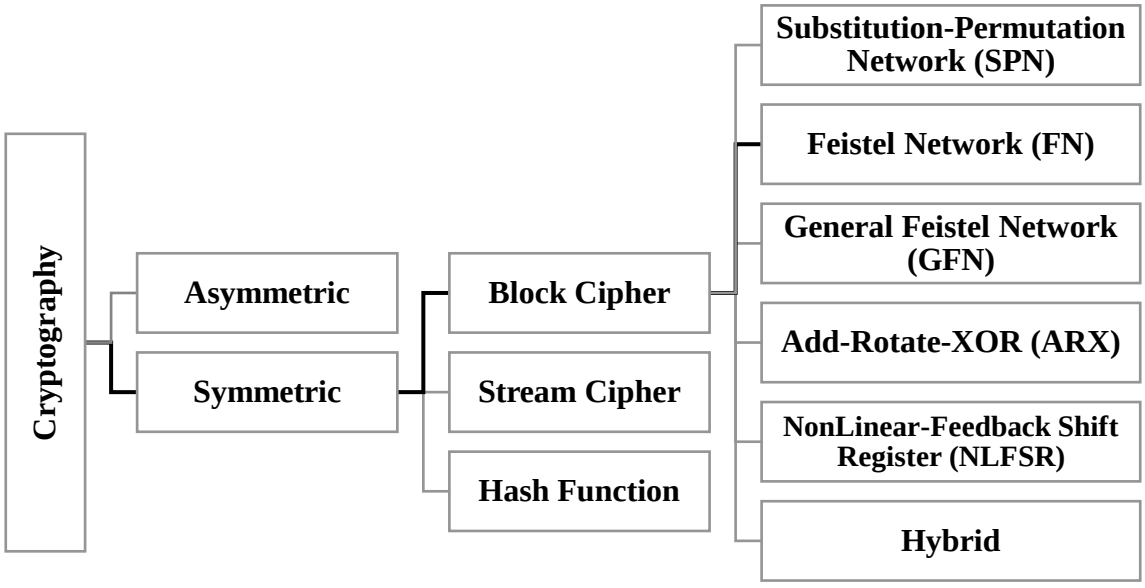

Figure 2.18: Classification of Cryptography

Block cipher encrypts and decrypts a fixed-size block, while stream cipher processes the whole message byte by byte. Hash function is a mathematical formula that maps data of any size to a string of bits with a fixed length. One-way functions can't be turned around [80].

The following structures are used in symmetric lightweight block ciphers:

i. **Substitution-Permutation network (SPN)** processes data through a series of substitution (S-box) and permutation (table) by altering the data and finally formulating them for the next round [81].
ii. A **Feistel network (FN)** is a multi-round cipher that divides the input block into two parts and operates only on a half (diffusion) in each round of encryption or decryption. Between rounds, the left and right halves of the block are swapped [81].
iii. The **generalized Feistel network (GFN)** is a generalized form of the classical Feistel cipher. In GFN, input block is split into two or more sub-blocks and applies a (classical) Feistel transformation for every two sub blocks, and then performs a cyclic shift relevant to a number of subblocks [82].
iv. **Add-Rotate-XOR (ARX)** performs encryption-decryption using addition, rotation and XOR functions without making any use of S-box. Implementation of ARX is fast and compact but limits in security properties compared to SPN and Feistel ciphers [82].
v. **Nonlinear feedback shift register (NLFSR)** can be applied in both stream cipher and block cipher designs. It utilizes the building blocks of stream ciphers whose current state is a nonlinear feedback function of its previous state [83].
vi. **Hybrid cipher** combine the any three types (SPN, FN, GFN, ARX, NLFSR) to improve specific characteristic (for example, throughput, energy, GE, etc.) based

on its application requirements or even could mix of block and stream cipher [83].

More than fifty symmetric lightweight algorithms have been shown in the table (2.2). These algorithms aim to reduce costs and improve hardware and software performance [84].

Table 2.2: The LWC Algorithms.

Structure Type	Algorithms
SPN	AES, PRESENT, RECTANGLE, MIDORI, mCrypton, NOEKEON, ICEBERG, PUFFIN-2, PRINCE, PRIDE, PRINT, Klein, LED
FN	DESL/DESXL, **TEA**/XTEA/XXTEA, Camellia, SIMON, SEA, KASUMI, MIBS, LBlock, ITUbee, FeW, GOST, Robin, Fantomas
GFN	CLEFIA, PICCOLO, TWIS, TWINE, HISEC
ARX	SPECK, IDEA, HIGHT, BEST-1, LEA
NLFSR	KeeLoq, KATAN/KTANTAN, Halka
Hybrid	Hummingbird, Hummingbird-2, PRESENT-GRP (SPN+GRP (Group Permutation))

2.5.3 Tiny Encryption Algorithm (TEA)

The Tiny Encryption Algorithm (TEA) is a lightweight encryption algorithm created by David J. Wheeler and Roger M. Needham from Cambridge University in 1994. It is a block cipher with block length of 64 bits and key lengths of 128 bits. It is a Feistel type cipher that uses operations from mixed (orthogonal) algebraic groups. A dual shift causes all bits of the data and key to be mixed repeatedly. The key schedule algorithm is simple; the 128-bit key K is split into four

32-bit blocks (K[0], K[1], K[2], K[3]) and the 64-bit plaintext is split into two 32-bit blocks (Yi and Zi) [85]. Table (2.3) showed the main properties of TEA algorithm.

Table 2.3: The TEA properties.

Designers	**Roger Needham, David Wheeler**
First published	1994
Key sizes	128 bits
Block sizes	64 bits
Structure	Feistel network
Rounds	variable; recommended 64 Feistel rounds (32 cycles)

A. Encryption Routine

Figure (2.19) shows the structure of the TEA encryption routine. The inputs to the encryption algorithm are a plaintext block and a key K. The plaintext is P = (Left[0], Right[0]) and the cipher text is C = (Left[64], Right[64]). The plaintext block is split into two halves, Left[0] and Right[0]. Each half is used to encrypt the other half over 64 rounds of processing and then combine to produce the cipher text block [85], [86].

- Each round i has inputs Left[i-1] and Right[i-1], derived from the previous round, as well as a subkey K[i] derived from the 128 bit overall K [85], [86].
- The sub keys K[i] are different from K and each other.
- The constant delta =$(\sqrt{5}-1)*2^{31}$=9E3779B9, is derived from the golden number ratio to ensure that the sub keys are distinct and its precise value has no cryptographic significance [85], [86].

- The round function differs slightly from a classical Feistel cipher structure in that integer addition modulo 2^{32} is used instead of exclusive-or as the combining operator [85], [86].

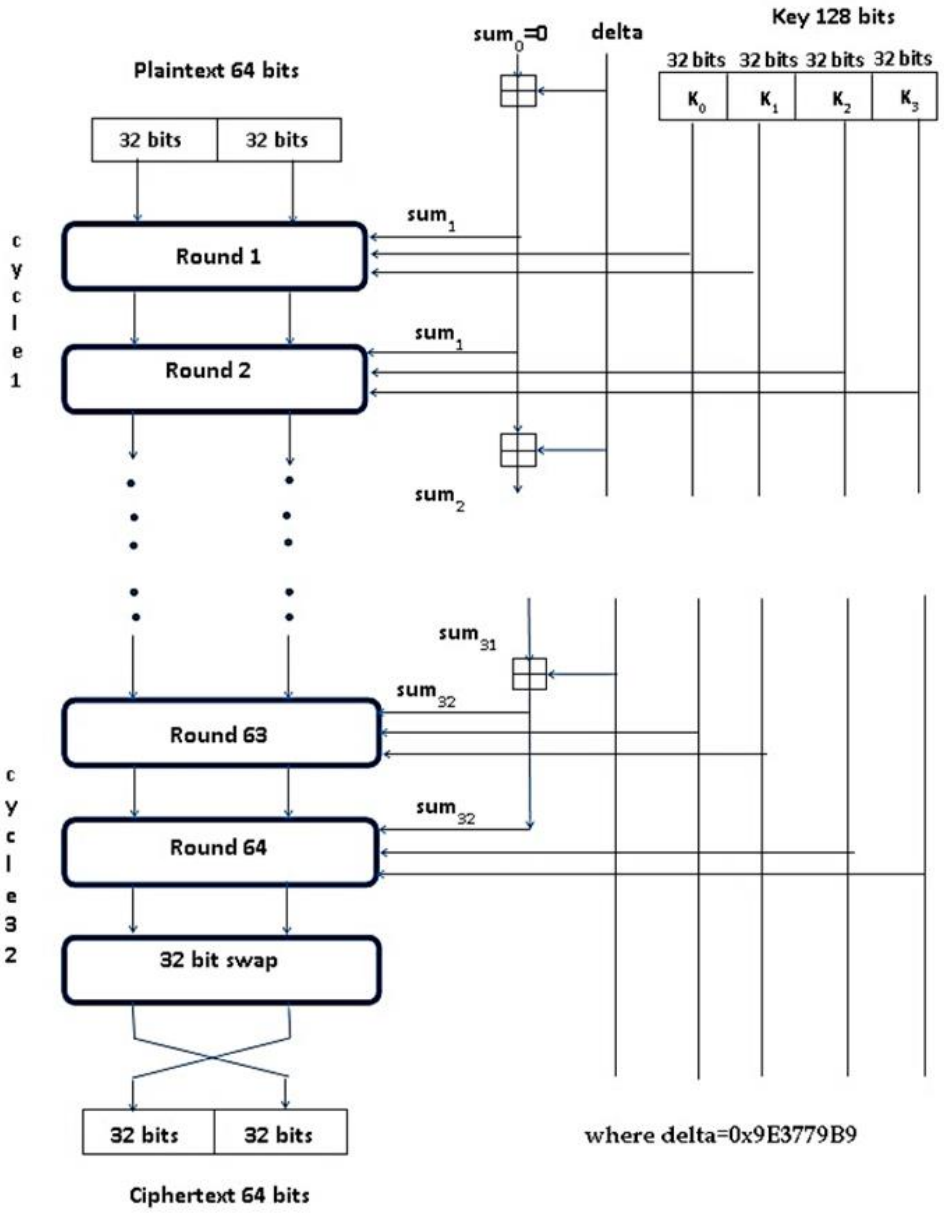

Figure 2.19: Encryption routine of the tiny encryption algorithm (TEA) [87].

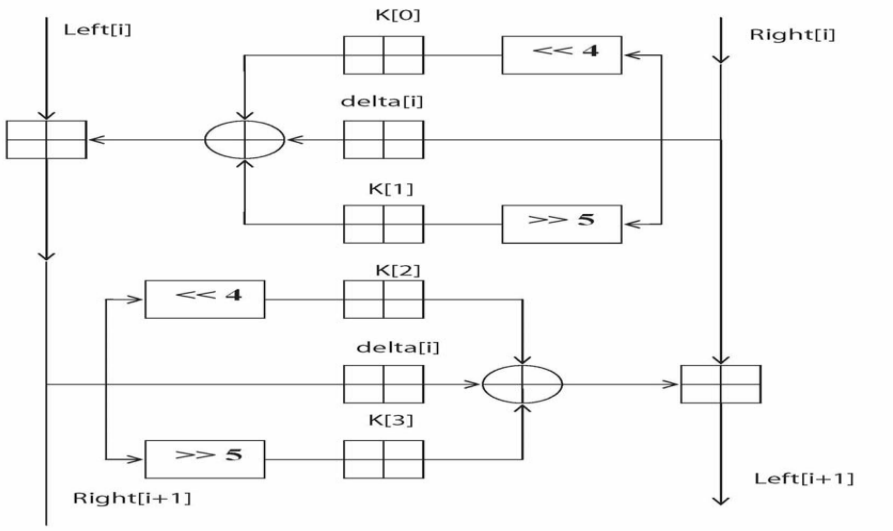

Figure 2.20: An abstraction of i-th cycle of TEA [87].

Figure (2.20) presents the internal details of the ith cycle of TEA. The round function, F, consists of the key addition, bitwise XOR and left and right shift operation [88]. We can describe the output (Left[i +1] , Right[i +1]) of the ith cycle of TEA with the input (Left[i] ,Right[i]) as follows

$$Left\ [i] = Left\ [i-1] + F\ (Right\ [I-1], key\ [0, 1], delta\ [i]),$$

$$Right\ [i] = Right\ [i-1] + F\ (Left\ [i-1], key\ [2, 3], delta\ [i]),$$

$$Delta\ [i] = Floor\ ((i + 1)/2)*delta$$

The round function F is

$$F(M,K[a,b],delta[i]) = ((M<<4)AND\ k[a])\ XOR\ (M\ AND\ delta[i])\ XOR\ ((M>>5)AND\ k[b]\ ).$$

The round function has the same general structure for each round but is parameterized by the round sub key K[i]. The key schedule algorithm is simple; the 128-bit key K is split into four 32-bit blocks K = (K[0], K[1], K[2], K[3]). The keys K[0] and K[1] are used in the odd rounds and the keys K[2] and K[3] are used in even rounds [88].

B. Decryption Routine

Decryption is essentially the same as the encryption process; in the decode routine the cipher text is used as input to the algorithm, but the sub keys K[i] are used in the reverse order [85], [86].

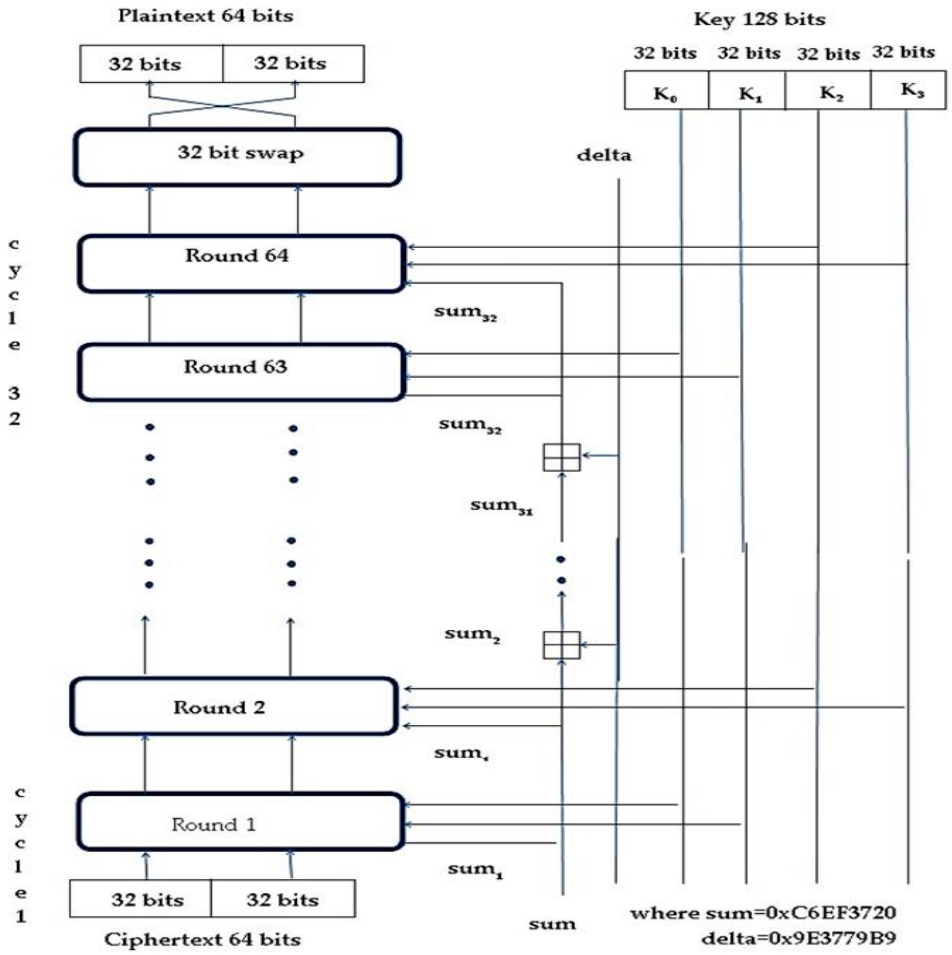

Figure 2.21: Decryption routine of TEA [87].

Figure (2.21) presents the structure of the TEA decryption routine. The intermediate value of the decryption process is equal to the corresponding value of the encryption process with the two halves of the value swapped [85], [86]. For example, if the output of the nth

encryption round is

ELeft[i] || ERight[i] (ELeft[i] concatenated with ERight[i]).

Then the corresponding input to the (64-i)th decryption round is

DRight[i] || DLeft[i] (DRight[i] concatenated with DLeft[i]).

After the last iteration of the encryption process, the two halves of the output are swapped, so that the cipher text is ERight[64] || ELeft[64], the output of that round is the final cipher text C. Now this cipher text is used as the input to the decryption algorithm. The input to the first round is ERight[64] || ELeft[64], which is equal to the 32-bit swap of the output of the 64^{th} round of the encryption process [85], [86].

2.5.4 Key Agreement

One of the most important aspects of cryptography is establishing a shared secret between two parties, a process known as key agreement or key exchange. A key agreement is a protocol or process used to generate a secret key in such a manner that is available for two parties [89]. The purpose of a key exchange system is to securely exchange cryptographic keys between two parties, preventing the keys from being obtained by anybody other than the people involved [90].

There are many different kinds of key exchange schemes, but some of the most common ones are Diffie–Hellman key exchange (DHE) and Elliptic-curve Diffie–Hellman (ECDH), RSA-OAEP and RSA-KEM (RSA key transport), PSK (pre-shared key), SRP (secure remote password protocol), FHMQV (fully hashed Menezes-Qu-Vanstone), ECMQV (elliptic-curve (quantum-safe key agreement) [91].

A. Elliptic-curve Diffie–Hellman (ECDH)

The Elliptic-curve Diffie–Hellman (ECDH) is a key agreement protocol that allows two parties, each having an elliptic-curve public–private key pair, to establish a shared secret over an insecure channel, as showed in figure (2.22). This

shared secret may be directly used as a key, or to derive another key. The key, or the derived key, can then be used to encrypt subsequent communications using a symmetric-key cipher. It is a variant of the Diffie–Hellman protocol using elliptic-curve cryptography [92], [93].

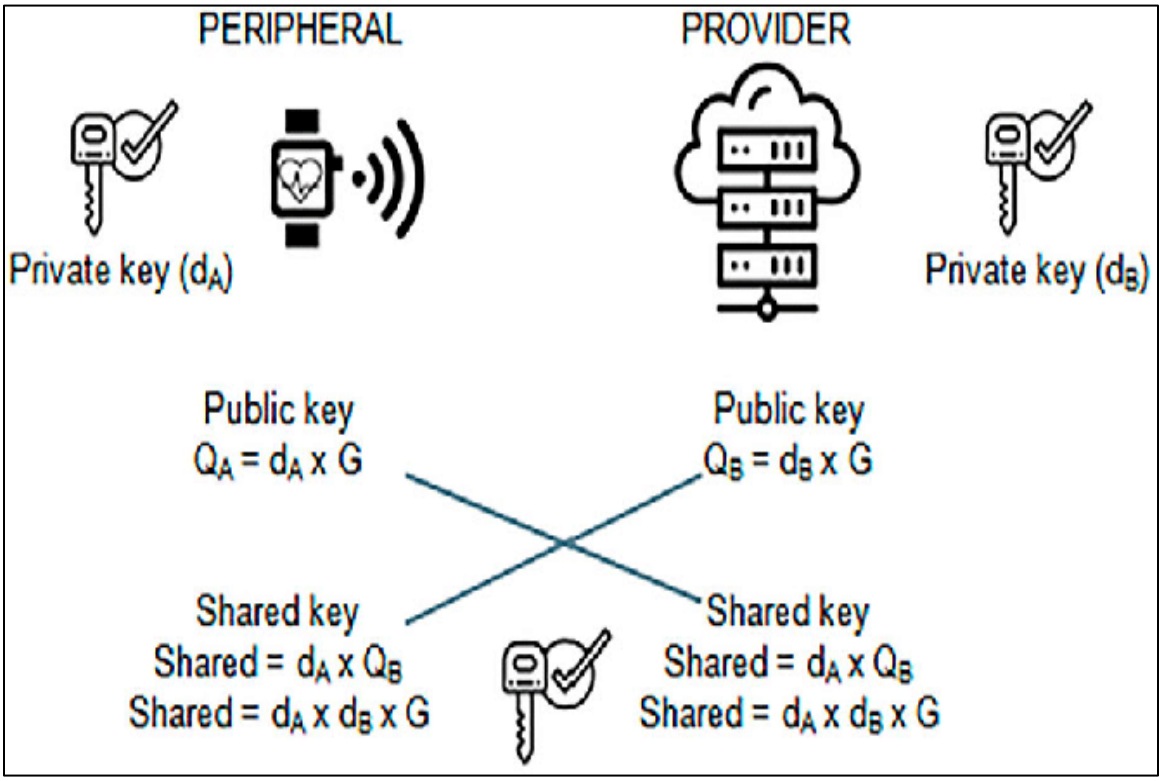

Figure 2.22: Elliptic-curve Diffie–Hellman (ECDH) work [92], [93]

2.6 Implementation Tools

The work of this thesis is based on the Raspberry Pi (RPi) as a fundamental construction component due to its characteristics: it is tiny, inexpensive, powerful, and completely adaptable. In addition, it's a programmable tiny computer board that comes pre-loaded with support for a wide variety of input and output ports, as well as a network connection. It is used in this work for processing and controlling sensing devices. Figure (2.23) showed the Raspberry Pi 4 model B 8GB that was used in this work.

Figure 2.23: The Raspberry Pi (RPi) 4 model B 8GB.

Besides, the Temperature sensor is used in this work to detect body temperature and its changes. Furthermore, the Oximeter (SPO2) sensor is used to determine heart rate and blood oxygen saturation. SPO2 results are reported as the percentage of hemoglobin that is saturated with oxygen. Figures (2.24) and (2.25) showed the Temperature and SPO2 sensor respectively.

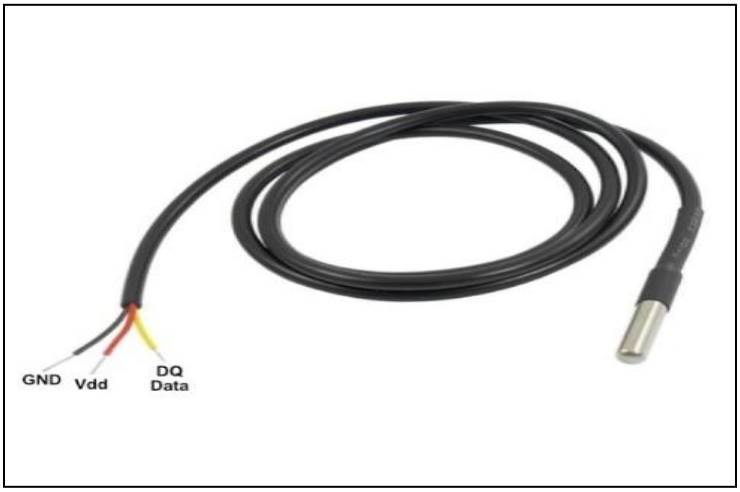

Figure 2.24: The Temperature (DS18B20 Waterproof) sensor.

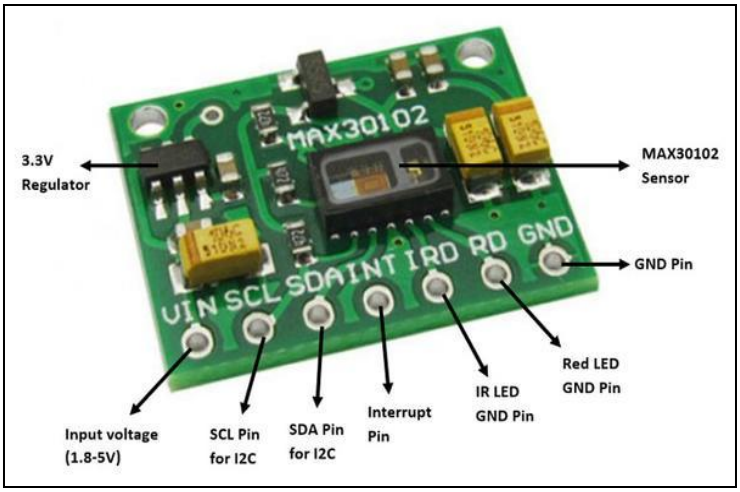

Figure 2.25: The SPO2 Sensor MAX30102.

The programming language that is used in this work is Python programming language, and it is used on both the Raspberry Pi device and the server device. Python is a computer language that is useful for a wide range of tasks and is both adaptable and powerful. The python is succinct and simple to understand, and easy to implement. Python can do any task that is set for it. This programming language may be used for a variety of tasks, including resource management, data analysis, data compression, and different approaches for network data analysis.

2.7 Evaluation Metrics

There are several evaluation metrics that will be used to evaluate and test the proposed cryptosystem:

2.7.1 Data Compression Ratio (DCR)

Compression Ratio is a way used in this thesis to measure how much data representation can be compressed before being transferred. The most common way to describe this is the ratio of compressed to uncompressed data size. Equation (2-1) is used to calculate the DCR [94].

$$DCR\ (\%) = \left(\frac{size\ of\ data - size\ of\ Compressed\ data}{size\ of\ data}\right) * 100\% \quad \ldots.(2\text{-}1)$$

2.7.2 Throughput

It is used in this thesis to measure the maximum quantity of data that can be sent from source to destination in a particular length of time through an internet connection. Equation (2-2) is used to calculate the throughput [95].

$$Throughput\ = \left(\frac{Number\ of\ send\ data}{Time}\right) \quad \ldots.(2\text{-}2)$$

2.7.3 Execution Time

It is used in this thesis to measure how much time it takes to execute (compression and decompression processes, encryption and decryption processes, and transfer data). Equation (2-3) is used to calculate the execution time [96].

$$Execution\ time = End\ Time\ -\ start\ Time \quad \ldots.(2\text{-}3)$$

2.7.4 Entropy

The most significant indication of information randomness is the information entropy, which is defined and used to quantify the unpredictability of data. It is defined as the anticipated average scale of information from the data after encryption. Equation (2-4) is used to calculate the entropy [97].

$$Entropy = -\sum_{i=0}^{m} pi\ log_2(pi) \quad \ldots..(2\text{-}4)$$

2.7.5 Memory Usage

Memory usage is the space that is reserved for the implementation of the algorithms. It mainly depends on the number of operations in algorithm, initialization vectors and the type of mode operations [98].

2.7.6 Avalanche Effect

Avalanche effect means any change in plaint text or key leads to change in cipher text. Equation (2-5) is used to calculate the avalanche effect by using the Hamming distance [99].

$$Avalanche\ effect = (\ hamming\ distance\ \div\ file\ size\ (in\ bytes) \quad \ldots (2\text{-}5)$$

The hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. It can also be calculated using Equation (2-6) [99].

$$Avalanche\ effect = \frac{no.of\ flipped\ bits\ in\ ciphered\ text}{no.of\ bits\ in\ ciphered\ text} * 100 \quad \ldots (2\text{-}6)$$

2.8 Summary

This chapter discussed the main concepts of the Internet of things (IoT). It explains data compression and the big stream IoT data and how to benefit from compression. It exhibits the security issue of IoT and how to benefit from lightweight techniques to secure data of IoT. Thus, the Compcrypt system based on the data compression and Lightweight encryption in IoT environment proposed to provide a promising way for secure IoT data and increase the speed data transmitted in this thesis.

Chapter Three

The Proposed Compcrypt system

CHAPTER THREE

THE PROPOSED COMPCRYPT SYSTEM

3.1 Introduction

This chapter explains the details of the proposed Compcrypt system to secure IoT data and save network bandwidth. The Zstandard compression algorithm and Tiny Encryption Algorithm (TEA) are adopted to design and implement the Compcrypt approach. In addition, the Compcrypt system use ECDH key exchange to generate a shared secret key. The main steps of the proposed Compcrypt system consists of the following: (1) generating and collecting sensor data; (2) the key generation phase; (3) collecting and compressed the sensor data; (4) data encryption and sending to the server; (5) receive data from client node and data decryption; (6) data decompress to be ready for analysis and evaluation.

3.2 The proposed Implementation Requirements

The proposed system installation is based on the IoT devices such as a temperature sensor (DS18B20 Waterproof Digital Thermal Sensor), a heart rate and oxygen sensor (pulse Oximeter (SPO2) Sensor MAX30102), and the management device as a (Raspberry Pi (RPi 4 model B 8GB)), and server (Computer system with Windows operating system), the main steps of implementation requirements in the proposed system sorted as follow:

1- Configuration sensors devices with (network settings, ports, and addresses)
2- Configuration Raspberry Pi (Programming core of addresses, import libraries, and medium configuration).
3- Configuration the Server (connecting sockets, ports, and addresses translation).

3.3 The proposed Methodology

The proposed system method is based on combining data compression and a lightweight cryptographic technique called the Compcrypt approach. It uses the Zstandard algorithm for compressing and decompression data, while it uses the Tiny Encryption Algorithm (TEA) for encrypting and decryption data, and shared secret key is generated by using the Elliptic-curve Diffie–Hellman (ECDH) key exchange protocol. It works at the client-side level to compress and encrypt their readings efficiently way to minimize the amount of data transmitted to the server, save IoT network bandwidth, decrease network overloading, and also protect IoT data. The general view of the proposed system is shown in Figure (3.1).

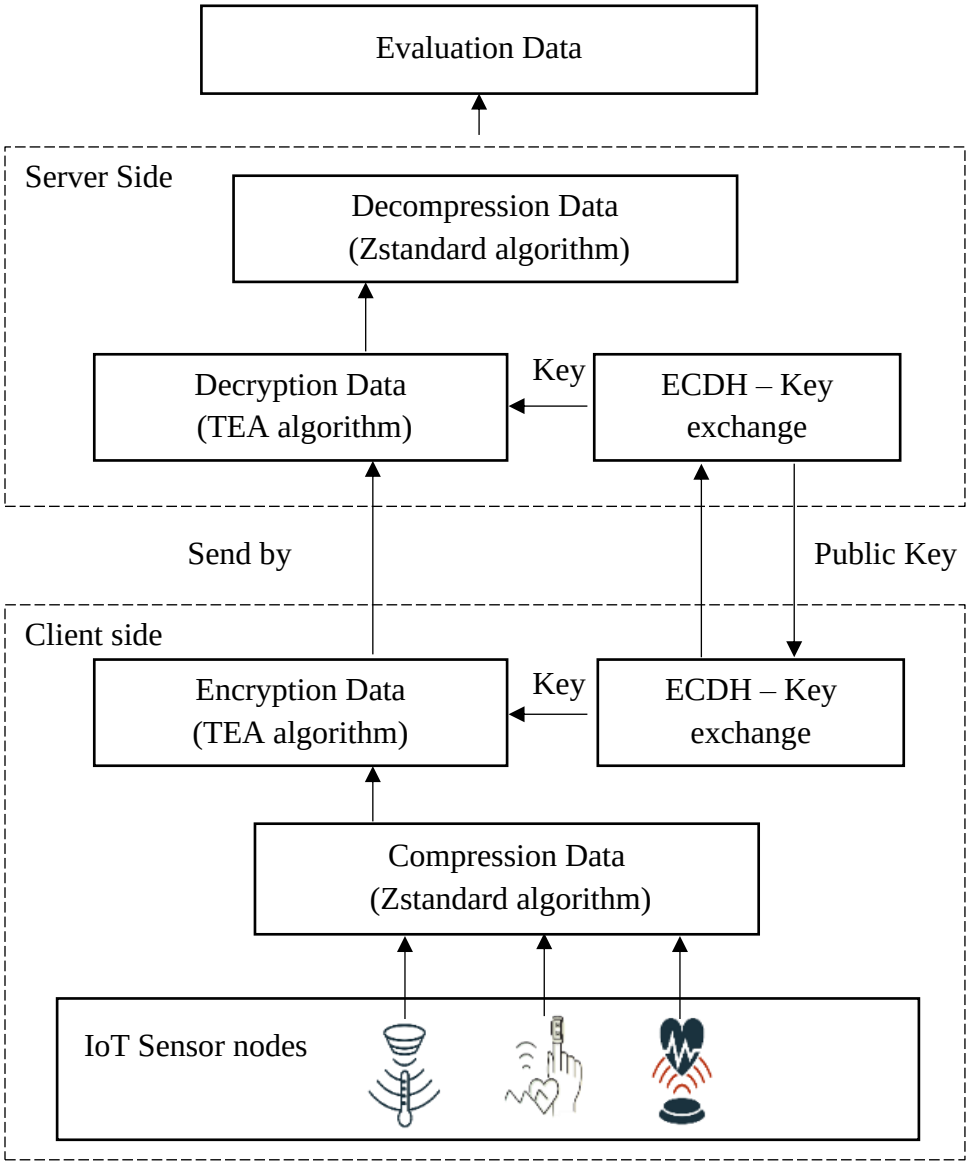

Figure 3.1: The general view of the proposed system.

In the first step of Compcrypt system in client node collects data from the IoT sensor to compress them, in the second step Compcrypt exchange the public keys to generate shared secret key. The third step using the secret key and compressed data as input data for encryption step. In the fourth step, the data is redirected to the server node for decompressed / decryption steps and then evaluate the results.

So, the proposed Compcrypt system is responsible for protecting and minimizing the quantity of sensor data that is sent through the network. Therefore, the security side of the proposed method which is based on the TEA encryption method employed to protect sensor data from unauthorized access and hackers as it makes the sensor reads unclear and encrypted cipher text, it is difficult to break them through transmission journey. In addition, the Zstandard compression algorithm is utilized to encode and reduced the volume of this data, so it added another level for security approach through encoding incoming sensor reads.

The used IoT sensors in the proposed Compcrypt system are Temperature Sensor, and Pulse Oximeter (SPO2) sensors are used to measure the body's temperature, oxygen levels, and heart rate at sensor node level, while the Raspberry Pi will collect this data and processing them, and then transfer this data to a server via the wireless network medium. As a result, these details play a significant part in the server's decision-making. Figure (3.2) shows the block diagram of the proposed Compcrypt system implementation.

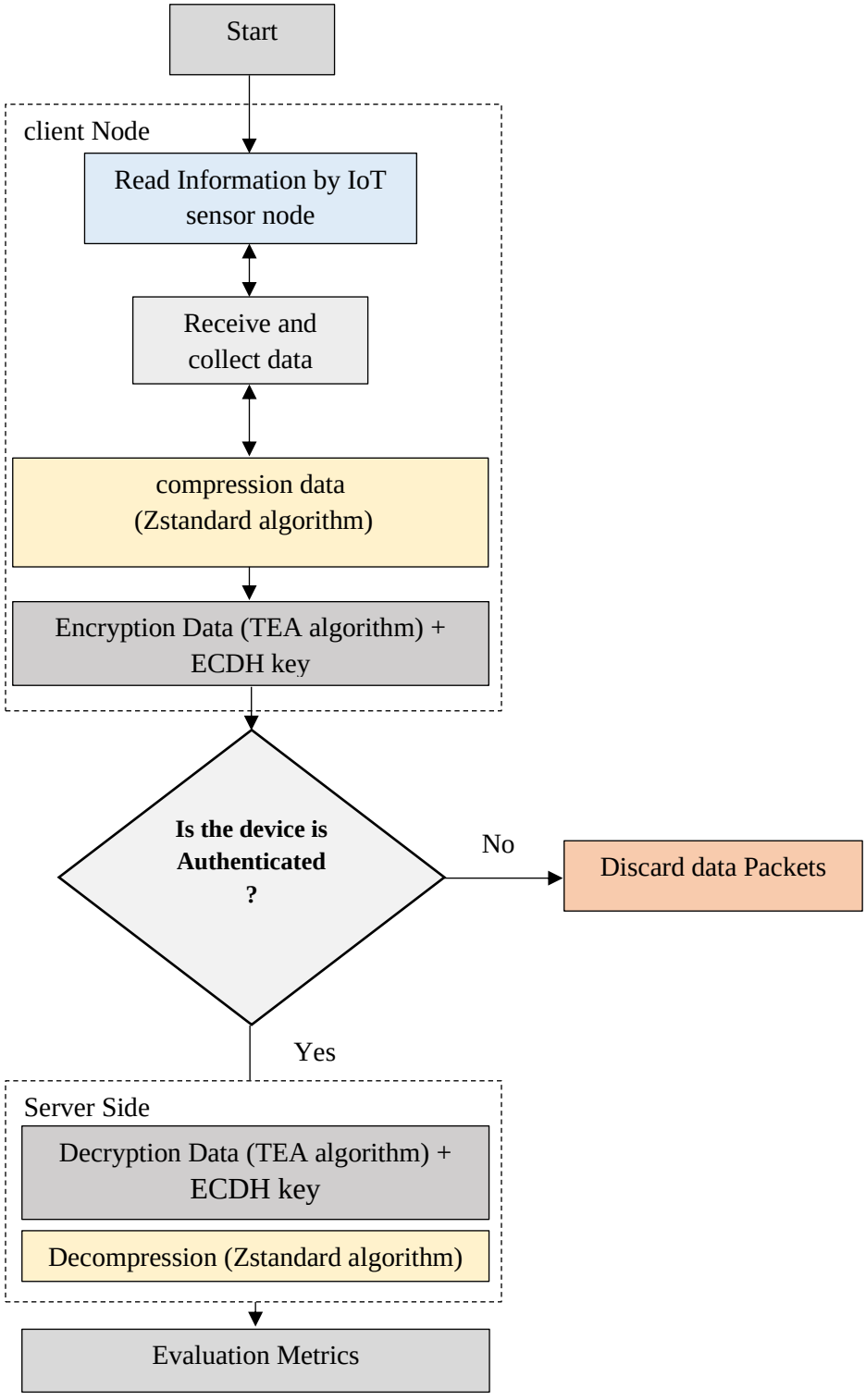

Figure 3.2: The flowchart of proposed Compcrypt system steps.

The proposed Compcrypt system has several main steps, as follows:

3.3.1 Reading sensor data

The sensors are initialized to read stream of data from the patient's body. Several IoT sensors, including Temperature, High-Sensitivity Pulse Oximeter and Heart-Rate Sensor, were connected to the Raspberry Pi, which receives the data from them and collects sensing information as data messages showed as follow:

A- **Temperature Sensor**: An integer value minimum sensor read is (-55 degrees), and maximum is (+125 degrees).

B- **High-Sensitivity Pulse Oximeter and Heart-Rate Sensor:** The MAX30102 is integrated pulse oximetry and heart-rate monitor module. Communication is through a standard I2C-compatible interface.

3.3.2 Key exchange using Elliptic-curve Diffie–Hellman (ECDH) protocol

The proposed Compcrypt system based on Elliptic Curve Diffie Hellman protocol to enable two parties for public key pair exchange, to agree on a shared secret key approach. A key is used to encrypt and decrypt whatever data is being encrypted/decrypted.

In this work, a key generator using the Elliptic Curve Diffie Hellman protocol is used with TEA lightweight encryption algorithm. The reason beyond using this protocol is based on the Elliptic Curve Cryptography (ECC) is modern family of public-key cryptosystems. The ECC uses smaller keys and signatures than RSA for the same level of security and provides very fast key generation, fast key agreement and fast signatures.

The process of ECDH consists of the following steps: Firstly, each client (Raspberry pi) and server has the private key and an ECC elliptic curve with

generator point G. Secondly, both the client and the server generate the public key by multiplying the private key with G, and then exchange their public key over an insecure channel. Thirdly, then both client and server can derive a shared secret key by multiplying the received public key with your private key.

Figure (3.3) and algorithm (3.1) showed the process of Elliptic-curve Diffie–Hellman key exchange protocol. Where G generator point, d_A, d_B private key, Q_A, Q_B public key, K, K' shared secret key.

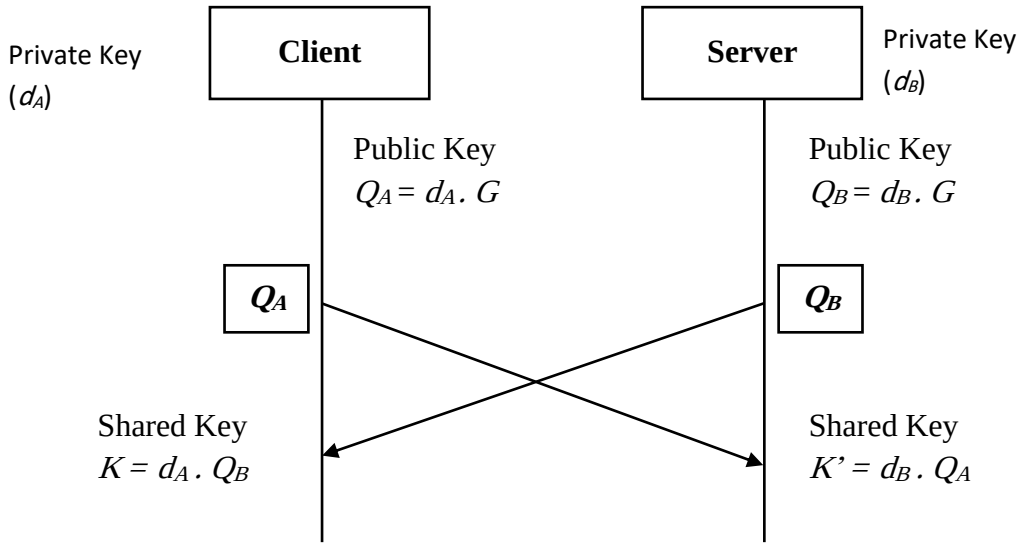

Figure 3.3: The process of Elliptic-curve Diffie–Hellman protocol.

Algorithm 3.1: Elliptic-curve Diffie–Hellman algorithm

Input: Domain parameter (G)

Output: private key: d_A, d_B , public key: Q_A, Q_B ,and shared key : K, K'

1. System Initialization;
2. Choose an integer d_A, $d_B \in [1, n - 1]$;
3. Client A computes $Q_A = d_A . G$ send to server B
4. Server B computes $Q_B = d_B . G$ send to client A
5. Client A calculate $K = d_A . Q_B = d_A (d_B . G)$
6. Server B calculate $K' = d_B . Q_A = d_B (d_A . G)$

End of algorithm

3.3.3 Compression data using Zstandard Algorithm

The compression process in the Raspberry Pi compresses the data collected by sensors using the Zstandard algorithm, resulting in compressed data (CD). The used Zstandard (zstd) provides a fast and high ratios of lossless data compression. The zstd in the proposed system implemented in Python programming language within Client hardware device to compress smaller and faster, and it selected in the proposed system due to it easy to use and simple installation requirements with less computation power resources (RAM, Main Memory, CPU, Execution Time). In addition, it depends on standard dictionary-matching stage (LZ77) combined with a large search window and a fast entropy coding stage, as well as it consists of both Finite State Entropy and Huffman coding. Figure (3.4) showed the block diagram of the proposed compression data with Zstandard method.

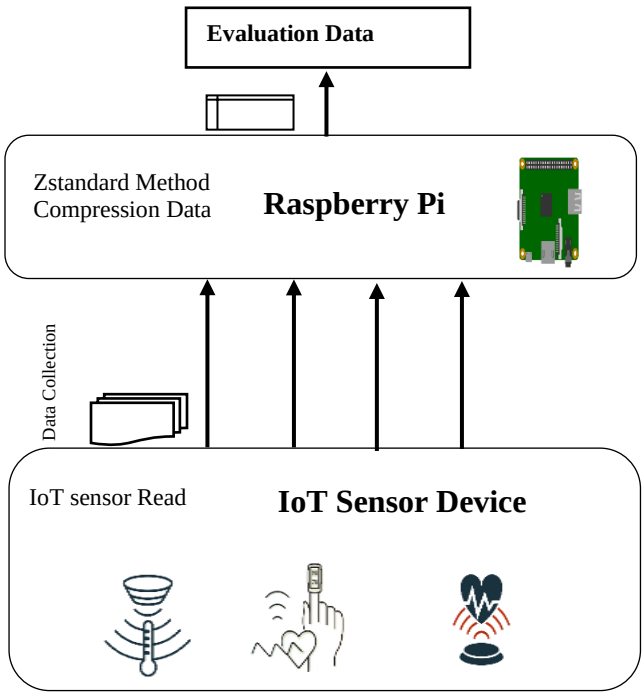

Figure 3.4: Compression Data with Zstandard Method

Algorithm (3.2) the used Zstandard algorithm compression. As mentioned, the results compressed data is encoded and smaller size compared with the original data.

Algorithm 3.2: Algorithm for data compression in client node
Input: Generic Numerical Sensor Data **Output**: Compressed Numerical data 1. System Initialization; 2. Receive data packets from IoT sensor nodes; 3. **if** DeviceID connected= True then 4. **Set** T timeperiod 5. Create array for input data collection; 6. **while** data receiving **do** 7. Take data in a certain range; 8. **for** Rangeeach **do** 9. Sort the input data in ascending order;

10.	**while** T timeperiod **do**
11.	Calculate the dictionary of each pair of values;
12.	Store mean values in a file;
13.	Send files towards to the second encryption state of Compcrypt system;
14. else	
15. Discards Packet	
End of algorithm	

3.3.4 Encryption data using Tiny Encryption Algorithm (TEA)

The encryption data is applied in the Raspberry Pi, which encrypts the compressed data (CD) outputted from the compression phase of Compcrypt system using the Tiny Encryption Algorithm (TEA), resulting is unreadable (ciphertext) data depending on Elliptic Curve Diffie-Hellman (ECDH) key exchange protocol to agree on a shared key to be used in this algorithm. The proposed Compcrypt system based on TEA algorithm due to its simplicity of description and implementation, typically a few lines of code, it provides a lightweight secure code to be equivalent to work in resource constraints environment like IoT sensor network, as it designed to minimize the memory resources, and to maximize encryption speed. Algorithm (3.3) the used TEA Encryption Algorithm. Figure (3.5) showed the encryption state with TEA of Compcrypt system.

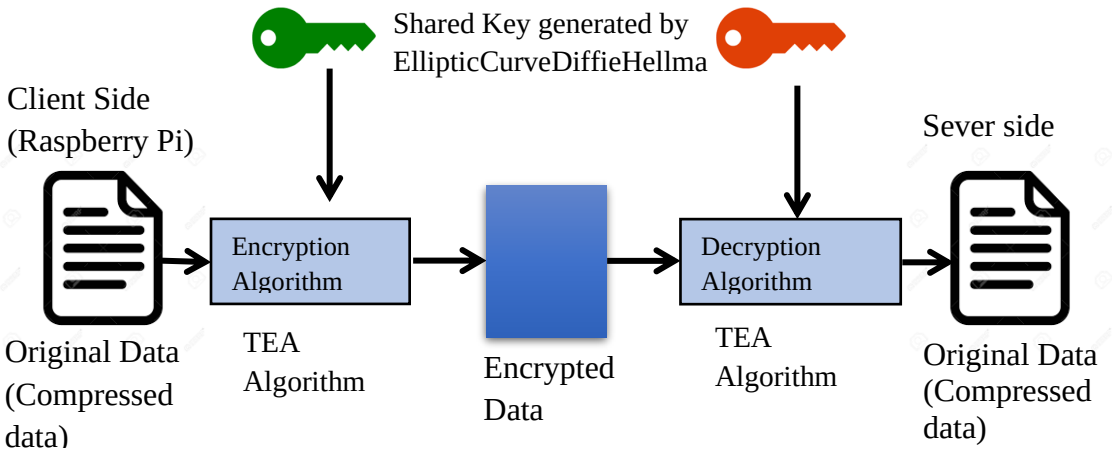

Figure 3.5: The used TEA encryption and decryption algorithm approach.

Algorithm 3.3: TEA Encryption Algorithm

Input: Compressed data from Compcrypt system+shared secret key

Output: Encrypted data (Ciphertext)

1. System Initialization;
2. Receive compressed data packets from Compcrypt system;
3. **If** data compressed= true then
4. Get keyvalue
5. Set encryptvalue
6. **while** data receiving **do**
7. Build encrypt ciphertext;
8. **for** Rangecompresseddata **do**
9. Make finalcipher from incoming compressed data;
10. **while** stillreceiving **do**
11. update ciphertext;
12. Store ciphertext in a file;
13. Send files towards to the server-side level;

14. Else

- Waiting Compcrypt system for compressed data

End of algorithm

3.3.5 Decryption data using Tiny Encryption Algorithm (TEA)

Decryption data is implemented in the server side as the inverse process of Compcrypt system. As it is essentially the same as the encryption process; in the decryption routine the cipher text passed from the Raspberry Pi is used as input to the decryption algorithm, with the same secret keys in the encryption process which is generated from Elliptic Curve Diffie-Hellman key exchange protocol. Figure (3.5) above showed the decryption with TEA of Compcrypt system in the server node. Algorithm 3.4 showed the used TEA Decryption Algorithm within the Compcrypt system in server.

Algorithm 3.4: TEA Decryption Algorithm

Input: Encrypted(ciphertext) data + shared secret key

Output: Decrypted data (encoded or compressed data)

1. System Initialization;
2. Receive encrypt data packets from client node;
3. **if** DeviceID connected= True then
4. Get keyvalue
5. Set decryptvalue
6. **while** data receiving from client **do**
7. Build decrypt data;
8. **for** Rangeencryptdata **do**
9. Make finalresult from incoming encrypt data;
10. **while** stillreceiving **do**
11. update encoded value;
12. Store encodedtext in a file;
13. Send files towards to Compcrypt system;

14. Else

- Waiting client node for encrypt data

End of algorithm

3.3.6 Decompression data using Zstandard algorithm

The decompression process implemented in server; it receives data from the decryption phase of Compcrypt system. It used of a Zstandard algorithm to decompress the compressed data for returning the data message to its original form. Next, the data is analyzed and stored on the server. Figure (3.6) showed the block diagram of the proposed decompression data with Zstandard method. The used Algorithm (3.5) for Zstandard Decompression algorithm in server.

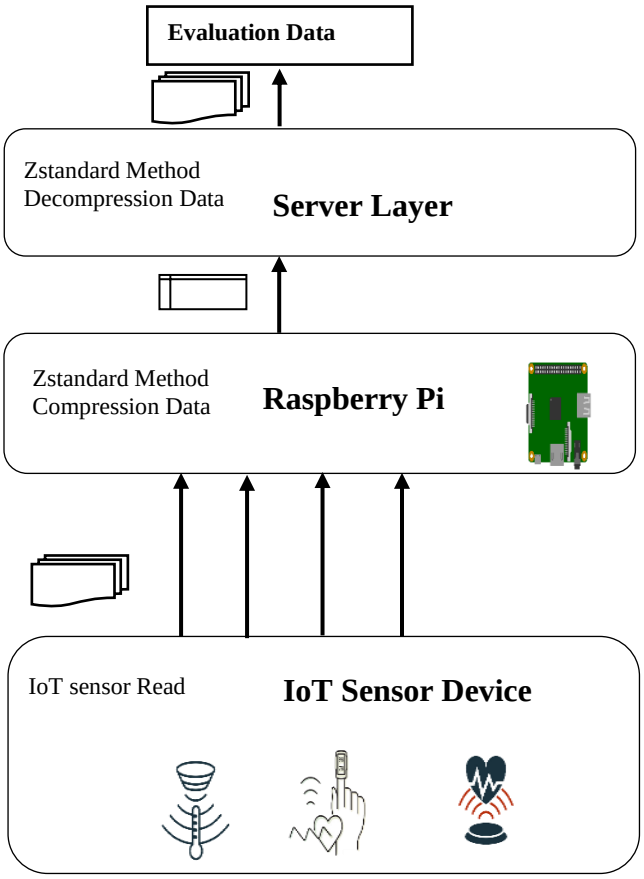

Figure 3.6: Decompression Data with Zstandard Method.

Algorithm 3.5: Zstandard Decompression
Input: Compressed data **Output**: Decompressed Output data 1. System Initialization; 2. Receive request from client device; 3. **If** data compressed= true then 4. **while** Requestapp **do** 5. Get file according to request; 6. Regenerate data from file Create array for regenerated data; 7. **for** Rangeeach **do** 8. Generate each dictionary value form Compcrypt system; 9. Store values in a temporary file; 10. Send the results to evaluation; **11. Else** 12. Waiting Compcrypt system for decrypt data
End of algorithm

3.4 Summary

This chapter explain the Compcrypt approach based on Zstandard compression algorithm and TEA encryption algorithm which use ECDH protocol for generation secret key. It deals with different data sizes. The proposed Compcrypt responsible to minimize the amount of data transmitted to the server, save network bandwidth, decrease network overloading, and also protect IoT data.

Chapter Four

The Implementation and Results

CHAPTER FOUR
IMPLEMENTATION AND RESULTS

4.1 Introduction

This chapter presents the implementation and results of the proposed Compcrypt system introduced in Chapter Three. In addition, the chapter shows the results of the implementation based on the main four case studies. The 1st case study based on the TEA lightweight encryption algorithm. The 2nd case study dealt with the compression Zstandard algorithm. While, the 3rd case study based on combining the model which compressed the data encrypted. The 4th case study is based on the Compcrypt system which encrypted the compressed data. Furthermore, the proposed system evaluated with network and security parameters. Finally, the used system is compared with previous related works.

4.2 The proposed system Implementation

The proposed system is implemented based on the core device as the microcontroller Raspberry pi, supported devices as temperature sensor, and SPO2 sensor. The implementation results show the sensing information are collected by the core device and it collected the generated healthcare data by IoT sensor devices, then, it passed them into server for data analysis for later processing. Table (4.1) contains information on the specifications of the used hardware devices.

Table 4.1: Specification of Devices Nodes.

Device Type	Device Model	Supply voltage	Product Size
Raspberry Pi	4 model B 8GB	DC 5V/2.5A	82 * 56 * 19.5
Temperature Sensor	DS18B20 Waterproof	3.0V to 5.5V	6 * 50
Oxygen and Heart Rate Sensors	SPO2 Sensor MAX30100	3.3V	5.6 * 2.8 * 1.2
Server	Dell Inspiron 5559	19.5 V	

In addition, the proposed general view of Compcrypt system components is showed in Figure (4.1)

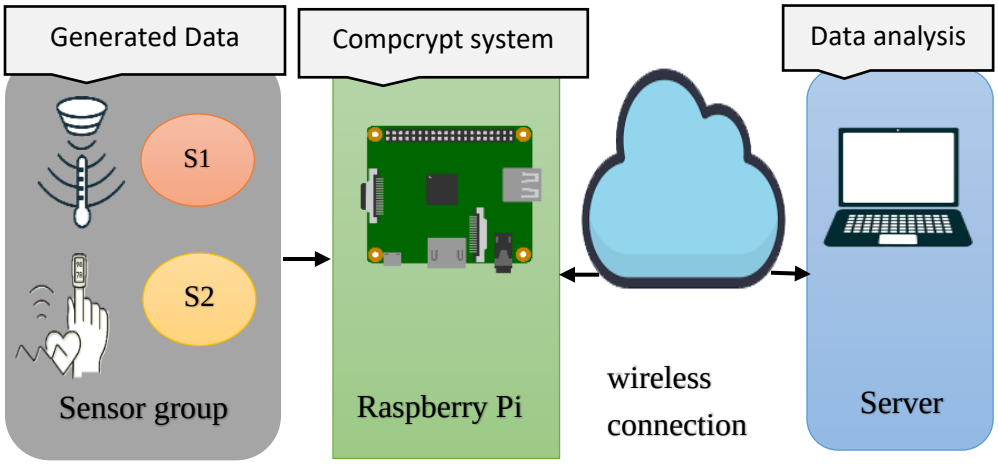

Figure 4.1: Implementation of the proposed system.

4.3 The proposed System Results

The following results based on four case studies as the 1st case study is based on data encryption with a lightweight encryption algorithm (TEA algorithm), the 2nd is based on data compression with the Zstandard algorithm, the 3rd is based on the case of compressed data with the Zstandard algorithm after data encryption, and the 4th is based on the case of compressed data with the Zstandard algorithm before data encryption.

4.3.1 The 1st case study Results

The proposed system results of security system based on TEA lightweight algorithm to encrypt and decrypt data traffic depending on different sensor numbers as (20 sensors to 10000 sensors) and different size of plaintext from 255 to 121632 bytes. The results showed the increase number of sensor nodes generated increase data traffic and the required time for encrypt high data rate is bigger, in addition the throughput is increased. Table (4.2) shown the results of encryption with TEA algorithm.

Table 4.2: The results of Encryption Data using TEA algorithm.

No. of sensor	Size of Data in byte	Size of Encryption Data in byte	Encryption Time in ms	Throughput in byte/ms
20	255	544	1.8	142
100	1228	2480	7.5	164
1000	12175	24384	57.5	212
6000	72993	146016	310	235
10000	121632	243296	500	245

Moreover, Figure (4.2) shown the evaluation metrics of data size before and after the TEA encryption algorithm.

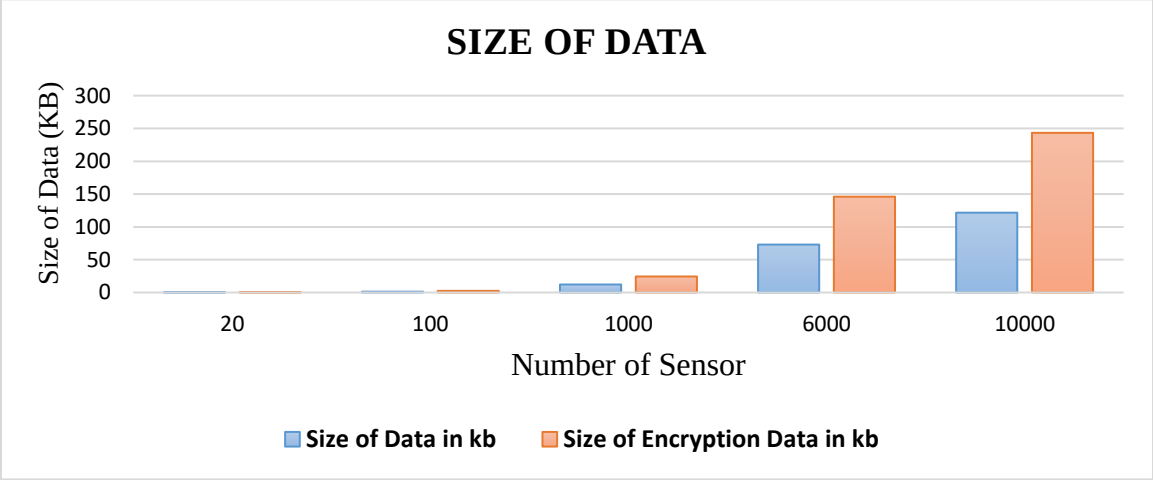

Figure 4.2: Encryption different data size with TEA algorithm.

Figure (4.3) showed the required time of TEA encryption algorithm, the time is increased due to the number of sensors is increased and the amount of generated sensor reads also increased.

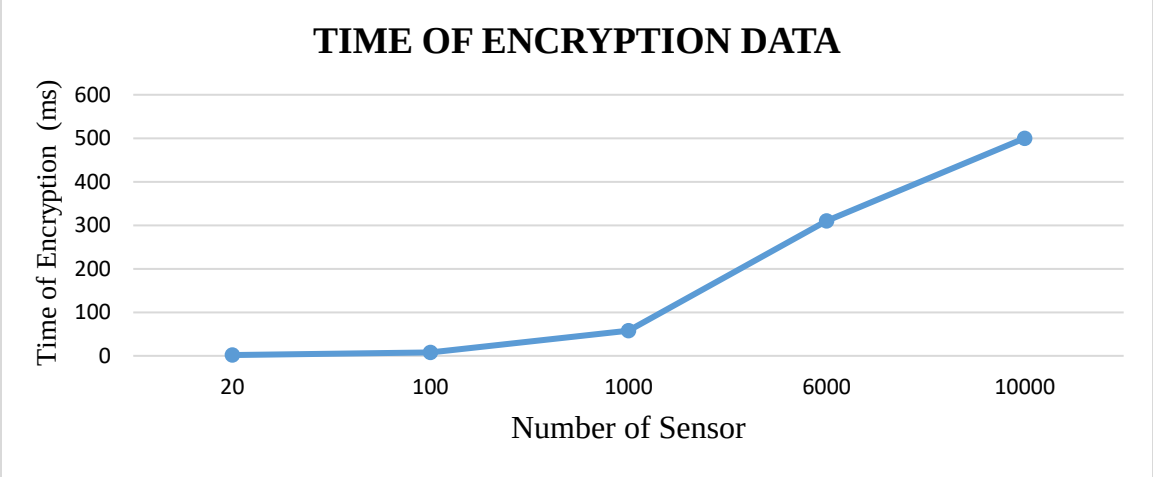

Figure 4.3: Encryption time of TEA algorithm.

Figure (4.4) showed that the throughput value is decreased due to the required time of data encryption is increased because of the increased sensor numbers.

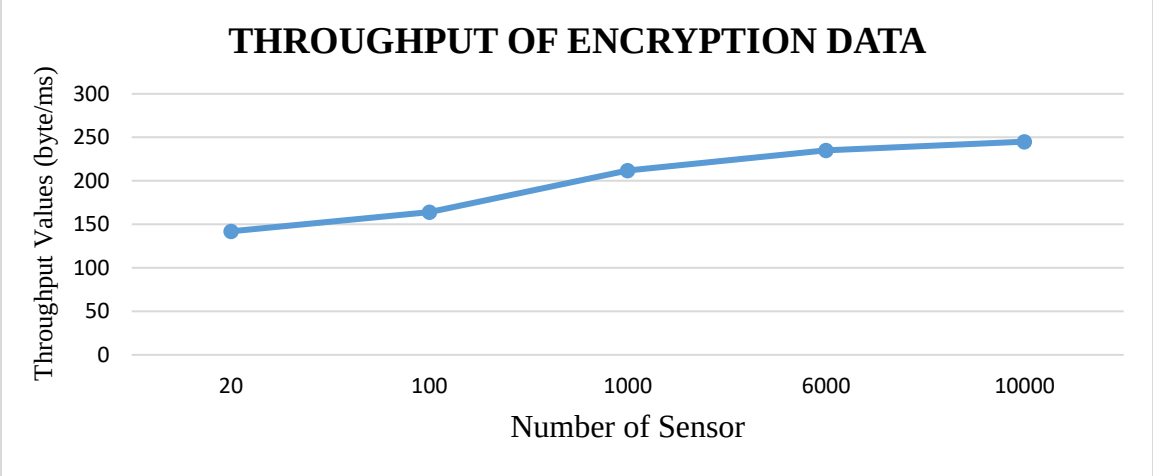

Figure 4.4: Throughput Encryption Value of TEA algorithm.

Furthermore, the decryption results showed that the sensor number increased effects on required time for decryption, and the throughput is

increased due to the increased number of IoT sensor nodes. Table (4.3) showed the results of decryption with TEA algorithm.

Table 4.3: The results of Decryption Data using TEA algorithm.

No. of sensor	Decryption Time in ms	Throughput in byte/ms
20	2	150
100	7	175
1000	54.5	224
6000	290	252
10000	480	253

Figure (4.5) showed the required time for decryption data depending on the different number of sensors.

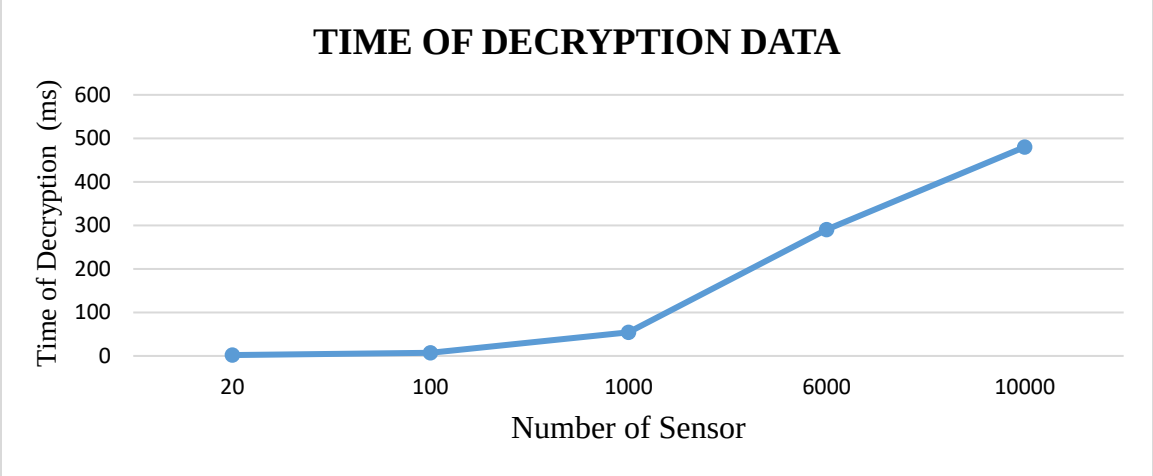

Figure 4.5: Decryption Time of TEA algorithm.

Figure (4.6) showed the throughput of decryption data, the maximum throughput is showed in state of 10000 sensor device with 253 byte/ms.

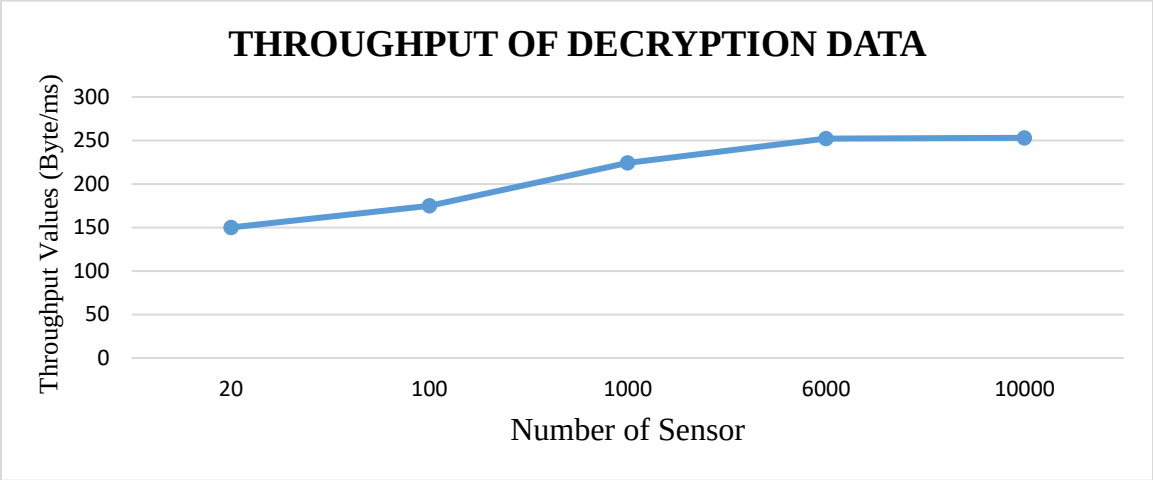

Figure 4.6: Throughput Decryption Value of TEA algorithm.

Table (4.4) showed the computation processes of the encryption algorithm as the delay is increased due to the increased number of sensors and size of input data, as well as RAM usage are changed depending on the process computation.

Table 4.4: Delay, and RAM through TEA encryption algorithm.

No. of sensor	**Delay in ms**	**RAM in Mbps**
20	4	12.976
100	20	13.514
1000	116	14.019
6000	675	14.265
10000	990	16.519

Figure (4.7) showed the value of delay (latency) in TEA lightweight encryption algorithm. The value is increased due to increased number of sensors and size of input data.

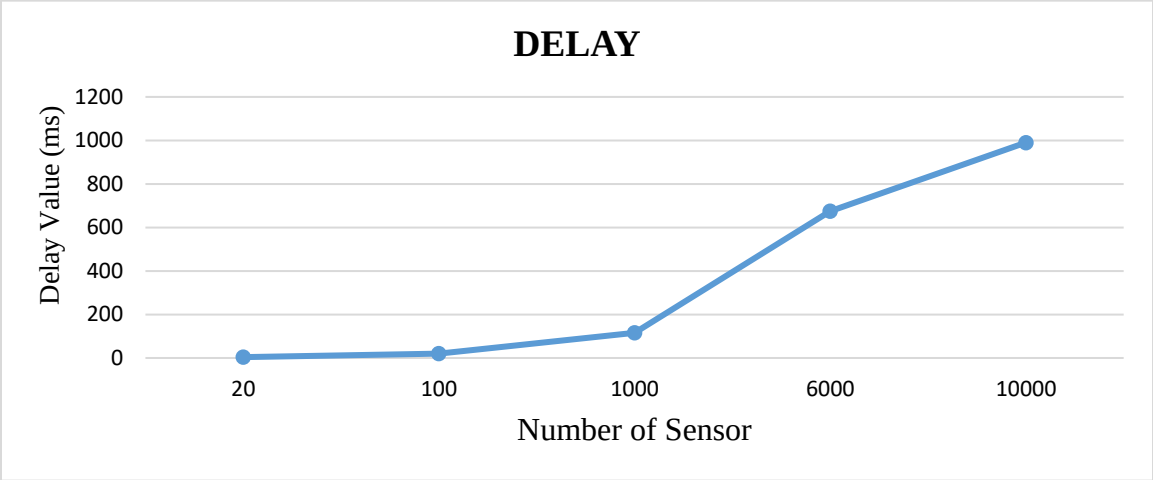

Figure 4.7: Delay Value of TEA algorithm.

Figure (4.8) showed an effective result of the RAM usage. Notice a change in RAM values due to the available space and of software running in at the runtime of implementation of cryptosystem.

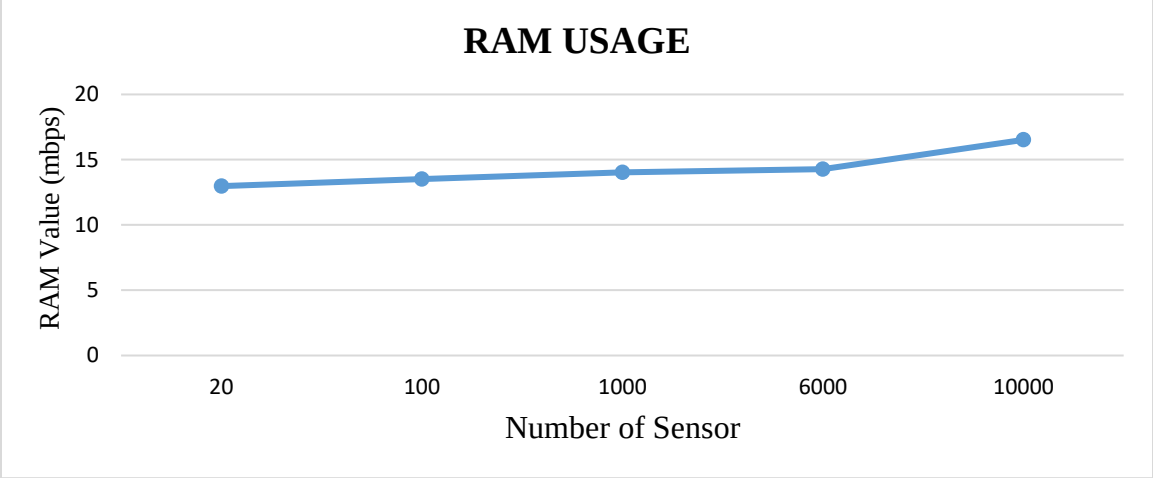

Figure 4.8: RAM usage of TEA algorithm.

4.3.2 The 2nd case study Results

This case based on the proposed Zstandard compression algorithm only which showed in Table (4.5). It explained as the increased number of sensors effect on generated data from sensors, and when size of data traffic is increased the size of compression is compressed to small amount of data.

Table 4.5: The results of Compression Data using Zstandard algorithm.

No. of sensor	Size of Data in byte	size of Data after Compression in byte	Compression Ratio in %	Execution Time in ms	Throughput in byte/ms
20	255	114	55%	0.8	320
100	1228	455	62%	3	491
1000	12175	3587	70%	20	608
6000	72993	15336	78%	100	730
10000	121632	22730	81%	150	811

Figure (4.9) showed the size of data increased with increased number of sensors, and the compression state decreased size of original data to be lightweight to the resource constraints of sensors.

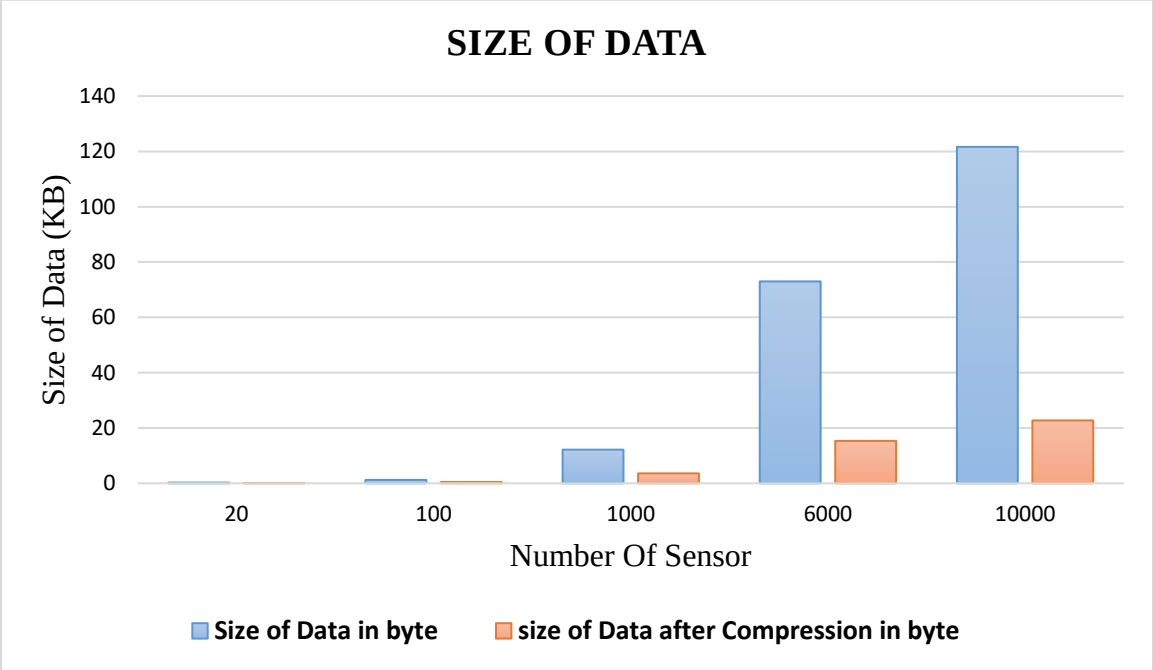

Figure 4.9: Compression different data size with Zstandard algorithm.

Figure (4.10) showed the Zstandard compression ratio among data created by sensors, and it explained that the compression ratio increased due to the data generated also increased.

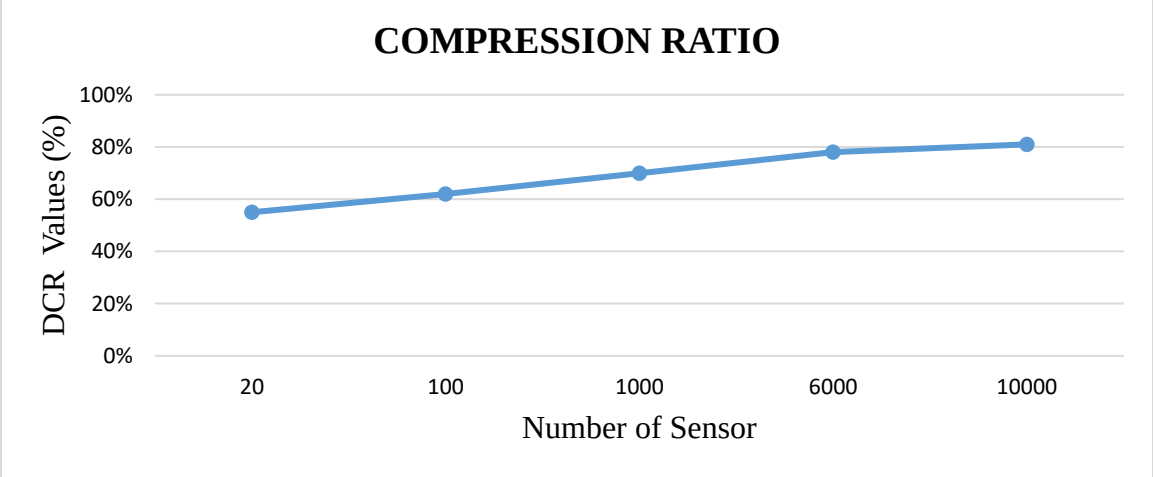

Figure 4.10: Compression Ratio with Zstandard algorithm.

Figure (4.11) showed the execution time of Zstandard algorithm with different number of sensors.

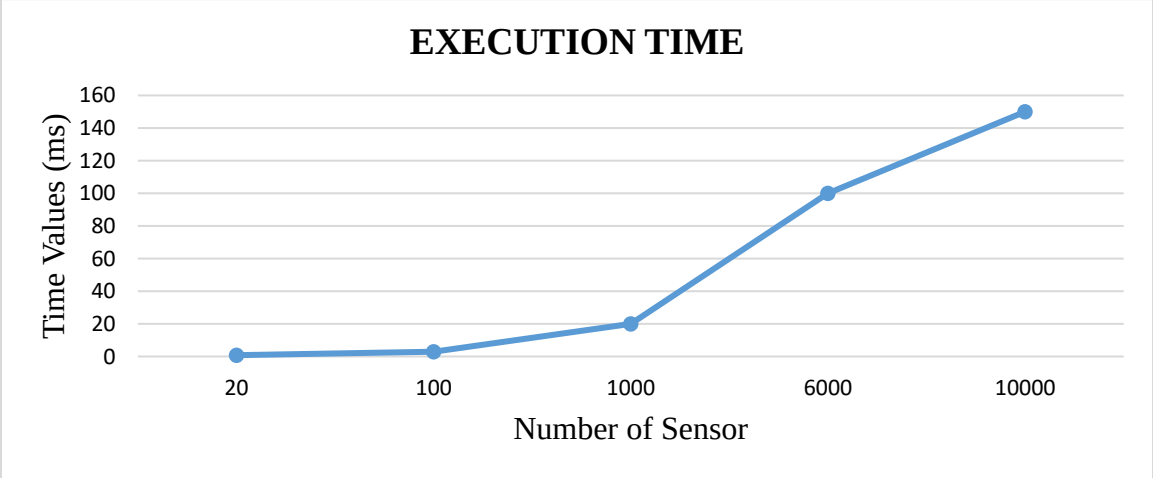

Figure 4.11: Execution Time of Zstandard algorithm.

Figure (4.12) showed the Zstandard throughput for different incoming data values from sensors.

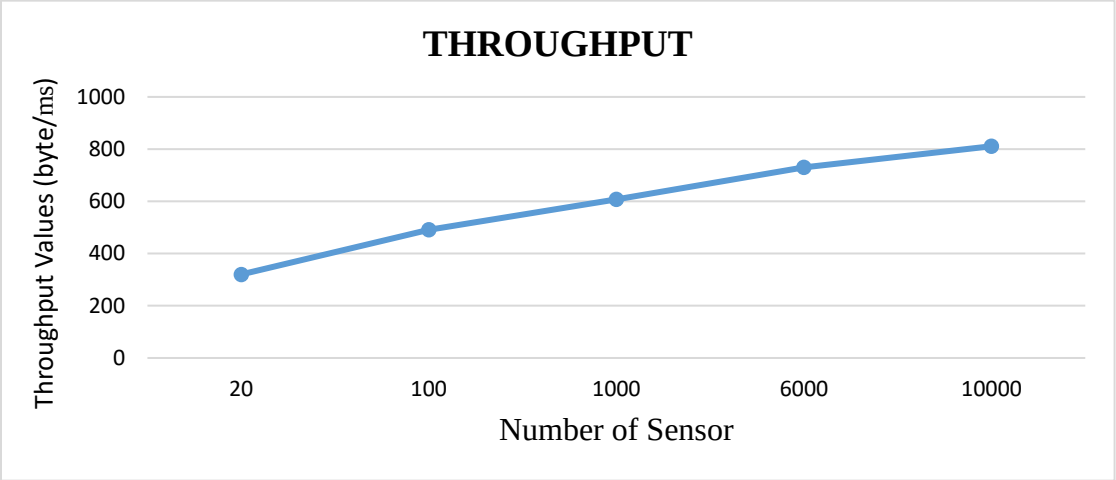

Figure 4.12: Throughput calculation of Zstandard algorithm.

4.3.3 The 3rd case Study Results

It explained as encryption firstly compression secondly within the proposed system, it provided encryption of incoming sensor reads with TEA encryption algorithm and the generated cipher is compressed with Zstandard to satisfy requirements of network constraints and transmit sensor data with less delay, high throughput, and decrease size of encryption data about 49 % as showed in table (4.6).

Table 4.6: Encryption, compression sizes, and compression ratio of 3rd case study.

No. of sensor	Size of Data in byte	Size after Encryption Data in byte	Size after Compression Data in byte	Compression Ratio in %
20	255	544	282	48%
100	1228	2480	1250	49%
1000	12175	24384	12202	50%
6000	72993	146016	73008	50%
10000	121632	243296	121660	50%

Figure (4.13) showed the increased number of sensors led to increased size of data which effect on the size of data also increased with

encryption data; and the compression process was used to decrease the size of encryption data.

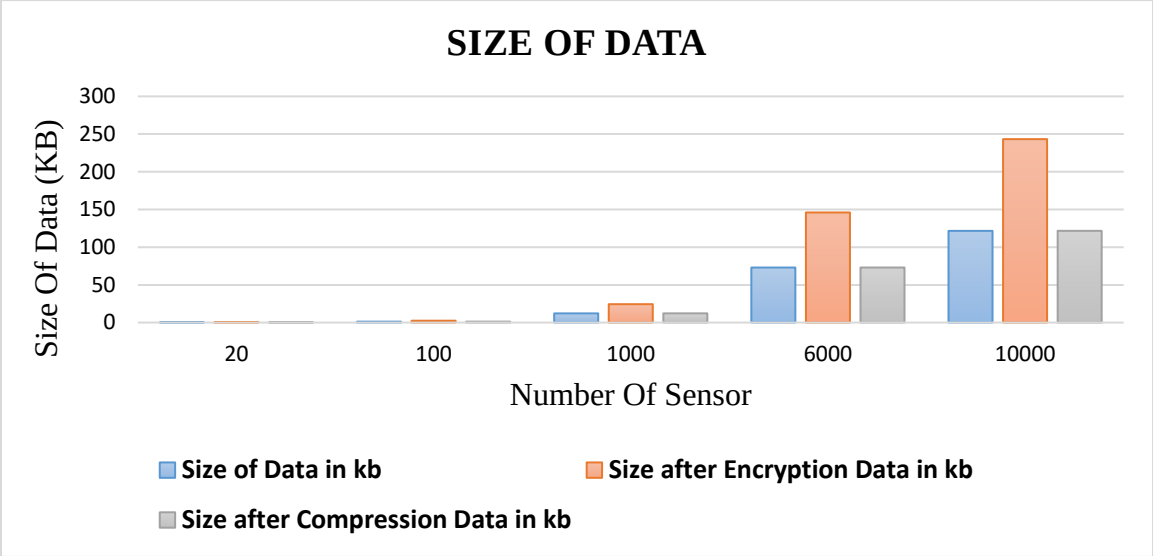

Figure 4.13: Data sizes of plain, encryption, and compression.

Figure (4.14) showed the compression process results in the same ratio due to the high randomness that is generated by the encryption algorithm.

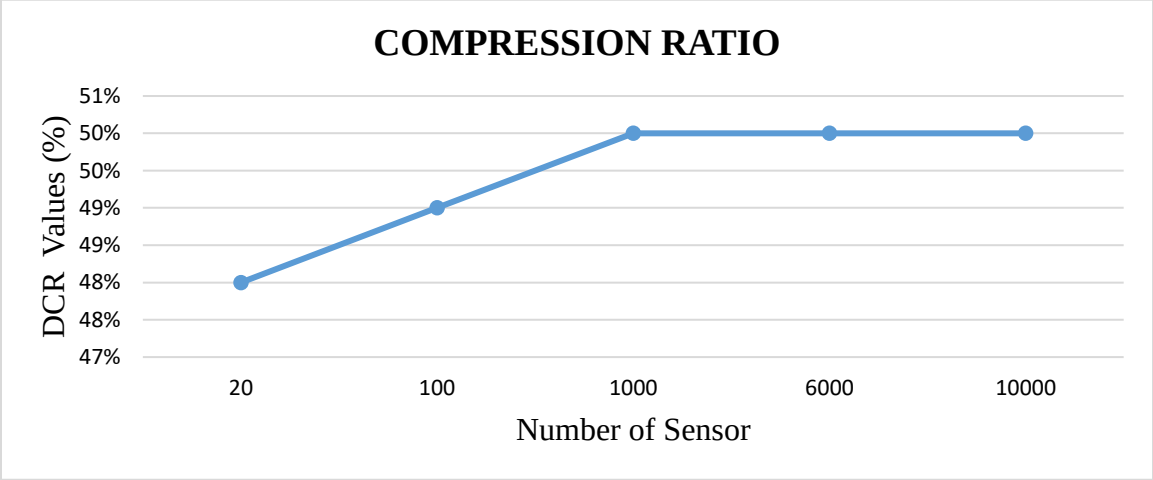

Figure 4.14: Compression Ratio of 3rd case.

Table (4.7) showed the used sensor number is (20 to 10000) with different sizes of generated reads as (255 to 121632) as the size of data

increased to (544 to 243296) respectively but the size of data decreased to (282 to 121660) when using compression algorithm. The encryption time was also calculated. In addition, the throughput value showed the amount of data per millisecond.

Table 4.7: The results of Encryption Data of 3rd case study.

No. of Sensor	Size of Data in byte	Size of Encryption in byte	Time Encrypt in ms	Throughput Encrypt in byte/ms
20	255	544	2	127.5
100	1228	2480	7	175.43
1000	12175	24384	60	202.92
6000	72993	146016	295	247.43
10000	121632	243296	455	267.32

Figure (4.15) showed time of encryption how it changed from sensor number to another number.

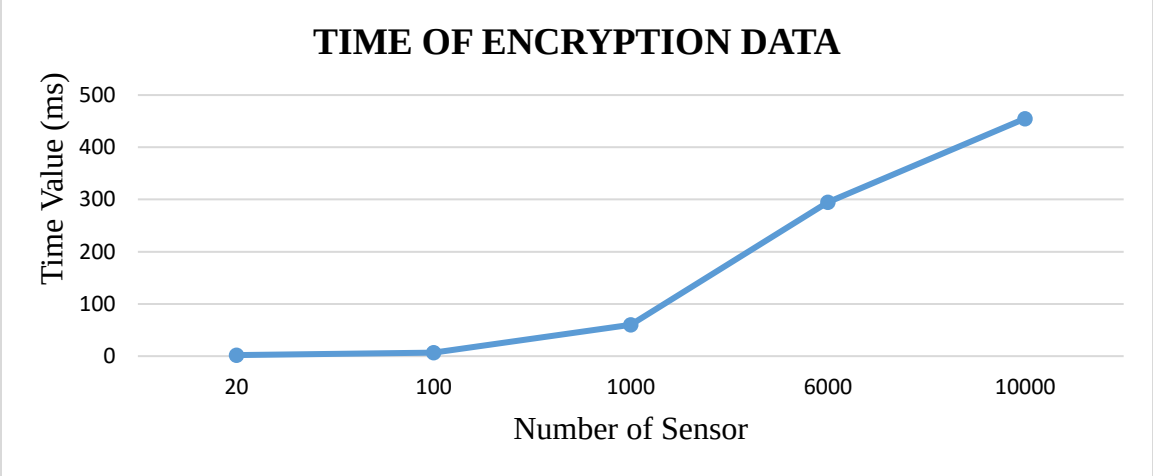

Figure 4.15: Time of Encryption Data.

Figure (4.16) showed the amount of throughput of encryption data with the used different sensor types.

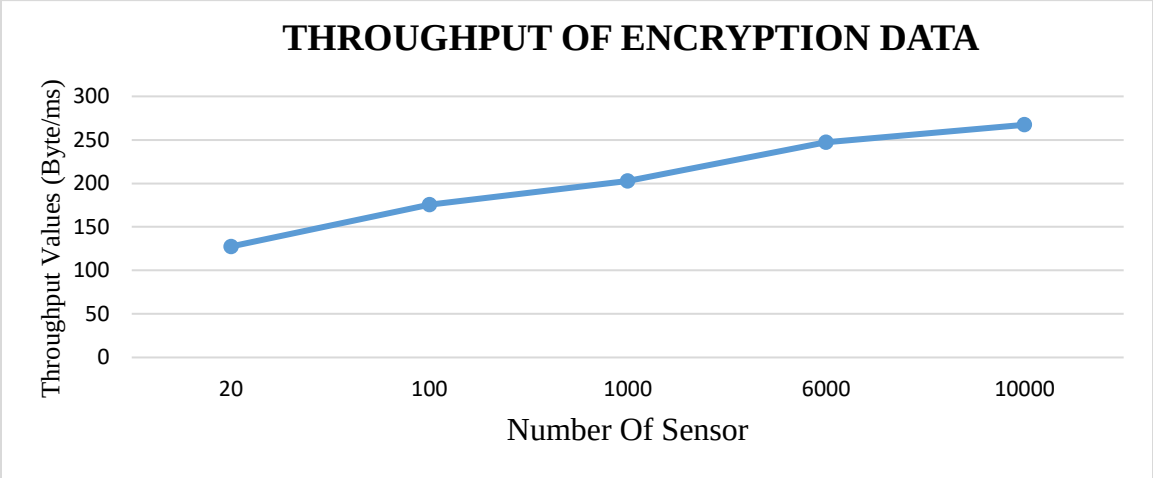

Figure 4.16: Throughput of Encryption Data.

Table (4.8) showed the time required to decryption and throughput value depending on the different sensor number. It explained with increased number of sensors, the required time is changed compared with encryption process.

Table 4.8: The results of Decryption Data of 3rd case study.

No. of Sensor	**Size of Data in byte**	**Time Decryption in ms**	**Throughput Decrypt in byte/ms**
20	255	2.3	110.87
100	1228	6	204.67
1000	12175	55	221.36
6000	72993	300	243.31
10000	121632	435	279.61

Figure (4.17) shows how long the system takes to decrypt data, which gets longer as the number of sensors and the size of the data goes up.

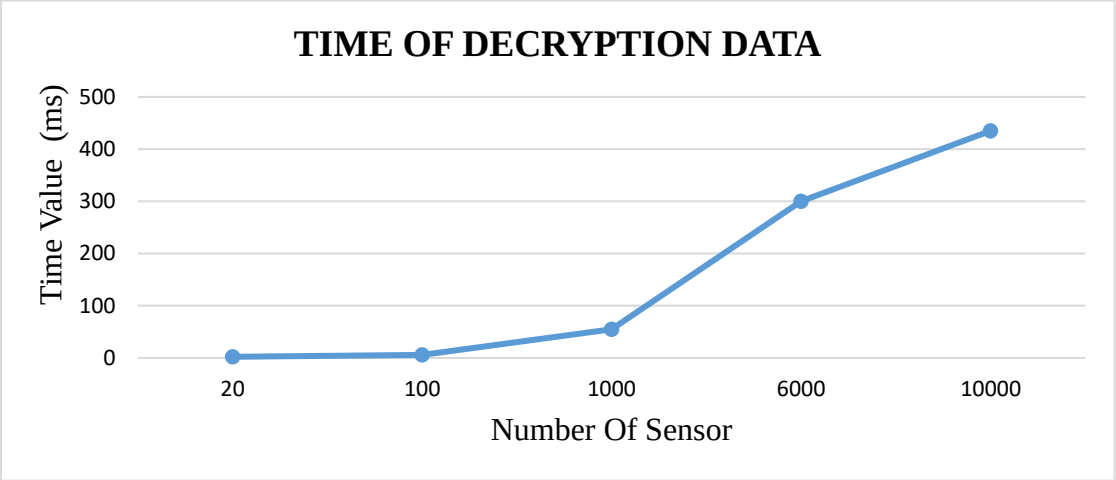

Figure 4.17: Time of Decryption Data.

Figure (4.18) showed the amount of throughput of encryption data with the used different sensor types

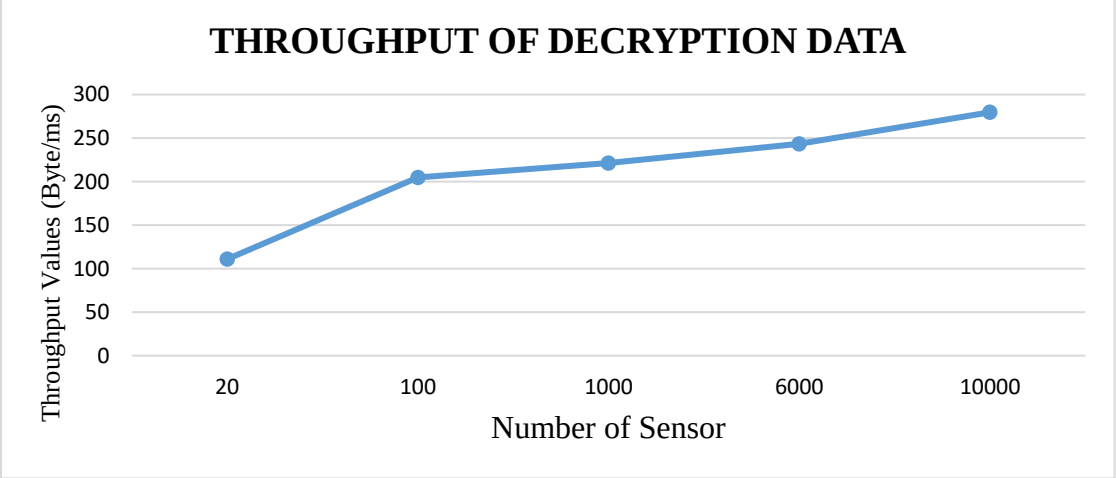

Figure 4.18: Throughput of Decryption Data of 3rd case study.

Table (4.9) showed the computation processes of the 3rd case as the delay is increased due to the increased number of sensors and size of input data, as well as RAM usage are changed depending on the process computation.

Table 4.9: Delay, and RAM through 3rd case study.

No. of Sensor	Delay in ms	RAM in Mbps
20	6	13.22
100	17.7	13.81
1000	140	14.44
6000	650	14.91
10000	1103	16.29

Figure (4.19) showed the value of delay (latency) in 3rd case study. The value is increased due to increased number of sensors and size of input data.

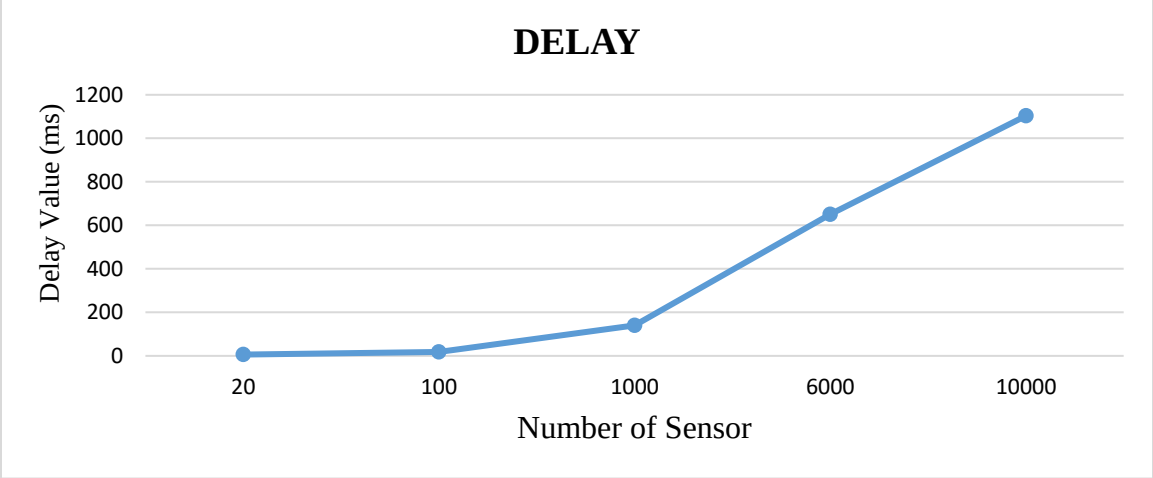

Figure 4.19: Delay Value of 3rd case study.

Figure (4.20) showed an effective result of the RAM usage. Notice a change in RAM values due to the available space and of software running in at the runtime of implementation of 3rd case study.

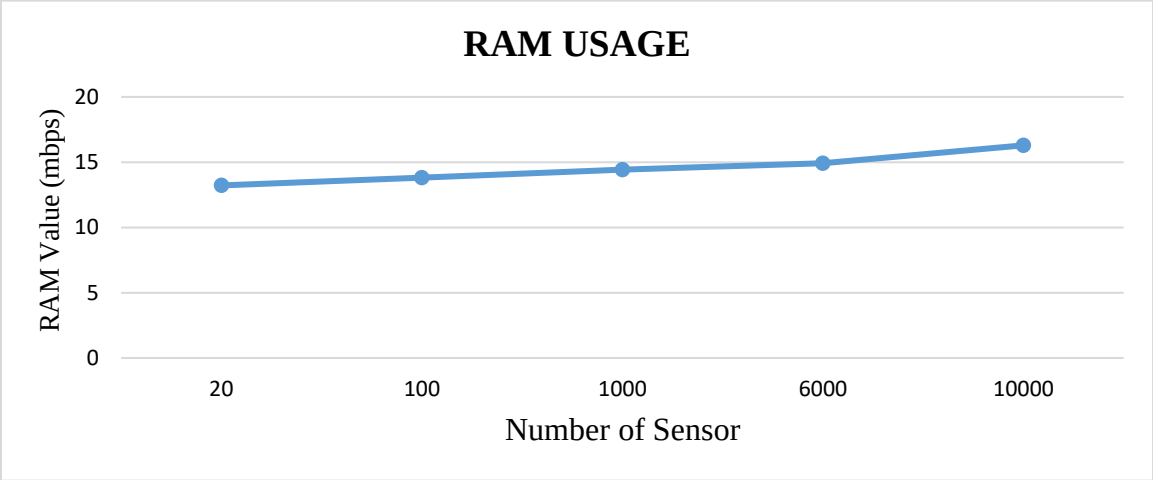

Figure 4.20: RAM usage of 3rd case study.

4.3.4 The 4th case study Results

This case is implemented as compression process firstly, and encryption process secondly. The Zstandard compressed incoming sensor reads and pass them into another process of TEA lightweight encryption, table (4.10) showed the better results for compression/encryption processes compared with 3rd case study. The compression ratio arrived to 81% in 10000 sensor data reads size 121632 Bytes.

Table 4.10: Encryption, compression sizes, and compression ratio Compcrypt system.

No. of Sensor	Size of Data in byte	Size after Compression Data (CD) in byte	Size after Encryption of CD in byte	Compression Ratio in %
20	255	114	256	55%
100	1228	455	736	62%
1000	12175	3587	6208	70%
6000	72993	15336	34704	78%
10000	121632	22730	57488	81%

Figure (4.21) showed the difference of sensor data generated depending on the sensor number for 4th case study.

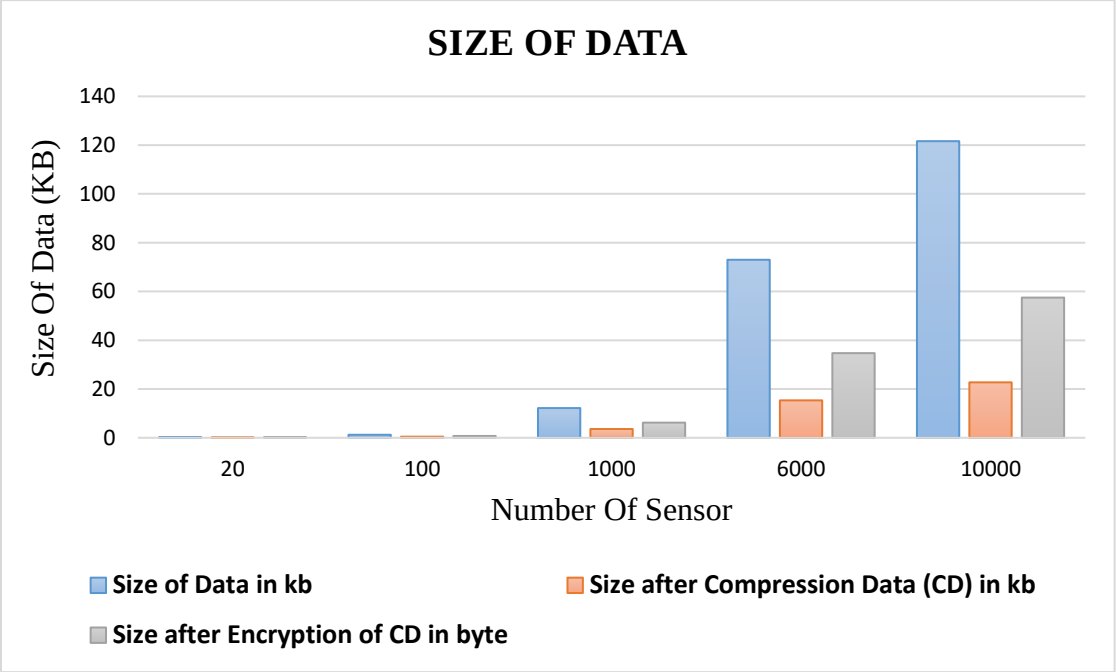

Figure 4.21: Size of Data for Different Sensor for Compcrypt System.

Figure (4.22) showed the enhancement process of Compcrypt system to be equivalent to the resource computation sensors due to huge amount of data generated from these sensors. The Compcrypt system is flexible model to provide a lightweight method for processing high data rate, it works in less delay and compatible to IoT sensor environment as it provided 81 % compression of encrypted data.

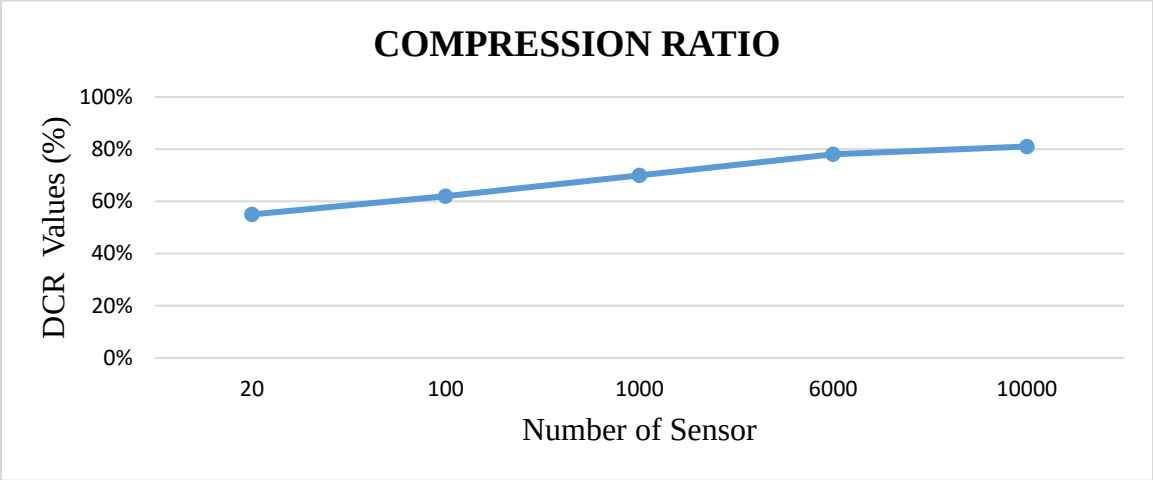

Figure 4.22: Compression Ratio of Compcrypt System.

Table (4.11) showed the better results of used sensor number is (20 to 10000) with different size of generated reads as (255 to 121632) as the size of encryption decreased into (256 to 57488) respectively. The encryption time is decreased into (0.7 of 20 sensors, and 109 of 10000).

Table 4.11: The results of Encryption Data using Compcrypt system.

No. of Sensor	**Size of Data in byte**	**Size of Encryption in byte**	**Time Encrypt in ms**	**Throughput Encrypt in byte/ms**
20	255	256	0.7	364.29
100	1228	736	1.8	682.22
1000	12175	6208	14.7	828.23
6000	72993	34704	72.7	1004.03
10000	121632	57488	108.5	1121.03

Figure (4.23) showed the time of encryption changed depending on the number of sensors and data sensor reads.

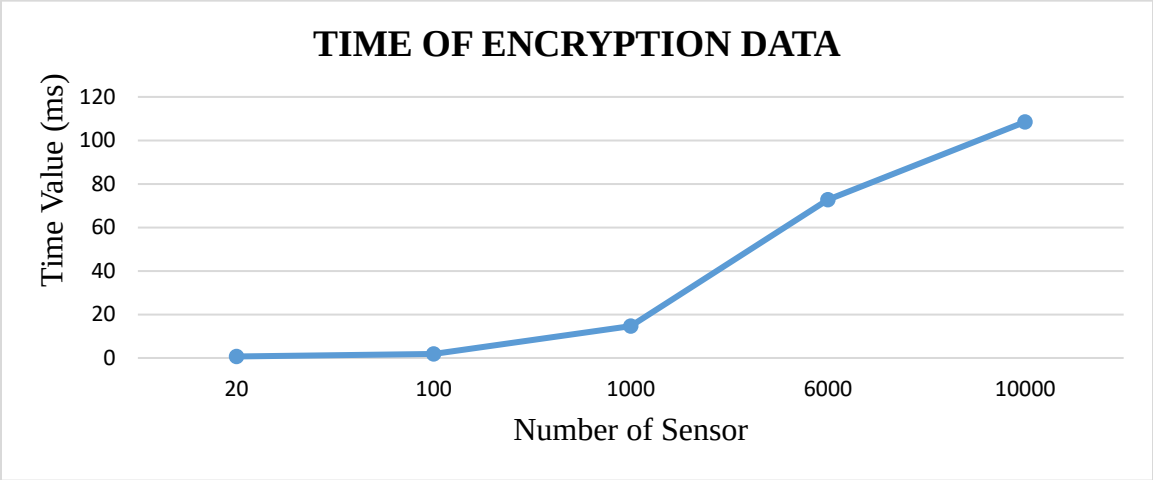

Figure 4.23: Encryption Time of Compcrypt System.

Figure (4.24) showed the throughput value changed depending on the number of sensors and data sensor reads.

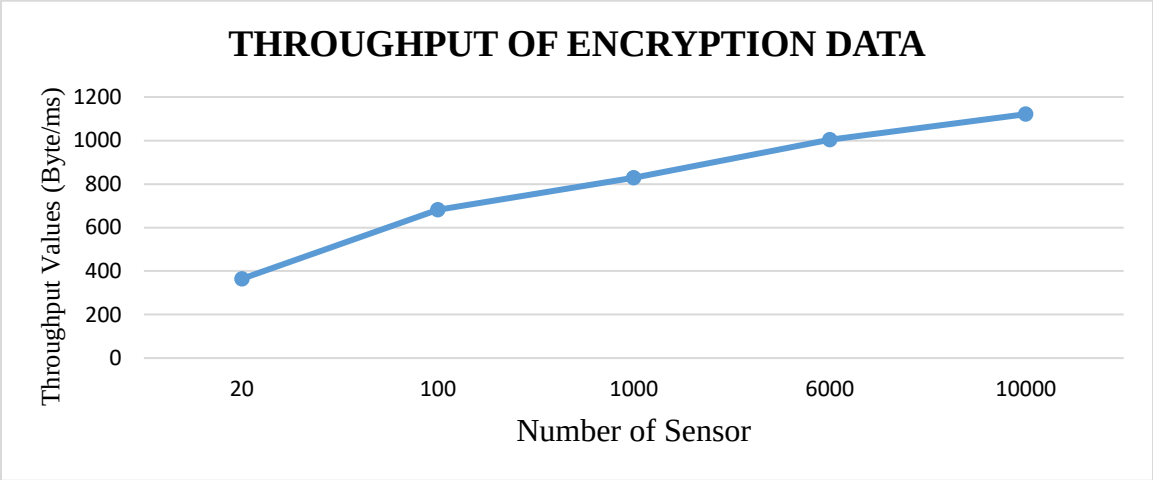

Figure 4.24: Throughput of Encryption Data of Compcrypt system.

Table (4.12) showed the time required to decryption and throughput value depending on the different sensor number of Compcrypt system. It explained with increased number of sensors, the required time is changed compared with encryption process as the better results is 105.5 ms and the better results of throughput decrypt is 1152.92 byte/ms of 10000 sensors.

Table 4.12: The results of Decryption Data using Compcrypt system.

No. of Sensor	Size of Data in byte	Time Decrypt in byte	Throughput Decrypt in byte/ms
20	255	0.8	318.75
100	1228	2.1	584.76
1000	12175	16.4	742.38
6000	72993	69.4	1051.77
10000	121632	105.5	1152.91

Figure (4.25) shows how long the system takes to decrypt data, which gets longer as the number of sensors and the size of the data goes up.

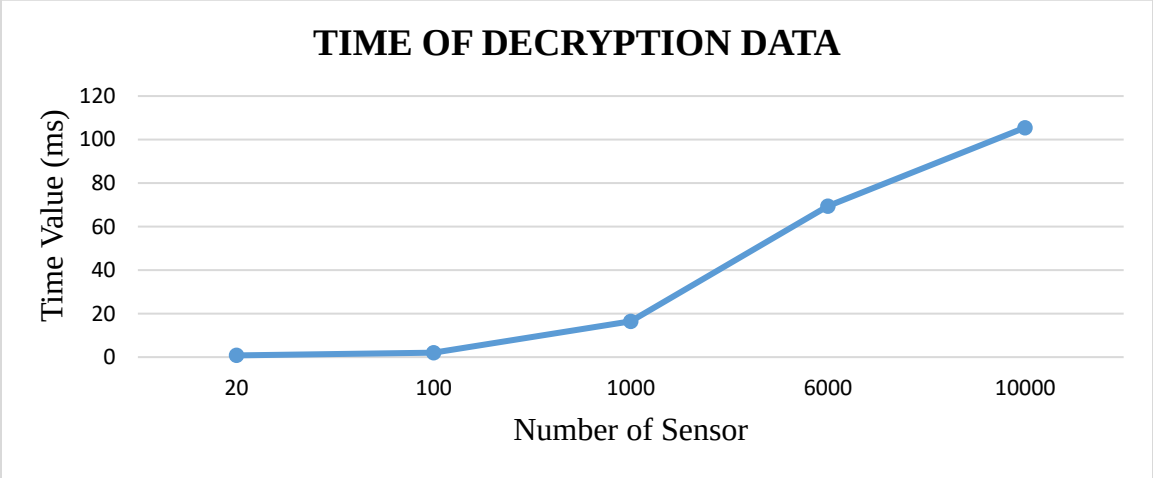

Figure 4.25: Decryption Time of Compcrypt system.

Figure (4.26) showed the throughput value changed depending on the number of sensors and data sensor reads.

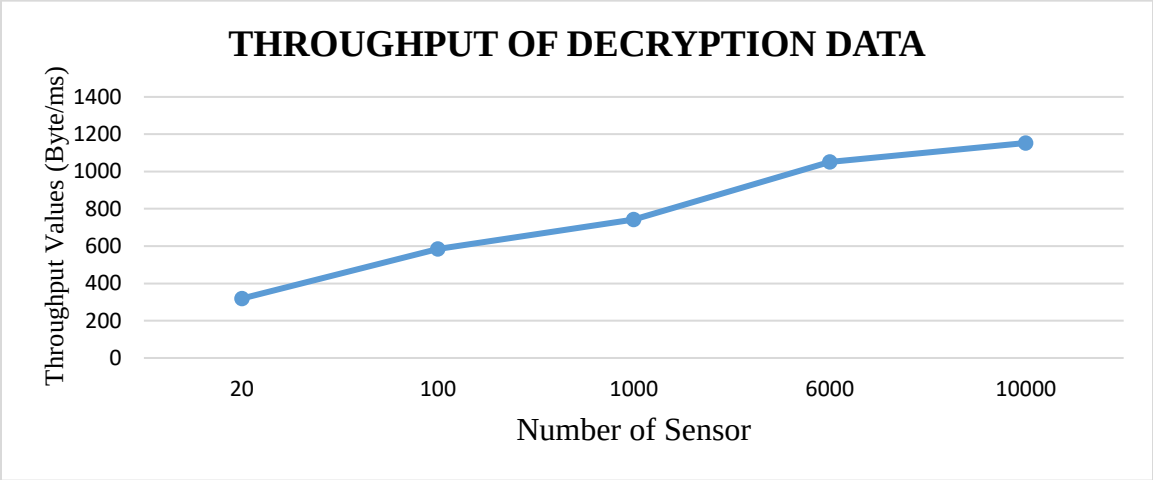

Figure 4.26: Throughput of Decryption Data of Compcrypt system.

Table (4.13) showed the computation processes of the proposed Compcrypt system as the delay and RAM usage which is better in compared with encryption process as the delay results is 246.28 ms and the RAM usage results is 14.46 Mbps of 10000 sensors.

Table 4.13: Delay, and RAM through Compcrypt System.

No. of sensor	**Delay in ms**	**RAM in Mbps**
20	3.99	13.18
100	6.82	14.21
1000	35.1	14.30
6000	124.8	14.42
10000	246.28	14.46

Figure (4.27) showed the value of delay (latency) in Compcryt system. The value is increased due to increased number of sensors and size of input data.

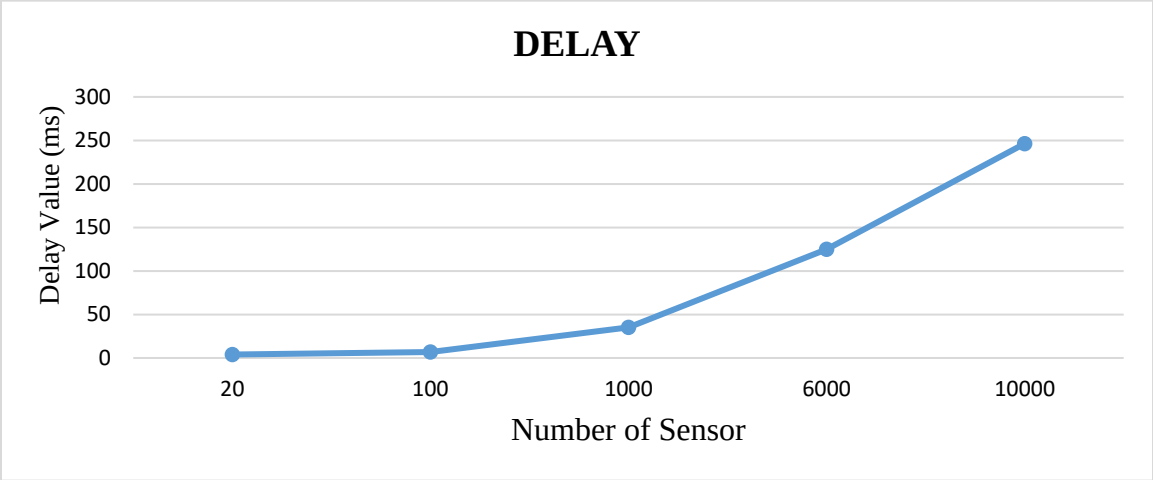

Figure 4.27: Delay Value of Compcrypt system.

Figure (4.28) showed an effective result of the RAM usage. Notice a change in RAM values due to the available space and of software running in at the runtime of implementation of Compcrypt system.

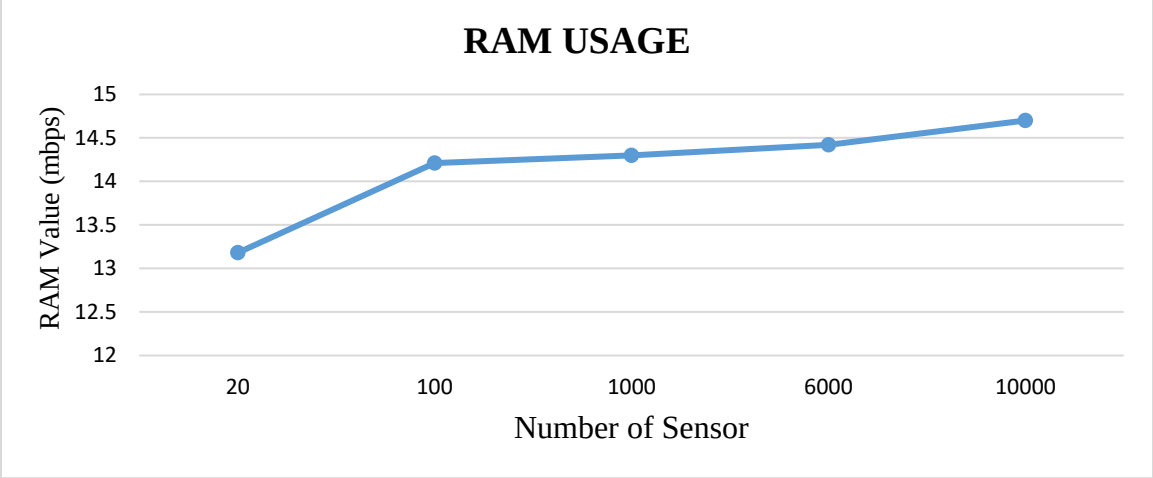

Figure 4.28: RAM usage of Compcrypt system.

4.4 Security Analysis

There are many security parameters used to evaluate the security strength concept for instance, Avalanche effect, and Entropy. The used parameters implemented and tested for cases 1st, 3rd, and 4th.

4.4.1 Entropy (randomness)

It is an important parameter for security analysis that measures the randomness of data for plain text and cipher text. The effect of entropy can be observed by the equation that has been explained in chapter (2) section (2.4.6).

Table (4.14), and figure (4.29) showed the secure randomness evaluator (entropy) value of generated as the original data has minimum randomness and the randomness value is highly changed after encryption process, which is excellent and efficient values of the entropy.

Table 4.14: Entropy value in 1st, 3rd, 4th case study.

No. of sensor	Entropy of Plain Text	Entropy of Cipher Text of 1st case (TEA)	Entropy of Cipher Text of 4th case (compcrypt)
20	2.96553063	7.1859800	6.5135573
100	3.029158646	7.8302923	7.4459457
1000	3.049859535	7.9840099	7.9319280
6000	3.049670111	7.9973801	7.9908201
10000	3.050035171	7.9984134	7.9987151

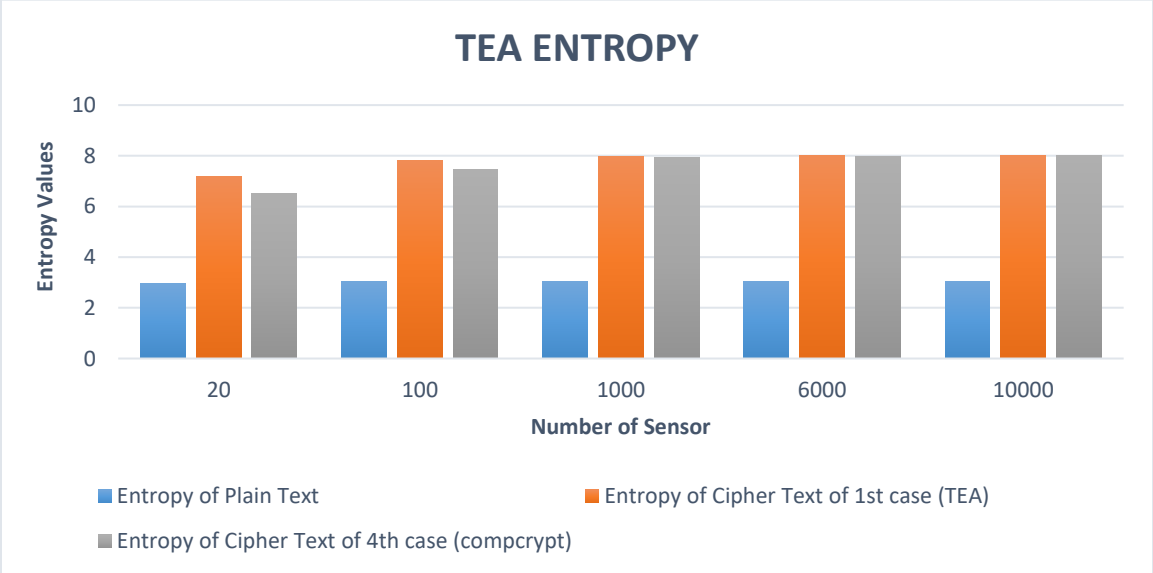

Figure 4.29: Entropy value in 1st, 4th case study.

4.4.2 Avalanche Effect (AE)

It is viewed as an important cryptographic security analysis that gauges the goodness of a technique of encryption. The effect of the avalanche can be observed by equation that been explained in chapter (2) section (2.6.6).

Table (4.15), and table (4.16) explained the AE values based on the proposed sensor reads of (34, 100, 97) for TEA lightweight encryption algorithm, and the proposed Compcrypt system. The proposed method based on the convert the used text to binary and then match the change one bit in binary plaintext and it matched the two states from the cipher and it found in general that the proposed Compcrypt there is a few repeated letters from the used cases.

Table 4.15: AE steps for TEA algorithm.

The proposed plaintext = (34, 100, 97)
Plaintext in binary = 110011 110100 101100 100000 110001 110000 110000 101100 100000 111001 110111
CipherText without change bits= 160 bits 36**c**08514**b**493b61dd053ed84439cd1d6e89f10b3afa7d2c7**4**b7c88**f**e**3**f5e607996**0**89ecd 64bdd800b84e6ffaa**4**a6c451778f1df**41**cf4631ab2be90a7e**1**4f3f7f4ade**d**9**3d**b**8**dc**d**e6a20 f77563127610be
CipherText with change bits= 160 bits 12**c**db1e1**b**054e1d522490bbe1d2a8bc95a1488662644769f**4**3f7bb**f**7**3**af1458bc0**0**4aa72 5c9b369a8e58c94df**4**334b751946ff9**41**f5a19718dc537b08**1**93612166e2**d**5**3d**9**8**98**d**62 10e0cb700969e0e54
AE=160-15=145 / 160 = **0.906**

Table 4.16: AE steps Compcrypt system.

The proposed plaintext = (34, 100, 97)
Plaintext in binary = 110011 110100 101100 100000 110001 110000 110000 101100 100000 111001 110111
CipherText without change bits= 128 bits b**2**fe928ac6cd7ac552efd4d74d7b6099**2**e62**8**26c9a7403b4fb1302**8**56b0ac8b8ed**9**be0184e904**a**e1c05d0fafb0**2**cad8bc8e93fab3504336682b3b475a77e3b4b
CipherText with change bits= 128 bits 8**2**85f477143e94fc4cbdbd4a119e2be6**2**a9e**8**ae0de62ce1128564a**8**67abb196aa7**9**f5fc3ef816**a**2b8e28b661f6**2**3726c0c865c4a872aebd49aa0059d2dd561ef
AE=128-7= 121 / 128 = **0.945**

4.5 Discussion of the Results

The main discussion of this work and the results are summarized in the following points:

i. Size of data: the size of encryption data in the 1st case (using TEA lightweight encryption algorithm) and the 3rd case (encryption firstly compression secondly) increased compared to the 4th case (Compcrypt system (compression firstly encryption secondly)). This indicates that the proposed Compcrypt system is saving network bandwidth and consuming fewer resources.

ii. Compression Ratio: the 4th case (Compcrypt system (compression firstly encryption secondly)) has highest compression ratio as compared to the 3rd case (encryption firstly compression secondly). This indicates

that the size of transfer data in the proposed Compcrypt system is small then other case.

iii. Encryption Time: the two cases (1st and 3rd) take time more in the encryption process. While, the Compcrypt system (4th case) take less time for encryption. This paves the way to implement the proposed Compcrypt system.

iv. In terms of the throughput: the Compcrypt system have highest throughput as compared to those of the 1st and 3rd case. This indicates that the proposed Compcrypt is faster (speed) than another case.

v. Entropy: is a high score in all the case which shows that the randomness is high. This performance value is preferable in any cryptosystem to make the difficulty to be guessed by an attacker.

vi. Avalanche Effect: the proposed Compcrypt system has the highest avalanche effect score and hence is used in the application has confidentiality and integrity are the highest priory.

Figure (4.30) showed the main comparison of different cases and explained the proposed Compcrypt system is better results in security and network metrics.

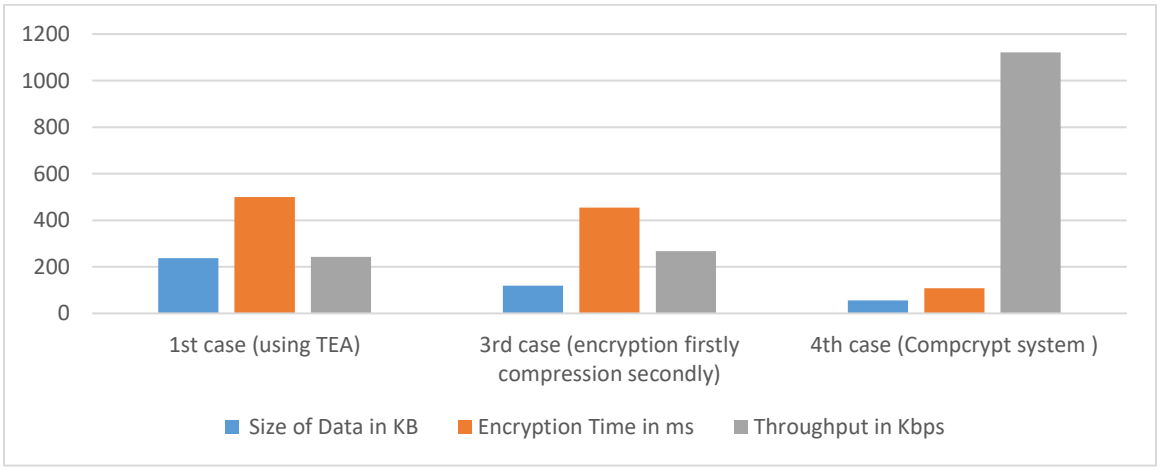

Figure 4.30: The main comparison of 1st, 3rd, and 4th cases.

Table (4.17) shows the security goals achieved through the use of the proposed Compcrypt system.

Table 4.17: The Security Goals.

Security Goals	**Yes/ No**	**The Achievement Function**
Conditionality	✓	Compcrypt system
Integrity	✓	TEA encryption
Availability	✓	IoT sensor + Raspberry Pi + server
Authentication	✓	Elliptic curve Diffie Hellman protocol

4.6 System Comparison

The proposed system was compared with different related work based on different evaluation parameters. Table (4.18) showed the comparison of the proposed Compcrypt system with other related work for the encryption and decryption process. It is observed that the proposed system provides better results in time and throughput as the data size is (118.87 KB), time encryption is (108 ms), time decryption is (105.5 ms), and throughput is (1121 Kbps).

Table 4.18: A comparison the Compcrypt system with other related works.

Ref. No Year	**Algorithm**	**Data Size in KB**	**Encryption Process**		**Decryption Process**	
			Time in ms	**Throughput in Kbps**	**Time in ms**	**Throughput in Kbps**
[23] 2020	Lightweight PGP	3.68	235	14.54	NA	NA
[21] 2020	RC4, ECC, and SHA-256	20	75	273	75	273
[24] 2021	LAES	0.068	2.3	0.03	NA	NA

[25] 2022	LWC-AES	45.1	280	161.07	291	154.98
The proposed Compcrypt system	**Zstandard + TEA + ECDH**	**118.78**	**108**	**1121**	**105.5**	**1152.91**

Table (4.19) showed the comparison of the proposed Compcrypt system with other related work for the Compression Ratio (CR), Entropy, Avalanche Effect (AE). It is observed that the proposed system provides better results in CR, Entropy, and AE as the data size is (118.87 KB), CR is (81%), Entropy is (7.999), and AE is (0.94).

Table 4.19: A comparison the Compcrypt system CR, Entropy, and AE.

Ref. No Year	**Algorithm**	**Data Size in byte**	**CR**	**Entropy**	**AE**
[19] 2019	AES+ Huffman	218.895	34%	7.998	50%
The proposed Compcrypt system	**Zstandard + TEA + ECDH**	**118.78**	**81%**	**7.999**	**94%**

The proposed system was compared to different related works based on the compression of IoT sensing data. The results indicate that the proposed system provides better results in compressed data size and ratio value, with the compression data size being 881 bytes and the compressed ratio being 70.6 bytes. Table (4.20) shows the comparison with other related

works. Considering the work in [94], they made use of the Huffman based LZW algorithm with an original data size of 3000 bytes and compression data size of 1072 bytes, having a compression ratio of 64%.

Table 4.20: A comparison the Compcrypt system for Compression process.

Ref. No	[94] 2021	**The Proposed System**
Programming language	Python	Python
Algorithm	Huffman based LZW	**Zstandard**
Original Data size in Byte	3000	**3000**
Data size after Compression process	1072	**881**
Ratio (CR)	64%	**70%**

4.7 Summary

This chapter demonstrates the implementation of the proposed Compcrypt system in an IoT healthcare application. There are two sensors used namely the temperature sensor, and SPo_2 sensor for the number of suspected patients, management device (Raspberry pi), and server for data evaluation. Firstly, the sensing data is generated from the sensor and collected in the Raspberry Pi. Secondly, the collected data is compressed by the Zstandard algorithm to minimize the size of the data transfer. Thirdly, the compressed data is encrypted by using the TEA lightweight encryption algorithm with the EllipticCurveDiffieHellman key exchange protocol and then sent to the server. In addition, this chapter explained the evaluation data after the decryption and decompression processes. The obtained results of the proposed Compcrypt system are a good value in terms of the performance evaluation parameters such as size of data, encryption time, throughput, RAM usage, compression ratio, entropy, and avalanche effect.

Chapter Five

Conclusions and Suggestions for Future Works

CHAPTER FIVE

CONCLUSION AND SUGGESTIONS FOR FUTURE WORK

5.1 Conclusions

This thesis provided an efficient way to minimized the volume of streaming IoT sensor data, increase the speed of data transfer, save the IoT network bandwidth, and protect sensing data by combine data compression and lightweight cryptographic techniques. The following points are concluded from the work of this thesis:

1. Robust keys generation method is based on the principles the Elliptic Curve Diffie Hellman protocol.
2. The size of encryption data in the case using TEA lightweight encryption algorithm and the case using encryption firstly compression secondly increased compared to the case using the proposed system (compression firstly encryption secondly). This indicates that the proposed Compcrypt system is saving network bandwidth and consuming fewer resources.
3. The case using proposed system (compression firstly encryption secondly)) has highest compression ratio as compared to the case using encryption firstly compression secondly. This indicates that the size of transfer data in the proposed Compcrypt system is small then other case.
4. The proposed system takes less time for encryption compared with TEA encryption algorithm. This paves the way to implement the proposed Compcrypt system.
5. the proposed system has highest throughput as compared with TEA encryption algorithm. This indicates that the proposed Compcrypt is faster (speed) than another case.

6. Entropy is a high score in all the proposed system which shows that the randomness is high. This performance value is preferable in any cryptosystem to make the difficulty to be guessed by an attacker.
7. The proposed system has the highest avalanche effect score and hence is used in the application has confidentiality and integrity are the highest priory.

5.2 Future Works

This section describes plans and suggestions for future work that might follow the work described in this thesis.

1. Using a new data mining technique to reduce the volume of sensing data that is sent to decision makers.
2. Evaluation of the proposed Compcrypt system with different attackers such as Man in Middle, Guessing, Brute Force, Sybil, and Flooding Attacks.
3. Evaluating the proposed Compcrypt system in Secure Distributed Framework for Smart Mobile IoT environments.
4. Implementing the proposed Compcrypt system with take into account the energy evaluation parameters of IoT sensors.
5. Using a hybrid lightweight encryption technique to increase the strength of security.
6. Implementing the proposed Compcrypt system with 5G environment for IoT healthcare application.

REFERENCES

[1] M. P. Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, "Internet of Things (IoT) A Vision, Architectural Elements, and Future Directions," *Am. Soc. Mech. Eng. Heat Transf. Div. HTD*, vol. 369, no. 6, pp. 321–332, 2013, doi: 10.1115/imece2001/htd-24365.

[2] S. Hammoudi, Z. Aliouat, and S. Harous, "Challenges and research directions for Internet of Things," *Telecommun. Syst.*, vol. 67, no. 2, pp. 367–385, 2018, doi: 10.1007/s11235-017-0343-y.

[3] A. M. Rahmani *et al.*, "Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach," *Futur. Gener. Comput. Syst.*, vol. 78, no. February, pp. 641–658, 2018, doi: 10.1016/j.future.2017.02.014.

[4] K. Rose, S. Eldridge, and L. Chapin, "The Internet of Things (IoT): An Overview," *Int. J. Eng. Res. Appl.*, vol. 5, no. 12, pp. 71–82, 2015, [Online]. Available: https://crsreports.congress.gov.

[5] K. Padmavathi, C. Deepa, and P. Prabhakaran, "Internet of Things (IoT) and Big Data," *Internet Things Big Data Anal.*, no. April, pp. 217–246, 2020, doi: 10.1201/9781003036739-10.

[6] B. Vijayalakshmi and N. Sasirekha, "Comparative Analysis of Lossless Text Compression Methods with Novel Tamil Compression Technique," vol. 9, no. 7, pp. 38–44, 2021.

[7] S. K. Routray, A. Javali, A. Sahoo, W. Semunigus, and M. Pappa, "Lossless compression techniques for low bandwidth io ts," *Proc. 4th Int. Conf. IoT Soc. Mobile, Anal. Cloud, ISMAC 2020*, pp. 177–181, 2020, doi: 10.1109/I-SMAC49090.2020.9243457.

[8] S. Anand and S. K. Routray, "Issues and challenges in healthcare narrowband IoT," *Proc. Int. Conf. Inven. Commun. Comput. Technol. ICICCT 2017*, no. March, pp. 486–489, 2017, doi: 10.1109/ICICCT.2017.7975247.

[9] S. Chandra, S. Paira, S. S. Alam, and G. Sanyal, "A comparative survey of symmetric and asymmetric key cryptography," *2014 Int. Conf. Electron. Commun. Comput. Eng. ICECCE 2014*, pp. 83–93, 2014, doi: 10.1109/ICECCE.2014.7086640.

[10] S. S. Dhanda, B. Singh, and P. Jindal, *Lightweight Cryptography: A Solution to Secure IoT*, vol. 112, no. 3. Springer US, 2020, doi: 10.1007/s11277-020-07134-3.

[11] M. Rana, Q. Mamun, and R. Islam, "Current Lightweight Cryptography Protocols in Smart City IoT Networks: A Survey," pp. 1–22, 2020, [Online]. Available: http://arxiv.org/abs/2010.00852.

[12] S. Di Matteo, L. Baldanzi, L. Crocetti, P. Nannipieri, L. Fanucci, and S. Saponara, "Secure elliptic curve crypto-processor for real-time iot applications," *Energies*, vol. 14, no. 15,

2021, doi: 10.3390/en14154676.

[13] P. S. Mukesh, M. S. Pandya, and S. Pathak, “Enhancing AES algorithm with arithmetic coding,” *Proc. 2013 Int. Conf. Green Comput. Commun. Conserv. Energy, ICGCE 2013*, pp. 83–86, 2013, doi: 10.1109/ICGCE.2013.6823404.

[14] S. Malekzadeh, “A Fast Combination of AES Encryption and LZ4 Compression Algorithms,” pp. 1–6, 2018, doi: 10.13140/RG.2.2.33644.56960.

[15] A. J. Abboud, A. N. Albu-Rghaif, and A. K. Jassim, “Balancing compression and encryption of satellite imagery,” *Int. J. Electr. Comput. Eng.*, vol. 8, no. 5, pp. 3568–3586, 2018, doi: 10.11591/ijece.v8i5.pp3568-3586.

[16] H. Noura, R. Couturier, C. Pham, and A. Chehab, “Lightweight Stream Cipher Scheme for Resource-Constrained IoT Devices,” *Int. Conf. Wirel. Mob. Comput. Netw. Commun.*, vol. 2019-Octob, 2019, doi: 10.1109/WiMOB.2019.8923144.

[17] Z. M. J. Kubba and H. K. Hoomod, “A Hybrid Modified Lightweight Algorithm Combined of Two Cryptography Algorithms PRESENT and Salsa20 Using Chaotic System,” *1st Int. Sci. Conf. Comput. Appl. Sci. CAS 2019*, pp. 199–203, 2019, doi: 10.1109/CAS47993.2019.9075488.

[18] V. Prakash, A. V. Singh, and S. Kumar Khatri, “A New Model of Light Weight Hybrid Cryptography for Internet of Things,” *Proc. 3rd Int. Conf. Electron. Commun. Aerosp. Technol. ICECA 2019*, pp. 282–285, 2019, doi: 10.1109/ICECA.2019.8821924.

[19] M. R. Ashila, N. Atikah, D. R. Ignatius Moses Setiadi, E. H. Rachmawanto, and C. A. Sari, “Hybrid AES-Huffman Coding for Secure Lossless Transmission,” *Proc. 2019 4th Int. Conf. Informatics Comput. ICIC 2019*, no. October, 2019, doi: 10.1109/ICIC47613.2019.8985899.

[20] R. Chatterjee and R. Chakraborty, “A Modified Lightweight PRESENT Cipher for IoT Security,” 2020, doi: 10.1109/ICCSEA49143.2020.9132950.

[21] S. K. Mousavi, A. Ghaffari, S. Besharat, and H. Afshari, “Improving the security of internet of things using cryptographic algorithms: a case of smart irrigation systems,” *J. Ambient Intell. Humaniz. Comput.*, vol. 12, no. 2, pp. 2033–2051, 2021, doi: 10.1007/s12652-020-02303-5.

[22] P. Bhargavi and K. Srilakshmi, “Efficient dual compression and encryption technique for enhanced cloud security,” *J. Green Eng.*, vol. 10, no. 10, pp. 8721–8735, 2020.

[23] W. Ullah Khan, T. Ullah, H. Khan Jadoon, and N. Ul Arfeen, “Lightweight Pretty Good Privacy Email Encryption,” *Int. Res. J. Mod. Eng. Technol. Sci. www.irjmets.com @International Res. J. Mod. Eng.*, no. 12, pp. 2582–5208, 1040, [Online]. Available: www.irjmets.com.

[24] M. S. Fadhil, A. K. Farhan, and M. N. Fadhil, "A lightweight aes algorithm implementation for secure iot environment," *Iraqi J. Sci.*, vol. 62, no. 8, pp. 2759–2770, 2021, doi: 10.24996/ijs.2021.62.8.29.

[25] H. M. Mohammad, A. A. Abdullah, and A. Info, "Enhancement Process of AES : A Lightweight Cryptography Algorithm-AES For Constrained Devices," *TELKOMNIKA Telecommun. Comput. Electron. Control Vol.*, vol. 20, no. 3, pp. 2–11, 2021, doi: 10.12928/TELKOMNIKA.v20i3.23297.

[26] M. Marjani *et al.*, "Big IoT Data Analytics: Architecture, Opportunities, and Open Research Challenges," *IEEE Access*, vol. 5, pp. 5247–5261, 2017, doi: 10.1109/ACCESS.2017.2689040.

[27] M. M. Islam, A. Rahaman, and M. R. Islam, "Development of Smart Healthcare Monitoring System in IoT Environment," *SN Comput. Sci.*, vol. 1, no. 3, pp. 1–11, 2020, doi: 10.1007/s42979-020-00195-y.

[28] I. K. Dutta, B. Ghosh, and M. Bayoumi, "Lightweight cryptography for internet of insecure things: A survey," *2019 IEEE 9th Annu. Comput. Commun. Work. Conf. CCWC 2019*, no. January, pp. 475–481, 2019, doi: 10.1109/CCWC.2019.8666557.

[29] I. Kalyan Dutta, B. Ghosh, A. H. Carlson, and M. Bayoumi, "Lightweight Polymorphic Encryption for the Data Associated with Constrained Internet of Things Devices," *IEEE World Forum Internet Things, WF-IoT 2020 - Symp. Proc.*, no. June, 2020, doi: 10.1109/WF-IoT48130.2020.9221296.

[30] M. U. B. Shabbir Hassan, "Key Exchange Algorithm for Lightweight Cryptographic Primitive," *J. Univ. Shanghai Sci. Technol.*, vol. 22, no. 10, pp. 1–16, 2020.

[31] Y. Perwej, M. A. Aboughaly, H. Ali, and M. Harb, "An Extended Review on Internet of Things (IoT) and its," vol. 7, no. 26, 2019.

[32] L. Atzori, A. Iera, and G. Morabito, "Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm," *Ad Hoc Networks*, vol. 56, pp. 122–140, 2017, doi: 10.1016/j.adhoc.2016.12.004.

[33] A. Singh, A. Payal, and S. Bharti, "A walkthrough of the emerging IoT paradigm: Visualizing inside functionalities, key features, and open issues," *J. Netw. Comput. Appl.*, vol. 143, no. June, pp. 111–151, 2019, doi: 10.1016/j.jnca.2019.06.013.

[34] D. G. Darwish and E. Square, "Improved Layered Architecture for Internet of Things," *Int. J. Comput. Acad. Res.*, vol. 4, no. 4, pp. 214–223, 2015, [Online]. Available: http://www.meacse.org/ijcar.

[35] M. Kalmeshwar and A. P. D. N. P. K S, "Internet Of Things: Architecture,Issues and Applications," *Int. J. Eng. Res. Appl.*, vol. 07, no. 06, pp. 85–88, 2017, doi: 10.9790/9622-

0706048588.

[36] H. Mrabet, S. Belguith, A. Alhomoud, and A. Jemai, “A survey of IoT security based on a layered architecture of sensing and data analysis,” *Sensors (Switzerland)*, vol. 20, no. 13, pp. 1–20, 2020, doi: 10.3390/s20133625.

[37] X. Liu *et al.*, “Overview of Spintronic Sensors with Internet of Things for Smart Living,” *IEEE Trans. Magn.*, vol. 55, no. 11, 2019, doi: 10.1109/TMAG.2019.2927457.

[38] S. Sicari, C. Cappiello, F. De Pellegrini, D. Miorandi, and A. Coen-Porisini, “A security- and quality-aware system architecture for Internet of Things,” *Inf. Syst. Front.*, vol. 18, no. 4, pp. 665–677, 2016, doi: 10.1007/s10796-014-9538-x.

[39] S. Kumar, P. Tiwari, and M. Zymbler, “Internet of Things is a revolutionary approach for future technology enhancement: a review,” *J. Big Data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0268-2.

[40] K. R. Ismagilova, Elvira; Hughes, L.; Dwivedi, Y.K.; Raman and Ismagilova, “Smart cities Advances in research An information systems perspective 2019.pdf,” no. 2019, pp. 88–100, 2022.

[41] S. G. H. Soumyalatha, “Study of IoT: Understanding IoT Architecture,Applications, Issues and Challenges,” *Int. J. Adv. Netw. Appl.*, pp. 1–5, 2019.

[42] J. M. Talavera *et al.*, “Review of IoT applications in agro-industrial and environmental fields,” *Comput. Electron. Agric.*, vol. 142, no. 118, pp. 283–297, 2017, doi: 10.1016/j.compag.2017.09.015.

[43] N. Patil and B. Iyer, “Health monitoring and tracking system for soldiers using Internet of Things(IoT),” *Proceeding - IEEE Int. Conf. Comput. Commun. Autom. ICCCA 2017*, vol. 2017-Janua, pp. 1347–1352, 2017, doi: 10.1109/CCAA.2017.8230007.

[44] Y. Chen, “Challenges & Oppoetunities in IoT,” *IEEE Conf. Wirel. Sensors*, vol. 16, no. 12, pp. 383–388, 2012.

[45] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu, “Security of the Internet of Things: perspectives and challenges,” *Wirel. Networks*, vol. 20, no. 8, pp. 2481–2501, 2014, doi: 10.1007/s11276-014-0761-7.

[46] D. Bandyopadhyay and J. Sen, “Internet of things: Applications and challenges in technology and standardization,” *Wirel. Pers. Commun.*, vol. 58, no. 1, pp. 49–69, 2011, doi: 10.1007/s11277-011-0288-5.

[47] M. G. Kallitsis, G. Michailidis, and M. Devetsikiotis, “Securing the Internet of Things Challenges, Threats and Solutions,” *Perform. Res.*, no. December, p. #pagerange#, 2018, [Online]. Available: https://doi.org/10.1016/j.pharmthera.2018.05.013.

[48] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, *Vision and Challenges for Realising the Internet of Things The meaning of things lies not in the things themselves, but in our attitude towards them. Antoine de Saint-Exupéry*, no. March. 2010.

[49] T. Guarda *et al.*, "Internet of Things challenges," *Iber. Conf. Inf. Syst. Technol. Cist.*, pp. 7–10, 2017, doi: 10.23919/CISTI.2017.7975936.

[50] M. Ge, H. Bangui, and B. Buhnova, "Big Data for Internet of Things: A Survey," *Futur. Gener. Comput. Syst.*, vol. 87, pp. 601–614, 2018, doi: 10.1016/j.future.2018.04.053.

[51] J. Azar, A. Makhoul, M. Barhamgi, and R. Couturier, "An energy efficient IoT data compression approach for edge machine learning," *Futur. Gener. Comput. Syst.*, vol. 96, pp. 168–175, 2019, doi: 10.1016/j.future.2019.02.005.

[52] S. Chakraborty, A. Bandyopadhyay, and R. Yechangunja, "A two stage data compression and decompression technique for point cloud data," *DISA 2018 - IEEE World Symp. Digit. Intell. Syst. Mach. Proc.*, vol. 4, pp. 297–302, 2018, doi: 10.1109/DISA.2018.8490525.

[53] K. Sayood, *Introduction to Data Compression*. 2012.

[54] C. Moffatt and E. Rabe, *Understanding Compression Data Compression for Modern Developers*. 2003.

[55] K. A. Ramya and M. Pushpa, "A Survey on Lossless and Lossy Data Compression Methods," *Int. J. Comput. Sci. Eng. Commun.*, vol. 4, no. 1, pp. 1277–1280, 2016.

[56] A. J. Hussain, A. Al-Fayadh, and N. Radi, *Image compression techniques: A survey in lossless and lossy algorithms*, vol. 300. 2018.

[57] G. Vijayvargiya, S. Silakari, and R. Pandey, "A Survey: Various Techniques of Image Compression," vol. 11, no. 10, 2013, [Online]. Available: http://arxiv.org/abs/1311.6877.

[58] A. Gopinath and M. Ravisankar, "Comparison of Lossless Data Compression Techniques," *Proc. 5th Int. Conf. Inven. Comput. Technol. ICICT 2020*, pp. 628–633, 2020, doi: 10.1109/ICICT48043.2020.9112516.

[59] G. Giorgi, "Lightweight Lossless Compression for N -Dimensional Data in Multi-Sensor Systems," *IEEE Sens. J.*, vol. 19, no. 19, pp. 8895–8903, 2019, doi: 10.1109/JSEN.2019.2922666.

[60] A. Najah Kahdim and M. Ebady Manaa, "Design an efficient internet of things data compression for healthcare applications," *Bull. Electr. Eng. Informatics*, vol. 11, no. 3, pp. 1678–1686, 2022, doi: 10.11591/eei.v11i3.3758.

[61] N. Kasturi and I. L. Markov, "Text Ranking and Classification using Data Compression," 2021.

[62] Y. Collet, “Zstandard Compression and the ‘application/zstd’ Media Type,” pp. 1–54, 2021, [Online]. Available: https://www.rfc-editor.org/info/rfc8878.

[63] X. Delaunay, A. Courtois, and F. Gouillon, “Evaluation of lossless and lossy algorithms for the compression of scientific datasets in netCDF-4 or HDF5 files,” *Geosci. Model Dev.*, vol. 12, no. 9, pp. 4099–4113, 2019, doi: 10.5194/gmd-12-4099-2019.

[64] E. Irmak and M. Bozdal, “Internet of Things (IoT): The Most Up-To-Date Challenges, Architectures, Emerging Trends and Potential Opportunities,” *Int. J. Comput. Appl.*, vol. 179, no. 40, pp. 20–27, 2018, doi: 10.5120/ijca2018916946.

[65] K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni, “A Systematic Survey of Industrial Internet of Things Security: Requirements and Fog Computing Opportunities,” *IEEE Commun. Surv. Tutorials*, vol. 22, no. 4, pp. 2489–2520, 2020, doi: 10.1109/COMST.2020.3011208.

[66] S. A. Fadhil, “Internet of Things security threats and key technologies,” *J. Discret. Math. Sci. Cryptogr.*, vol. 24, no. 7, pp. 1951–1957, 2021, doi: 10.1080/09720529.2021.1957189.

[67] S. Panchiwala and M. Shah, “A Comprehensive Study on Critical Security Issues and Challenges of the IoT World,” *J. Data, Inf. Manag.*, vol. 2, no. 4, pp. 257–278, 2020, doi: 10.1007/s42488-020-00030-2.

[68] D. Mendez Mena, I. Papapanagiotou, and B. Yang, “Internet of things: Survey on security,” *Inf. Secur. J.*, vol. 27, no. 3, pp. 162–182, 2018, doi: 10.1080/19393555.2018.1458258.

[69] Y. Wan, K. Xu, G. Xue, and F. Wang, “IoTArgos: A Multi-Layer Security Monitoring System for Internet-of-Things in Smart Homes,” *Proc. - IEEE INFOCOM*, vol. 2020-July, pp. 874–883, 2020, doi: 10.1109/INFOCOM41043.2020.9155424.

[70] A. Shah and M. Engineer, *A Survey of Lightweight Cryptographic Algorithms for IoT-Based Applications*, vol. 851, no. January. Springer Singapore, 2019.

[71] I. R. Chiadighikaobi and N. Katuk, “A scoping study on lightweight cryptography reviews in IoT,” *Baghdad Sci. J.*, vol. 18, no. 2, pp. 989–1000, 2021, doi: 10.21123/bsj.2021.18.2(Suppl.).0989.

[72] G. Mustafa, R. Ashraf, M. A. Mirza, A. Jamil, and Muhammad, “A review of data security and cryptographic techniques in IoT based devices,” in *ACM International Conference Proceeding Series*, 2018, no. May 2019, doi: 10.1145/3231053.3231100.

[73] A. Hameed and A. Alomary, “Security issues in IoT: A survey,” *2019 Int. Conf. Innov. Intell. Informatics, Comput. Technol. 3ICT 2019*, pp. 1–5, 2019, doi: 10.1109/3ICT.2019.8910320.

[74] N. A. Gunathilake, W. J. Buchanan, and R. Asif, “Next Generation Lightweight

Cryptography for Smart IoT Devices: : Implementation, Challenges and Applications," *IEEE 5th World Forum Internet Things, WF-IoT 2019 - Conf. Proc.*, pp. 707–710, 2019, doi: 10.1109/WF-IoT.2019.8767250.

[75] M. Shadeed and M. Moreb, "Lightweight Encryption for Multimedia in the Internet of thing(iot)," *2021 Int. Conf. Inf. Technol. ICIT 2021 - Proc.*, no. August, pp. 27–32, 2021, doi: 10.1109/ICIT52682.2021.9491671.

[76] V. K. Sarker, T. N. Gia, H. Tenhunen, and T. Westerlund, "Lightweight Security Algorithms for Resource-constrained IoT-based Sensor Nodes," *IEEE Int. Conf. Commun.*, vol. 2020-June, no. June, 2020, doi: 10.1109/ICC40277.2020.9149359.

[77] V. A. Thakor, M. A. Razzaque, and M. R. A. Khandaker, "Lightweight Cryptography Algorithms for Resource-Constrained IoT Devices: A Review, Comparison and Research Opportunities," *IEEE Access*, vol. 9, pp. 28177–28193, 2021, doi: 10.1109/ACCESS.2021.3052867.

[78] W. Kassab and K. A. Darabkh, "A–Z survey of Internet of Things: Architectures, protocols, applications, recent advances, future directions and recommendations," *J. Netw. Comput. Appl.*, vol. 163, no. September, 2020, doi: 10.1016/j.jnca.2020.102663.

[79] A. Goulart, A. Chennamaneni, D. Torre, B. Hur, and F. Y. Al-Aboosi, "On Wide-Area IoT Networks, Lightweight Security and Their Applications—A Practical Review," *Electronics*, vol. 11, no. 11, p. 1762, 2022, doi: 10.3390/electronics11111762.

[80] M. Gurunathan and M. A. Mahmoud, "A review and development methodology of a lightweight security model for IoT-based smart devices," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 2, pp. 125–134, 2020, doi: 10.14569/ijacsa.2020.0110217.

[81] V. Rao and K. V. Prema, "A review on lightweight cryptography for Internet-of-Things based applications," *J. Ambient Intell. Humaniz. Comput.*, vol. 12, no. 9, pp. 8835–8857, 2021, doi: 10.1007/s12652-020-02672-x.

[82] V. A. Thakor, M. A. Razzaque, and M. R. A. Khandaker, "Lightweight cryptography for IoT: A state-of-the-art," *arXiv*, pp. 1–19, 2020.

[83] E. R. Naru, H. Saini, and M. Sharma, "A recent review on lightweight cryptography in IoT," *Proc. Int. Conf. IoT Soc. Mobile, Anal. Cloud, I-SMAC 2017*, pp. 887–890, 2017, doi: 10.1109/I-SMAC.2017.8058307.

[84] G. Hatzivasilis, K. Fysarakis, I. Papaefstathiou, and C. Manifavas, "A review of lightweight block ciphers," *J. Cryptogr. Eng.*, vol. 8, no. 2, pp. 141–184, 2018, doi: 10.1007/s13389-017-0160-y.

[85] D. J. Wheeler and R. M. Needham, "TEA, a tiny encryption algorithm BT - Fast Software Encryption," *Lect. Notes Comput. Sci.*, pp. 363–366, 1995.

[86] M. Jukl and J. Čupera, “Using of tiny encryption algorithm in CAN-Bus communication,” *Res. Agric. Eng.*, vol. 62, no. 2, pp. 50–55, 2016, doi: 10.17221/12/2015-RAE.

[87] M. B. Abdelhalim, M. El-Mahallawy, and M. A. A. Elhennawy, “Design and Implementation of an Encryption Algorithm for use in RFID System,” *Int. J. RFID Secur. Cryptogr.*, vol. 2, no. 1, pp. 51–57, 2013, doi: 10.20533/ijrfidsc.2046.3715.2013.0007.

[88] M. S. Novelan, A. M. Husein, M. Harahap, and S. Aisyah, “SMS Security System on Mobile Devices Using Tiny Encryption Algorithm,” *J. Phys. Conf. Ser.*, vol. 1007, no. 1, 2018, doi: 10.1088/1742-6596/1007/1/012037.

[89] NIST, “Recommendation for Key Management,” *Spec. Publ. 800-57 Part 1*, 2007.

[90] R. Roman, C. Alcaraz, J. Lopez, and N. Sklavos, “Key management systems for sensor networks in the context of the Internet of Things,” *Comput. Electr. Eng.*, vol. 37, no. 2, pp. 147–159, 2011, doi: 10.1016/j.compeleceng.2011.01.009.

[91] R. Alvarez, C. Caballero-Gil, J. Santonja, and A. Zamora, “Algorithms for lightweight key exchange,” *Sensors (Switzerland)*, vol. 17, no. 7, pp. 1–14, 2017, doi: 10.3390/s17071517.

[92] R. Mohan Naik, S. V. Sathyanarayana, and T. K. Sowmya, “Key Management Using Elliptic Curve Diffie Hellman Curve 25519,” *MPCIT 2020 - Proc. IEEE 3rd Int. Conf. "Multimedia Process. Commun. Inf. Technol.*, pp. 33–39, 2020, doi: 10.1109/MPCIT51588.2020.9350454.

[93] R. R. Ahirwal and M. Ahke, “Elliptic Curve Diffie-Hellman Key Exchange Algorithm for Securing Hypertext Information on Wide Area Network,” *Int. J. Comput. Sci. Inf. Technol.*, vol. 4, no. 2, pp. 363–368, 2013.

[94] G. Shrividhiya, K. S. Srujana, S. N. Kashyap, and C. Gururaj, “Robust data compression algorithm utilizing LZW framework based on huffman technique,” *2021 Int. Conf. Emerg. Smart Comput. Informatics, ESCI 2021*, pp. 234–237, 2021, doi: 10.1109/ESCI50559.2021.9396785.

[95] X. H. Nguyen, X. D. Nguyen, H. H. Huynh, and K. H. Le, “Realguard: A Lightweight Network Intrusion Detection System for IoT Gateways,” *Sensors*, vol. 22, no. 2, pp. 1–18, 2022, doi: 10.3390/s22020432.

[96] R. Sayal, C. Subbalakhmi, and H. S. Saini, “Secure and Lightweight Encryption Model for IoT Surveillance Camera,” vol. 9, no. 1, pp. 182–185, 2020.

[97] A. H. A. Al-Ahdal, G. A. AL-Rummana, and N. K. Deshmukh, “A Robust Lightweight Algorithm for Securing Data in Internet of Things Networks,” *Lect. Notes Data Eng. Commun. Technol.*, vol. 55, no. 12, pp. 509–521, 2021, doi: 10.1007/978-981-15-8677-4_41.

[98] M. Usman, I. Ahmed, M. I. Aslam, S. Khan, and U. A. Shah, “SIT: A Lightweight Encryption Algorithm for Secure Internet of Things,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 1, pp. 402–411, 2017, doi: 10.14569/IJACSA.2017.080151.

[99] M. Imdad, S. N. Ramli, and H. Mahdin, “An Enhanced Key Schedule Algorithm of PRESENT-128 Block Cipher for Random and Non-Random Secret Keys,” *Symmetry (Basel).*, vol. 14, no. 3, pp. 1–22, 2022, doi: 10.3390/sym14030604.

الخلاصة

يمكن اعتبار إنترنت الأشياء كائن مادي يربط البيانات ويتبادلها مع كائنات أخرى عبر الشبكات. أدى الظهور الأخير لإنترنت الأشياء إلى تركيز الكثير من الاهتمام عليها. يولد جهاز إنترنت الأشياء كمية هائلة من بيانات الدفق التي تؤثر على معدل عرض النطاق الترددي للاتصال. نظرًا للحجم الهائل لبيانات إنترنت الأشياء ، فإنها تتطلب تقنيات للتعامل مع هذا القيد ، مثل خوارزميات الضغط لتقليل حجم البيانات ، وحفظ النطاق الترددي للشبكة ، وزيادة سرعة نقل بيانات إنترنت الأشياء.

تنقل بيئة إنترنت الأشياء أنواعًا مختلفة من معلومات الرعاية الصحية الشخصية والبيانات الهامة وعناصر أخرى عبر الإنترنت. تصبح هذه البيانات هدفًا لقضايا الأمان. لذلك ، فإن أمان هذه البيانات أمر حيوي للحفاظ على سرية وسلامة بيانات إنترنت الأشياء. نظرًا للموارد المحدودة لأجهزة إنترنت الأشياء ، فإنها تتطلب خوارزميات تشفير خاصة مثل خوارزميات التشفير خفيفة الوزن (LWC).

يهدف النظام المقترح إلى التغلب على الحجم الهائل من البيانات والمشاكل الأمنية في إنترنت الأشياء. من خلال الجمع بين خوارزمية ضغط البيانات وتقنيات التشفير خفيفة الوزن تسمى نهج Compcrypt. استخدمت خوارزمية Zstandard لضغط بيانات مستشعر IoT ، بينما استخدمت خوارزمية TEA لتشفير البيانات المضغوطة بمفتاح سري مشترك تم إنشاؤه باستخدام بروتوكول Elliptic Curve Diffie Hellman من جانب العميل. تتم عملية فك الضغط وفك التشفير على جانب الخادم باستخدام نفس الخوارزميات.

يعتمد النظام المقترح على الأجهزة ، مثل مستشعر درجة الحرارة ، و SPO2 (مستشعر معدل ضربات القلب ومستوى الأكسجين) ، ونموذج Raspberry Pi 4. وأظهرت نتائج نظام Compcrypt المقترح أفضل من حيث نسبة الضغط 81 ٪ ، وقت التشفير 108.5 مللي ثانية ، والإنتاجية 1099.82 كيلوبت في الثانية لحجم البيانات 118.78 كيلوبايت. يعتمد تقييم قوة الأمان لنظام Compcrypt المقترح على إنتروبيا مثل 7.998 ، وتأثير AE بنسبة 94 ٪ لحجم 128 بايت من نص التشفير.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية تكنولوجيا المعلومات
قسم شبكات المعلومات

تحسين أمان إنترنت الأشياء باستخدام ضغط البيانات وتقنية التشفير خفيفة الوزن

رسالة مقدمة

إلى مجلس كلية تكنولوجيا المعلومات في جامعة بابل والتي هي جزء من متطلبات

الحصول على درجة الماجستير في تكنولوجيا المعلومات / شبكات المعلومات

من قبل الطالب

احمد نجاح كاظم كريم

باشراف

أ.م.د مهدي عبادي مانع مهدي

هـ 1444 م 2022