Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon

IMPROVED LIGHTWEIGHT CRYPTOGRAPHY APPROACH FOR REAL TIME STREAMING DATA USING CLOUD COMPUTING

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

ANAAM GHANIM HILAL GUMMA

Supervised by

Asst. Prof.Dr. Mehdi Ebady Manaa Mehdi

2023 A.D. **1445 A.H.**

يَرفعُ اللهَ الذينَ امنوا مِنكُم والذينَ
اوتوا الِعلم دَرجات

صدق اللَّهُ العظيم

سورة المجادلة /الآية 11

2023 A.D. **1445 A.H.**

2023 A.D. **1445 A.H.**

Supervisor Certification

I certify that the thesis entitled (IMPROVED LIGHTWEIGHT CRYPTOGRAPHY APPROACH FOR REAL TIME STREAMING DATA USING CLOUD COMPUTING) was prepared under my supervision at the department of Information Networks/ College of Information Technology/ University of Babylon as partial fulfillment of the requirements of the degree of Master in Information Technology-Information Networks.

Signature:

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

Date: / /2023

The Head of the Department Certification

In view of the available recommendations, I forward the thesis entitled "IMPROVED LIGHTWEIGHT CRYPTOGRAPHY APPROACH FOR REAL TIME STREAMING DATA USING CLOUD COMPUTING" for debate by the examination committee.

Signature:

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

Head of Information Networks Department

Date: / /2023

Certification of the Examination Committee

We hereby certify that we have studied the thesis entitled (**IMPROVED LIGHTWEIGHT CRYPTOGRAPHY APPROACH FOR REAL TIME STREAMING DATA USING CLOUD COMPUTING**) presented by the student (**Anaam ghanim hilal**) and examined him/her in its content and what is related to it, and that, in our opinion, it is adequate with (**Viva Result**) standing as a thesis for the degree of Master in Information Technology-Information Networks.

Signature:
Name:
Title:
Date: / / 2023
(**Chairman**)

Signature:
Name:
Title:
Date: / / 2023
(**Member**)

Signature:
Name:
Title:
Date: / / 2023
(**Member**)

Signature:
Name:
Title:
Date: / / 2023
(**Member and Supervisor**)

Approved by the Dean of the College of Information Technology, University of Babylon.

Signature:
Name: Prof. Dr. Wesam S. Bhaya
Date: / / 2023
(**Dean of Collage of Information Technology**)

Dedication

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: **Anaam Ghanim Hill**

Date: **/ / 2023**

Acknowledgement

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.

I dedicate this work and give special thanks to those who encouraged me to continue my scientific career, to my dear parents, my uncle the martyr (Salem Hilal), My beloved husband, and my dear children.

Researcher

Abstract

The Internet of Things (IoT) is a fast-growing field that involves connecting a wide variety of devices and sensors to the Internet to collect and share data. The vast amount of data generated by these devices requires secure storage and processing, which is where cloud computing comes in. Cloud computing is an effective solution for storing IoT data, but a large amount of data is also vulnerable to security threats and high latency when transmitted from IoT devices to the cloud, so need to provides high level of security for data to secure it from threats.

The proposed work aims to address security issues in IoT-Cloud computing systems by implementing an authentication phase and encryption mechanisms between IoT devices and the cloud. The system utilizes hybrid lightweight encryption algorithms (PRINCE and SPECK) to encrypt sensor data using a shared secret key generated through the Elliptic Curve Diffie-Hellman (ECDH) protocol on the client side. Additionally, a hashing algorithm is used to authenticate the encrypted data, ensuring its authenticity and integrity. On the server side, decryption processes are performed using the same encryption algorithms. The proposed system implemented a temperature sensor and an SPO2 sensor that are both connected to the human body. The temperature sensor is used to monitor body temperature, while the SPO2 sensor is used to monitor heart rate and oxygen levels. To make it all work, we used a Raspberry Pi model 3 B+ as the central hub. Overall, this system has proven to be quite effective in helping us keep track of important health metrics.

The results of the proposed system indicate improved encryption time, with an average encryption time of 512 milliseconds for a data size of 10 kilobytes. The security strength evaluation of the system is based on entropy, which measures the randomness and provide high level of security for data. The proposed system achieves an entropy value of 7.999 for a data size of 10 kilobytes, indicating a high level of security.

Declaration Associated with this Thesis

Some of the works presented in this thesis have been published or accepted as listed below.

(First Paper)

- Name of Journal: Lecture Notes in Networks and System.
- Paper Title: Securing Real-Time Data Streams with Cloud-Enabled Lightweight Cryptography.
- Authors:
  - Anaam Ghanim Hilal
  - Asst. Prof.Dr. Mehdi Ebady Manaa

Information Network Department, Information Technology College, Babylon University.

(Second Paper)

- Name of Conference: Sixth International Iraqi Conference on Engineering Technology and its Applications (6th IICETA).
- Paper Title: Data Protection between IoT devices and Cloud Computing Using PRINCE Lightweight Algorithm.
- Authors:
  - Anaam Ghanim Hilal
  - Asst. Prof.Dr. Mehdi Ebady Manaa

Information Network Department, Information Technology College, Babylon University.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Description
IoT	Internet of Things
LWC	Lightweight Cryptography
NIST	National Institute of Standards and Technology
GE	Gate Equivalent
DES	Data Encryption Standard
AES	Advanced Encryption Standard
RSA	Rivest–Shamir–Adleman
SHA-256	Secure Hash Algorithm 256-bit
HMAC	Hash-based Message Authentication Code
ECDH	Elliptic Curve Diffie Hellman
HTTPs	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
XML	Extensible Markup Language
WSN	Wireless Sensor Network
RFID	Radio Frequency Identification
WLAN	Wireless Local Area Network
HLCA	Hybrid Lightweight Cryptography Approach

Chapter One

Introduction

CHAPTER ONE
INTRODUCTION

1.1 Introduction

The Internet of Things (IoT) has emerged as a result of the fast development of information and communication technology, IoT is a paradigmatic technology, that enables data transmission between linked electrical and electronic devices. IoT and automation are anticipated to make it easier for people to transmit information in the context of data communication as a whole, enhancing both daily life and commercial operations. Smart homes, smart cities, smart cars, healthcare, smart grids, and smart agriculture are just a few examples of IoT applications. They mostly employ wearable's and sensors that can connect and other networked devices[1].

It is a concept that things can be equipped with identification, sensing, networking, and processing capabilities so that they can communicate with one another, other devices, and online services to carry out a specific task. Eventually, Internet of Things devices will be ubiquitous, context-aware, and able to exhibit ambient intelligence[2].

It makes it possible for devices with extremely low computing power, memory size, power consumption, physical size, and cost to communicate, compute, and make decisions in the communication network. Due to the limited resources of such confined devices, security is one of the difficult concerns in IoT[3].

It is a rapidly growing technology, with the number of connected devices 75 billion are anticipated by 2025[4]. The data generated from IoT devices could run into several fresh and escalating challenges, such as administrative

complexity, privacy, transmission, lifetime, support technologies, and security assaults like eavesdropping and alteration. Several requirements and characteristics, including security, privacy, and integrity must be addressed to develop a successful system[5].

As previously indicated, because IoT devices have limited resources, they cannot rely on the usual encryption technique and instead require a unique encryption algorithm with low resource requirements and simple operations. Lightweight cryptographic algorithms (LWC) that offer the same level of security are preferred, according to the National Institute of Standards and Technology (NIST)[6].

Lightweight cryptography, in contrast to certain other ways of encrypting data, doesn't need a particular device setup to function. When using devices with a lot of resources to share, it performs effectively (such as cellphones, tablets, PCs, servers, etc.). The primary objective of lightweight cryptography is to lower overall hardware and software implementation costs. Software implementation parameters include performance, RAM use, and size (bytes per cycle). On the other side, hardware has Gate Equivalence(GE), Code Size, and Memory Consumption (RAM)[7].

IoT devices produce a lot of data that conventional storage systems and processing platforms might not be able to handle. Finding appropriate systems that rely on vast resource pools, like cloud computing, is therefore crucial, to manage and process the data produced by IoT devices[4].

IoT and cloud computing are currently combined in a very fascinating technology. The Internet of Things (IoT) is all about networks, sensors, actuators, and widely scattered smart devices with a finite amount of storage and processing

capacity as well as privacy and security issues. The Internet of Things, generates enormous amounts of data also known as big data, as a result of communication between connected devices, placing a great deal of strain on the internet infrastructure. This problem has been addressed by the development of cloud computing, which provides on-demand and virtual services like boundless storage and processing power. Cloud computing is continually evolving to provide outstanding help to the Internet of Things. IoT produces ongoing or streaming data, while cloud computing provides remote data storage and access, allowing developers to start working right away. Customers are charged for the services they use under a pay-per-use model offered by cloud service providers[8]. the Interaction between IoT devices and Cloud is shown in Figure (1.1).

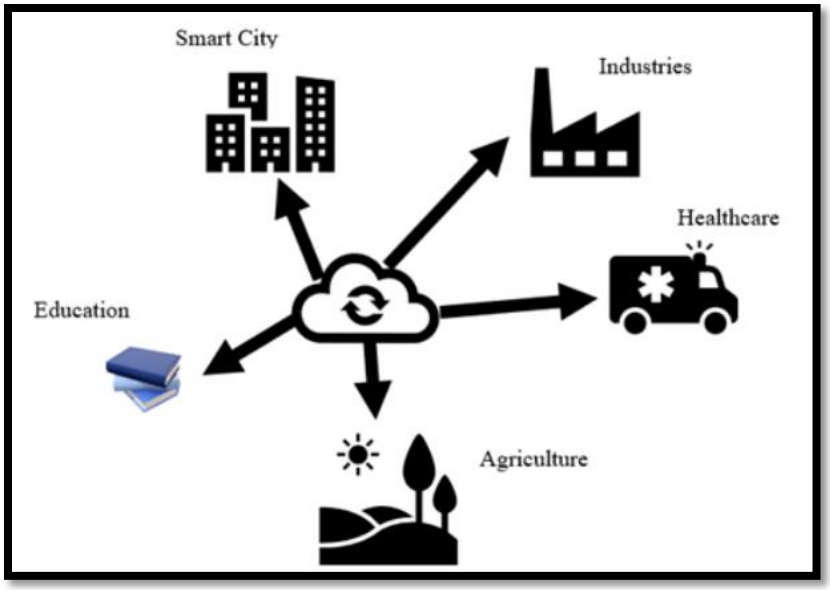


Figure 1.1:Interaction IoT and Cloud Computing [9]

Cloud computing is “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider

interaction”[10]. Its goal is to provide an easy-to-use endpoint access system without needing the acquisition of software, a platform, or physical network equipment; instead, these tasks are outsourced to third parties[11].

Cloud computing is a powerful platform for delivering applications over the Internet or a private network. cloud is the following technology is seen as being used in data processing and retrieval. These tools support data management and processing across a range of services and deployment patterns. Users are given the option to lease services in a certain range when receiving cloud services from cloud computing providers under a leased service delivery model. Today's big data situation calls for cloud data storage, which offers customers the same level of ease as on-premises devices[12]. As a result, cloud computing is now being deployed centrally on a worldwide scale and is becoming a necessary component of IoT data processing. Big data presents various challenges for cloud-powered IoT, including applications that require low latency, real-time processing, bandwidth restrictions, and high power consumption[13].

IoT devices and the cloud typically exchange information through open channels, which leaves them exposed to interception. Since the IT sector has consistently worried about hackers gaining access to sensitive data, cloud service providers and users have used a variety of security methods to protect cloud data, including encryption[14].

Encryption is an essential security mechanism for protecting data stored and transmitted to cloud systems from being accessed by unauthorized parties. To be effective, encryption mechanisms must have a high level of security, be efficient in terms of performance, and be cost-effective to implement[15].

1.2 Related work

In (Yu et al., 2020) The authors proposed using an attribute-based sign encryption scheme with hybrid access policy and verifiable outsourced decryption (LH-ABSC), and they outsource most signing overload and verification load to fog nodes, to provide high security level for data that transmit to cloud . ABSC can achieve message confidentiality and ciphertext unforgeability simultaneously and efficiently. And KPABS is 6more suitable for the situation that the central manager wants to control the anonymous signature in a certain authorized structure. While CPABE can achieve fine-grained one-to-many data sharing while keeping the right of who can decrypt in data owner, which is common for IoT cases. our signature scheme has signature size and satisfies public verification which is important for IoT system. In the end, they proved our scheme satisfies both CCA secure and CMA secure[16].

In (Thabit et al., 2021) The authors have proposed a hybrid lightweight cryptography encryption algorithm called NLCA, which is designed to protect data in the cloud environment. This algorithm uses a symmetric encryption algorithm that combines SP and Feistel architectural strategies. The NLCA algorithm uses a 16-byte (128-bit) block cipher and a 16-byte (128-bit) key with five rounds to encrypt data. To make it difficult for attackers to break the encryption, the NLCA algorithm uses mixed operations across several algebraic classes, such as XOR and addition operations. According to the experimental data, the NLCA algorithm offers excellent security at a low computational cost. It provides a strong security level and a notable improvement in encryption and decryption. This algorithm is well-suited for the cloud computing environment, as it is effective for quick data gathering and processing. Overall, the NLCA algorithm appears to be a promising

approach for protecting data in the cloud environment, offering strong security and efficient computation at a low cost[17].

In (Siam et al., 2021) The authors proposed using an Advanced Encryption Standard (AES) algorithm to protect data that trasfer from IoT devices to cloud computing, implemented system is designed to measure the key health parameters: heart rate (HR), blood oxygen saturation (SpO2), and body temperature, simultaneously. captured physiological signals are processed and encrypted using AES algorithm before sending them to the cloud. An ESP8266 integrated unit is used for processing, encryption, and providing connectivity to the cloud over Wi-Fi.On the other side, trusted medical organization servers receive and decrypt the measurements and display the values on the monitoring dashboard for the authorized specialists. their proposed system measurements are compared with a number of commercial medical devices. Results demonstrate that the measurements of the proposed system are within the 95% confidence interval. Moreover, Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Relative Error (MRE) for the proposed system are calculated as 1.44, 1.12, and 0.012, respectively, for HR, 1.13, 0.92, and 0.009, respectively, for SpO2, and 0.13, 0.11, and 0.003, respectively, for body temperature. results demonstrate the high accuracy and reliability of the proposed system[18].

In (Hussam et al., 2021) The authors have proposed the LMGHA128-bit algorithm, which is designed to provide high encryption efficiency and security for data transmission to the cloud. The algorithm utilizes a chaos system to generate 5-D chaos keys, and a hybrid of the P layer from the Present algorithm and block shuffle from the TWINE algorithm is applied with 32 rounds to further enhance the algorithm's performance. Their proposed algorithm uses a Feistel-type structure, It divides the input data into two equal sections of 64 bits each and applies the block shuffle of TA to process the first segment and the P-layer of PA to encrypt the

second segment. Finally, the two encrypted segments are added to create 128-bit encrypted data. The experimental results show that the proposed algorithm, achieves high encryption efficiency, resistance to a variety of attacks, memory optimization, and optimal execution time, with the execution time for a file of size 10MB being 11014.7665ms [19].

In (Abroshan., 2021) The authors have proposed an efficient cryptographic solution to increase the security of cloud computing by using an improved version of the Blowfish encryption algorithm and an elliptic curve-based approach. The proposed solution involves encrypting the data using the Blowfish algorithm, which is a widely used and trusted encryption technique, while the key is encrypted using elliptic curve cryptography, which provides additional security and efficiency benefits. MD5-based digital signatures are used to ensure the security of the data. The experimental result shows that the proposed algorithm performs better than other encryption techniques such as AES, DES, 3DES, and RSA in terms of throughput, memory usage, and runtime when using a file with data size (i.e., the 1024 KB). Their proposed technique reduced 230, 230, and 130 milliseconds in comparison with symmetric algorithms (DES, 3DES, and AES) and 731 milliseconds in comparison with the asymmetric algorithm (i.e., RSA). This show that the proposed technique is faster than the other techniques(AES, DES, 3DES, and RSA) [20].

In (Karuppusamy et al., 2021) the authors proposed a robust lightweight block cipher. The strategy is developed to make the encryption process more difficult for attackers and to generate confusion for them. the proposed method combines Feistel and SP networks, and This algorithm's main goal is to offer defense against most attacks, including Square Attacks, Interpolation Attacks, and Linear and Differential Cryptanalysis. The method provides good flexibility for

improving performance in the Internet of Things when compared to other algorithms. For 64-bit encryption, an 80-bit key is therefore used. Due to the merging of SP and Feistel network architectures, this method offers security performance and complexity [21].

In (Jaber and Manaa., 2021) The authors have proposed an application of fog computing with cloud services to reduce service latency and network congestion in IoT systems. The proposed system involves a three-layered IoT-fog computing model designed to enable secure data transfer between IoT devices and the cloud while minimizing potential network latency and reaction time. the system implements authentication, encryption, and data mining cluster analysis stages with cloud computing in a real environment. The authentication step is designed to address security and accuracy concerns in IoT-fog computing systems, while the connection between the cloud layer and the fog is encrypted to ensure security. the system relies on several medical data sensors, including temperature, heart echo, the oxygen level in the blood, and heart rate sensors. The fog server computes the digest between the sensor layer and the fog layer for the authentication process to transfer the sensor data. AES and a shared key created by ECDiffieHellmanCng are used to implement the encryption process between the fog and cloud servers. in data mining cluster analysis, k-means is used to cluster the data in the cloud server. The collected results show that the model was successful in locating a good network latency while boosting connection security and accuracy. The time of encryption between fog and cloud computing is between 1.1 milliseconds and 1.3 milliseconds, and the latency is calculated based on the test bed distance range. The system's performance in healthcare data analysis is evaluated using cohesion and silhouette index. The results show a cohesion of 98.195 and a silhouette index of 0.769, indicating that their proposed system performs well in healthcare data analysis [22].

In (Thabit et al., 2022) The authors have proposed a cryptographic technique that combines two levels of encryption to enhance cloud computing security. The first layer is a New Effective Light-Weight cryptographic algorithm (N.E.L.C.), which uses a symmetric key technique to encrypt data with a block encryption size of 8-16 bytes and a key size of 8-16 bytes. The second layer is the multiplicative homomorphic property of the RSA algorithm, which improves data security. The N.E.L.C. algorithm uses Network substitution-permutation (S.P.) properties with Feistel structure features to generate confusion and diffusion, increasing encryption complexity. The algorithm also incorporates basic Boolean operations such as XOR, Ex-NOR, and sequencing. The algorithm uses mathematical operations to spread noise and confuse the receiver, with each crypto round being crucial. Their proposed method restricts the algorithm to a maximum of 7 cycles, with each cycle requiring 32 bits of cryptographic data to operate, resulting in energy savings. Experimental results show that the proposed algorithm provides a high level of security and significantly improves the encryption's throughput, memory usage, and execution time, with the execution time of the file in size 1MB being 2.88s [23].

In (Alexander Suresh and Jemima Priyadarsini., 2022) The authors proposed a new encryption algorithm called Enhanced Modern Symmetric Data Encryption (EMSDE) to protect data in a cloud-based IoT environment. EMSDE uses a 64-bit block and has eight rounds of encryption to ensure that the resulting encrypted text is impenetrable to malicious users. The results indicate that EMSDE is more secure than other encryption techniques such as DES and Blowfish, with a security level of 90% compared to 78% for DES and 84% for Blowfish. Additionally, the proposed algorithm is faster than existing encryption techniques for both encryption and decryption [24].

In (Das and Namasudra., 2022) The authors proposed a hybrid encryption strategy that combines both symmetric and asymmetric methods to protect sensitive medical information in IoT-enabled healthcare infrastructure. The proposed method uses the AES symmetric encryption algorithm and Serpent Encryption Algorithm. The proposed system used The elliptic curve-based digital signature is also used to ensure data integrity and prevent data tampering. The proposed hybrid encryption strategy provides enhanced security for healthcare data by making it resistant to common attacks such as forgery, collision, and man-in-the-middle attacks. The experimental result shows that the proposed algorithm is faster in execution with encryption and decryption, where the key generation time is 62.56 ms, the encryption time of 23 ms, and a decryption time of 18 ms. The total execution time for the proposed scheme, including timestamp generation and signature generation, is 134.96 ms [25].

In (A. S. Kadhim et al., 2022) The authors proposed a hybrid lightweight cipher solution that utilizes the Present and Tea encryption algorithms to improve data security in IoT-based healthcare systems and uses ECC authentication and key generation technology to prevent unauthorized access to encrypted data. This method aims to enhance the secure transmission of data from IoT devices to healthcare staff.
The proposed solution improves network performance by reducing network latency, optimizing channel resource usage, and preserving network performance while providing a higher level of security through the use of PRESENT and TEA with ECC. The study found that the proposed system is effective in reducing packet loss rates when compared to a case study without security. The uploaded image data indicates that the proposed method resulted in an average payload throughput of 68.74475 kbps, an average delay of 17.158 s, and a total packet loss rate of 3.835%.

These results suggest that the proposed method is effective in improving data security in IoT-based healthcare systems[26].

In (Jammula et al., 2022) the authors proposed using The LWC-ABE (Lightweight Cryptography-Attribute-Based Encryption) method to improve the security performance of IoT environments against various attacks. The method uses multiple trusted authority environments to enhance security. The simulation results demonstrate that the proposed method reduces encryption and decryption times for multiple users and different message sizes compared to conventional approaches. The numerical outcomes of the proposed method show significant improvement in the performance of encryption and decryption times, with encryption time being 0.000835 and decryption time being 0.000310. This indicates that the proposed method is efficient and effective in providing secure communication in IoT environments[27].

In (Khalil., 2022) proposed system uses two lightweight encryption algorithms hummingbird and speck called TPHLE. With the help of three alternative recommended strategies, the two algorithms are hybridized. Each suggestion is a hybrid lightweight encryption algorithm made up of elements from both the speck and the hummingbird algorithms, but each method combines these elements differently. Additionally, the Chaos system is utilized to create random keys for encryption methods to increase their effectiveness. All three approaches passed all of the NIST's testing, and the proposed light-weight encryption algorithms produced acceptable encryption results. The encryption and decryption time is calculated for each proposed hybrid algorithm on different frame sizes 128x128, 256x256, and 512x512. For the frame size 256x256, the encryption and decryption time is calculated with a different number of rounds each time: 10 rounds, 14 rounds, and 16 rounds. The encryption results of the first proposed

hybrid lightweight algorithm are designed by merging the Speck algorithm (one round) with the Hummingbird algorithm to encryption the average of 7-frame, the 256x256 frame size is 16.024 ms and the decryption time is 15.927 ms. The second proposal is designed from combined the Speck algorithm (with 10 rounds to reduce the Speck encryption time) with the Hummingbird algorithm (with 3 rounds), giving encryption results for the same previous specifications: 33.81743 ms for encryption and 30.72014 ms for decryption. The third proposed is designed by merging the Hummingbird algorithm (three rounds) with the SPECK algorithm (one round). The results for the third proposal are 13.996 ms for encryption and 13.583 ms for decryption. As shown in the results, the third proposal is the fastest, because the two algorithms work at the same time[28].

In (Najah and Kareem., 2022) proposed solution seeks to address the Internet of Things' enormous data volume and security issues. using the Compcrypt approach, which combines a data compression algorithm and simple cryptography methods. It employed the Zstandard method to compress data from IoT sensors, and the Tiny Encryption Algorithm (TEA) to encrypt it using a shared secret key that is created on the client side using the Elliptic Curve Diffie Hellman protocol. The decompression and decryption processes are done on the server side with the same algorithms. The suggested Compcrypt system's results demonstrated improved throughput of 1099.82 Kbps for a data size of 118.78 Kb, an 81% compression ratio, and encryption times of 108.5 ms. Entropy is used to determine the security strength of the proposed Compcrypt system as 7.998, and the avalanche effect as 94% for the 128-byte size of cipher text[29].

Table 1.1: Summary of related work

Ref. No	**method**	**Amis**	**result**	**year**
[16]	The authors proposed using an attribute-based sign cryption scheme with hybrid access policy and verifiable outsourced decryption (LH-ABSC), and they outsource most signing overload and verification load to fog nodes	To provide high security level for data that transmit to cloud	they proved our scheme satisfies both CCA secure and CMA secure.	2020
[17]	The authors have Proposed the New Lightweight Cryptographic Algorithm (NLCA).	safeguard data stored in the cloud and ensure that it remains private, secure, and unaltered by any third parties.	The experimental results show that the NLCA approach, which provides high security at low computational cost, improved encryption and decryption significantly. In terms of quick data gathering and processing, It works much better in the cloud computing setting.	2021
[18]	The authors proposed using an Advanced Encryption Standard (AES) algorithm.	To protect data that transfer from IoT devices to cloud computing.	Results demonstrate that the measurements of the proposed system are within the 95% confidence interval. Moreover, Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Relative Error (MRE) for the proposed system are calculated as 1.44, 1.12, and	2021

			0.012, respectively, for HR, 1.13, 0.92, and 0.009, respectively, for SpO2, and 0.13, 0.11, and 0.003, respectively, for body temperature. results demonstrate the high accuracy and reliability of the proposed system.	
[19]	The authors have proposed the LMGHA128-bit algorithm, The algorithm utilizes a chaos system to generate 5-D chaos keys.	The main aim of this algorithm is to provide high encryption efficiency and security for data transmission to the cloud.	The experimental results show that the proposed algorithm, achieves high encryption efficiency, resistance to a variety of attacks, memory optimization, and optimal execution time, with the execution time for a file of size 10MB being 11014.7665ms.	2021
[20]	The authors have proposed a robust cryptographic approach based on an improved Blowfish algorithm and elliptic curve cryptography. The key will be encrypted with the elliptic curve method, and the data will be encrypted with Blowfish. The data integrity is ensured by MD5-based digital	increase security, performance, and data integrity	The experimental result shows that the proposed algorithm performs better than other encryption techniques such as AES, DES, 3DES, and RSA in terms of throughput, memory usage, and runtime when using a file with data size (i.e., the 1024 KB). The proposed technique reduced 230, 230, and 130 milliseconds in comparison with symmetric algorithms (DES,	2021

	signatures.		3DES, and AES) and 731 milliseconds in comparison with the asymmetric algorithm (i.e., RSA). This show that the proposed technique is faster than the other techniques(AES, DES, 3DES, and RSA).	
[21]	The authors have proposed a method combines Feistel and SP networks, it uses an 80-bit key for 64-bit encryption	the main goal is to offer defense against most attacks, including Square Attacks, Interpolation Attacks, and Linear and Differential Cryptanalysis.	The result shows that The proposed algorithm offers high-security performance and complexity due to the integration of SP and Feistel network designs.	2021

[22]	The authors proposed a mechanism for encrypting data traveling between fog and cloud servers using the Advanced Encryption Standard (AES) algorithm and a shared key generated via the ECC. K-means is a data mining cluster analysis method that is used to cluster data stored in cloud servers. To complete the authentication procedure for the transfer of the sensor data, the fog server computed the digest between the sensor layer and the fog layer.	Minimize potential network delay and response time while enabling safe data flow between the fog layer and the cloud.	The obtained results show that the time of encryption between fog and cloud computing was between 1.1 and 1.3 ms, and the latency is calculated as 1 ms, 2.5 ms, 5 ms, 8, 10.5 according to the distance range used in the testing environment. In a cloud-based k-means analysis of healthcare data, the cohesion and silhouette index evaluation metrics were 98.195 and 0.769, respectively.	2021
[23]	The authors proposed a new cryptographic approach, they use a two-layer encryption process, with the first layer consisting of the New Effective Light-Weight cryptographic algorithm (N.E.L.C.) and the second layer utilizing the multiplicative homomorphic property of the R.S.A.	The main aim is to improve the security of cloud computing.	Experimental results show that the proposed algorithm provides a high level of security and significantly improves the encryption's throughput, memory usage, and execution time, with the execution time of the file in size 1MB being 2.88 s.	2022

[24]	The authors proposed a new encryption algorithm called Enhanced Modern Symmetric Data Encryption (EMSDE)	Protect data as it moves from IoT devices to the cloud	The results indicate that EMSDE is more secure than other encryption techniques such as DES and Blowfish, with a security level of 90% compared to 78% for DES and 84% for Blowfish. Additionally, the proposed algorithm is faster than existing encryption techniques for both encryption and decryption.	2022
[25]	The authors proposed a hybrid encryption strategy (AES and Serpent Encryption Algorithm). And The proposed system used The elliptic curve-based digital signature	The proposed hybrid encryption strategy provides enhanced security and ensure data integrity for healthcare data by making it resistant to common attacks such as forgery, collision, and man-in-the-middle attacks.	The experimental result shows that the proposed algorithm is faster in execution with encryption and decryption, where the key generation time is 62.56 ms, the encryption time of 23 ms, and a decryption time of 18 ms. The total execution time for the proposed scheme, including timestamp generation and signature generation, is 134.96 ms.	2022
[26]	The authors proposed a hybrid lightweight algorithm combine of (PRESENT and TEA), and they used the ECC algorithm for the authentication system.	improves the security of data transmission between IoT devices and healthcare providers.	the result of the suggested system, which was tested using a specific file upload image data. The evaluation shows that the system's average payload throughput is 68.74475 Kbps, the average delay is 17.158 s, and the	2022

			total packet loss rate is 3.835%.	
[27]	the authors have proposed using The LWC-ABE (Lightweight Cryptography-Attribute-Based Encryption) method.	to improve the security performance of IoT environments against various attacks.	The numerical outcomes of the proposed method show significant improvement in the performance of encryption and decryption times, with encryption time being 0.000835 and decryption time being 0.000310. This indicates that the proposed method is efficient and effective in providing secure communication in IoT environments.	2022
[28]	The authors have proposed system using two lightweight encryption algorithms hummingbird and speck called TPHLE, And using Chaos system was utilized to create random keys	To make data more secure and reduce latency between IoT devices and server	The experimental result shows that the encryption results of the first proposed hybrid lightweight algorithm to encryption the average of 7-frame, the 256x256 frame size is 16.024 ms and the decryption time is 15.927 ms. The second proposal gave encryption results for the same previous specifications: 33.81743 ms for encryption and 30.72014 ms for decryption. The results for the third proposal are 13.996 ms for encryption and 13.583 ms for decryption. As shown in the results, the third proposal is the	2022

			fastest, because the two algorithms work at the same time.	
[29]	The authors have proposed a Compcrypt approach, which combines a data compression algorithm and simple cryptography methods. It employed the Zstandard method to compress data from IoT sensors, and the Tiny Encryption Algorithm (TEA) to encrypt it using a shared secret key that was created on the client side using the Elliptic Curve Diffie Hellman protocol.	address the Internet of Thing's enormous data volume and security issues	The suggested Compcrypt system's results demonstrated improved throughput of 1099.82 Kbps for a data size of 118.78 Kb, an 81% compression ratio, and encryption times of 108.5 ms. Entropy is used to determine the security strength of the proposed Compcrypt system as 7.998, and the avalanche effect as 94% for the 128-byte size of cipher text.	2022

1.3 Research Problem

The problem statements of this research are listed below :

- Stream data from the IoT devices that transfer to the cloud through wireless media. Protecting these data is mandatory and critical needs to be addressed.
- The IoT devices have resource constrained in computing capacity, memory capacity, battery power source, time, etc. so IoT devices needed resource efficient cryptographic algorithms.
- Traditionally, symmetric encryption suffered from one enormous shortcoming – it was necessary for either the sender or the recipient to create a key and then send it to the other party. While the key was in transit, it could be stolen or copied by a third party, who would then be able to decrypt any

ciphertext encrypted with that key. Key exchange and distribution is an extra problem in the IoT environment

- Real-time applications like healthcare must have low latency as a critical requirement. The delay in sending patient data is unacceptable, as it affects the timely treatment and care for patients .

To address this problem, we need to build a model that resolves security issues and satisfies acceptable latency in the transfer data, a proposed system based on authentication, and encryption to provide acceptable latency in the computation power of IoT sensors and extract knowledge discovery.

1.4 Thesis Contribution

The main contributions of this thesis focus on finding a security solution to the problem of data encryption in a resource-constrained environment such as the Internet of Things. The system is based on choosing the most suitable lightweight algorithms and integrating certain parts of the algorithms to form a strong hybrid algorithm with high encryption specifications. In this case, the problem of developing or improving the algorithm appears, as it sometimes focuses on achieving one goal, as it is difficult to improve to achieve more than one goal. Security may be compromised by hacking when optimizing, as well as when reducing algorithm operations and reducing the number of rounds that may expose data to attacks. For this reason, the trade-off is always between security, speed, and response time and they should all be taken into account.

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) A robust authentication technique between the IoT devices and Cloud Computing based on HMAC and RSA public and private generated keys

for first session. This technique does not need a third part for key management and it is scalable.

3) Implementing secure session key management to enhance the confidentiality security of IoT sensors, it used ECDH key agreement protocol for encrypting\ decrypting data.

4) A robust system for IoT environment to Improve lightweight encryption/decryption time, throughput, and increasing security using Lightweight Cryptographic technology.

1.5 Aims of the work

The main objective of protecting data using the Hybrid Lightweight Cryptography algorithm.

1) Providing save data transfer between IoT devices and cloud computing, by design a proactive lightweight approach efficiently integrated with cloud computing to increase security.

2) Employ Light-weight algorithms that are appropriate for contexts with limited resources. These include encryption and decryption techniques that are quickly responsive, more energy and storage efficient than standard algorithms, and supported by improved crypto engines

3) Creating a strong key that is hard to break in the IoT and cloud Node by using an elliptic curve The Diffie-Hellman key exchange protocol.

4) Reduce latency between IoT devices and cloud computing by using the Hybrid Lightweight Cryptography algorithm.

1.6 Thesis Outline

After the first chapter presents a general introduction to the topic, the thesis is organized as follows:

- **Chapter Two**: "Theoretical Background", provides a comprehensive description of the basic concepts of the Internet of Things system, security, and video streaming in IoT. In addition to explaining the compression data, encryption methods, and the criteria used to test the encryption system.
- **Chapter Three**: "The Proposed System Design ", describes the design of the proposed system and the algorithm of the system stages.
- **Chapter Four**: "Proposed System Implementation, Results, and Discussion ", describes the evaluation result of the proposed system.
- **Chapter Five:** Shows the main conclusions of this work and it also offers suggestions for future works.

CHAPTER TWO

Theoretical Background

2.1 Introduction

This chapter presents, basic background information about the Internet of things, IoT architecture, IoT applications, and IoT Challenges, Then it focuses on the main types of lightweight cryptographic algorithms. This chapter discussed an 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, the chapter discusses the performance criteria used to evaluate the effectiveness of encryption algorithms, such as throughput, execution time, and entropy. Figure (2.1) illustrates the main steps of the research landscape.

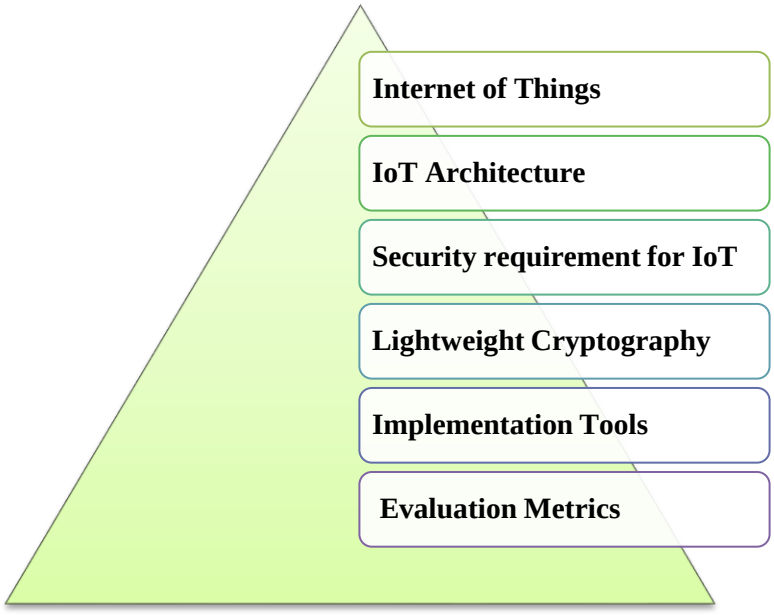


Figure 2.1:Pyramid of the Research Landscape

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)[30].

Internet of Things (IoT) refers to the interconnection of physical things, such as sensors, actuators, and other devices, through the internet. These things can include a wide variety of items, from everyday household appliances like refrigerators and TVs to industrial equipment, medical tools, and more. One of the key features of IoT is that it allows for a vast number of devices to be interconnected, making it possible to collect and analyze large amounts of data in real-time. This information can be used to improve efficiency, reduce costs, and enhance overall performance[31].

Another important aspect of IoT is the use of common web technologies, such as HTTP, JSON, and XML, which make it easy to integrate IoT devices into existing infrastructures. This means that businesses and organizations can easily incorporate IoT technology into their operations, without the need for extensive and costly infrastructure upgrades. Overall, IoT has the potential to revolutionize many industries and improve our daily lives by enabling smarter, more efficient systems and devices[31].

IoT is a collection of several technologies, including cloud services, Radio Frequency Identification (RFID), Wireless Sensor Networks (WSN), other machine-to-machine interfaces, and so on, that operate in concert as opposed to

being a single technology. Devices that aid in interacting with the physical world include sensors and actuators. To draw meaningful conclusions from the sensor data, it must be intelligently stored and analyzed. sensor refers to anything that can offer information about its current status, including a mobile phone or even a microwave. (internal state + environment)[32].

The Internet of Things(IoT) is shown in figure (2.2).

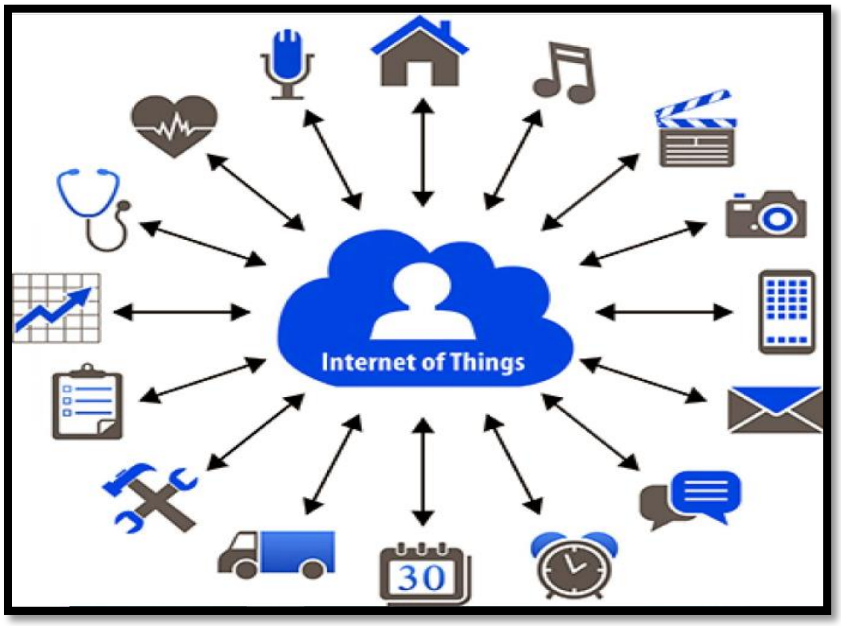


Figure 2.2:Internet of Things (IoT)-sensors/devices

John Romkey created the first "device" on the Internet in 1990—a toaster that could be operated remotely. In 1994, Steve Mann developed WearCam. It ran on a 64-processor system almost instantly. The first concise explanation of sensors and their possible applications is made in 1997 by Paul Saffo.

The term "Internet of Things" is coined in 1999 by Kevin Ashton, executive director of the AutoIDCentre at MIT, and that year they also constructed a global Radio Frequency Identifier (RFID) based item identification system. The electronics manufacturer LG unveiled its ambitions to launch a smart refrigerator in

2000, marking an important step toward the commercialization of IoT. Whether or not to replenish the food items it now has would be a decision that this refrigerator could make on its own. RFID was heavily utilized by the US Army's Savi program in 2003. Major retailer Walmart expanded the usage of RFID across all of its locations worldwide in that same year[33].

The introduction of IPv6 in 2011, which expanded the number of available IP addresses, was a critical development that enabled the growth of IoT. Today, IoT has become an essential technology in many industries, from manufacturing and transportation to healthcare and agriculture. As IoT continues to evolve, IoT has the potential to change how we work and live by opening up new avenues for innovation and enhancing the sustainability and efficiency of our infrastructure and systems[34].

2.3 IoT Architecture

The IoT architecture consists of users, sensors, a variety of physical devices, communication layers, IoT protocols, cloud services, developers, and business layers. Due to the widespread use of IoT devices, there isn't a global consensus on the IoT's organizational structure. Researchers have put out a wide range of IoT architectures, and there is also a standard and widely accepted IoT architecture. The core three layers that comprise the Internet of Things (IoT) are the perception layer, also known as the recognition layer, the network layer, and the application layer [35]. The Three Layer IoT architecture is shown in figure (2.3).

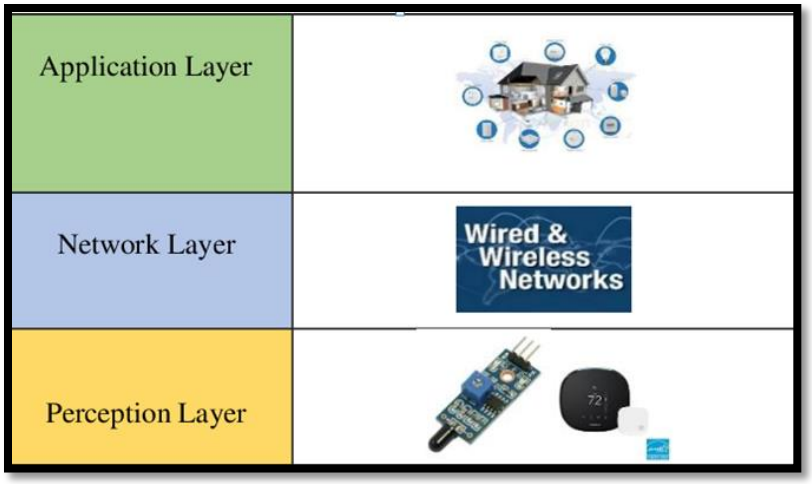


Figure 2.3: Three-Layer IoT architecture[36]

I. The perception layer

The IoT architecture's physical perception layer is sometimes referred to as the sensor layer. The main objective of the perception layer is to sense and gather information about the physical world, such as temperature, humidity, pressure, noise, and location, which is then converted into digital signals for transmission to the next layer of the IoT architecture, the network layer[37].

II. The network layer

The management and processing of the data gathered by the perception layer take place at the network layer, which is a crucial part of the Internet of Things architecture. It acts as the brain of the IoT system, responsible for managing network connectivity, data processing, and device management. The network layer is in charge of establishing connections between the various servers, smart objects, and other devices in the network so that they may communicate and exchange data. It offers a platform for data routing, storage, and analysis, enabling the processing and analysis of the data gathered from the perception layer in real-time[37].

III. The application layer

the application layer is the top layer of the IoT architecture, responsible for providing specific services and applications to users or other applications, based on the data gathered and processed by the lower layers. Also, it is in charge of making sure that the data being carried across the network is secure and private, by implementing strong encryption algorithms, access control mechanisms, and other security features to protect the data and devices from unauthorized access or attacks[37].

The three-layer architecture captures the essence of the Internet of Things, but because IoT research typically focuses on its more complex features, it is inadequate. According to some researchers, the Internet of Things framework comprises four layers. The fourth layer, which is conceived of as a support layer between the network layer and the application layer, consists of technologies including fog computing, cloud computing, and intelligent computing[38]. Figure (2.4) depicts the four layers of the Internet of Things architecture.

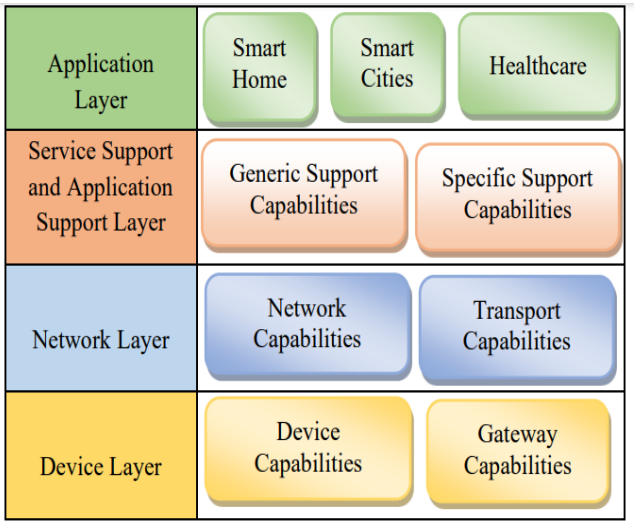


Figure 2.4: Four-Layer IoT Architecture[36]

1. The support Layer

   is in charge of collecting data from the IoT architecture's lower layers, particularly the Network Layer, and storing it in a database for subsequent processing. The advanced data analytics services offered by this layer can be utilized to evaluate the data and offer insights. Additionally, given that numerous IoT devices are used to support various IoT services, they are mostly employed for service management. Only the other smart objects engaged in the same type of service can connect and communicate with each other. This layer is in charge of overseeing these services and organizing communication among the smart objects[39].

The four-layer IoT architecture was replaced by a five-layer one due to storage and security concerns. In this five-layer architecture, a new layer called the business layer is introduced[36]. The five tiers include perception, transport, processing, application, and business layers. Similar to a three-layer design, the perception and application layers serve the same purpose. Figure (2.5) illustrates the five layers of the IoT architecture.

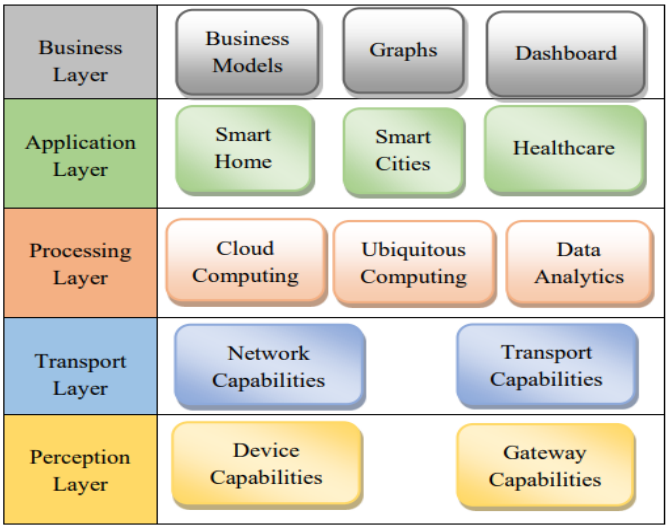


Figure 2.5: Five-Layer IoT Architecture[36]

I. The transport layer

In the IoT architecture, the Transport Layer is in charge of obtaining data from sensors in the Perception Layer and sending it to the Processing Layer or Processing Center. The transport layer may use various communication protocols and technologies such as WLAN, enterprise LAN, Bluetooth, 3G, and other wireless or wired technologies to transmit data between devices in the IoT system[37]

II. The processing layer

is additionally referred to as the middleware layer. It processes, stores, and analyzes a large amount of data that it receives from the transport layer. It can oversee and provide a variety of services to the lower tiers. It uses a range of technologies, including databases, large data processing modules, and cloud computing[40].

III. The business layer

The IoT system's Business Layer is in charge of overseeing all system operations and services. A business model, graphs, flowcharts, and other representations are created by this layer using the information from the application layer. Additionally, it creates, plans, evaluates, implements, monitors, and evaluates components of the IoT system[41].

2.4 IoT applications

The Internet of Things (IoT) has the potential to revolutionize various aspects of daily life, ranging from personal activities to commercial and industrial applications, as well as improving the environment. IoT application scenarios are found and divided into 14 categories, including supply chains, smart cities, smart homes, smart factories, lifestyle, retail, agricultural, emergency services, and health care. Other groups included user interaction, culture and tourism, environment, and energy. In the paragraphs that follow, some IoT applications are briefly outlined[42].

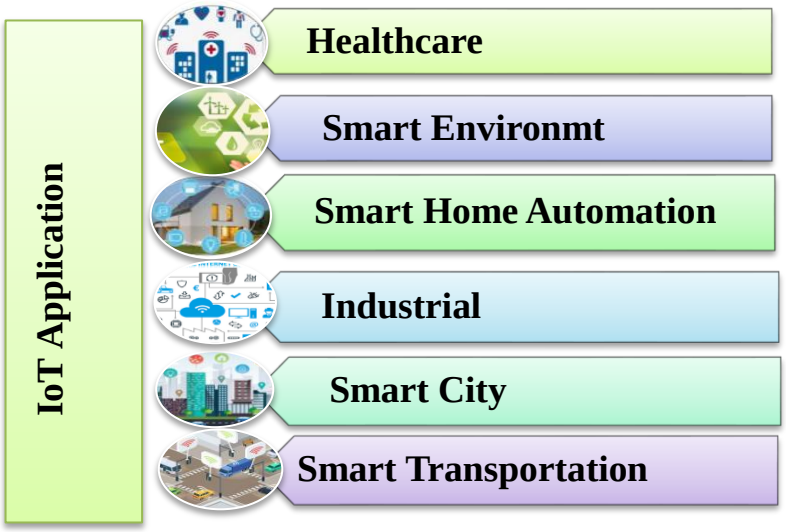


Figure 2.6: Some of the IoT Applications

A. Healthcare

The major goal of implementing IoT in healthcare is to collect and analyze real-time medical data to reduce the drawbacks of conventional medical care (i.e. medical errors). Additionally, the obtained medical data stream is stored and analyzed on cloud platforms. As a result, the information gathered about the patient's health status enables healthcare organizations to create universal healthcare applications and maximize the performance of already-existing

services and solutions, such as those for remote monitoring, nutrition, medication, medical equipment, medical facilities, or health insurance. As a result, the use of IoT in the healthcare industry helps patients locate optimal health conditions and treatment[43].

IoT appliances have proven really beneficial in the health and wellness domains. Many wearable devices are being developed, which monitor a person's health condition as showed in Figure (2.7)[44].

Health applications make independent living possible for the elderly and patients with serious health conditions. Currently, IoT sensors are being used to continuously monitor and record their health conditions and transmit warnings in case any abnormal indicators are found. If there is a minor problem, the IoT application itself may suggest a prescription to the patient[44].

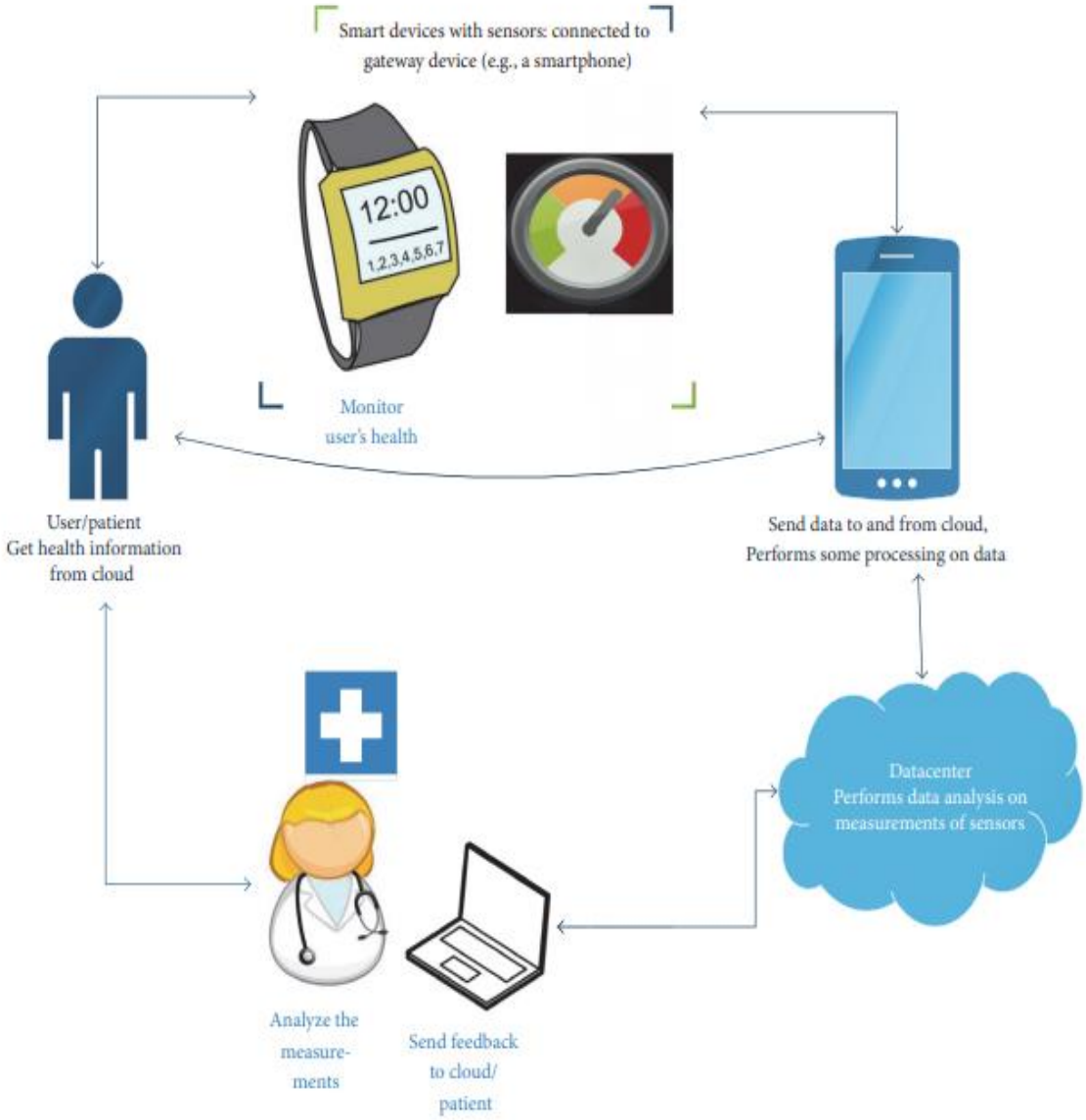


Figure 2.7: Block diagram of IoT Healthcare System[44].

B. Smart Environment

The smart environment is an important domain of IoT applications that focuses on using technology to monitor and manage environmental factors that affect human and animal life. such as forest fire warnings, high-altitude snow level monitoring, landslide prevention, early earthquake detection, pollution monitoring, etc[45].

C. Smart Home Automation

home automation is a popular and widely used IoT application. Home automation systems use connected devices and sensors to automate various tasks within the home and make it more efficient, convenient, and secure[45].

D. Industrial

The Industrial Internet of Things (IIoT) is an important domain of IoT applications that focuses on using technology to optimize industrial processes and increase efficiency. by connecting the networking of sensors, instruments, and other devices with computers for use in industrial applications including production and energy[46].

E. smart cities

IoT-based smart city applications can improve the quality of life for citizens, making their daily routine more convenient and efficient, while also providing valuable health monitoring, emergency assistance, monitoring parking availability, assessing air quality, and even sending notifications when garbage bins are full[47].

F. Smart Transportation

IoT (Internet of Things) plays a critical role in Smart Transportation, also known as Intelligent Transportation System (ITS). IoT-enabled devices such as sensors, cameras, and GPS trackers can be used to collect real-time data on traffic flow, road conditions, and vehicle location. The data collected by these devices can then be transmitted to a central platform where it is processed, analyzed, and integrated using advanced algorithms and machine learning techniques. The platform can then provide insights and recommendations to drivers, traffic management centers, and other stakeholders, enabling them to make informed decisions in real-time. the use of IoT in Smart Transportation can help to

improve the safety, efficiency, and sustainability of transportation systems, making them more responsive to the needs of drivers and the community[48].

2.5 IoT challenges

The Internet of Things has the potential to drastically change many facets of contemporary civilization. When billions of objects and devices connect to the internet, information converges, creating challenges from a wide range of IoT ramifications. Because of the intricacy of the IoT, numerous problems must be fixed before the system's full potential can be achieved[49].

The following section discusses some of the challenges faced by the Internet of Things, and Figure (2.8) summarizes the main challenges of the IoT system.

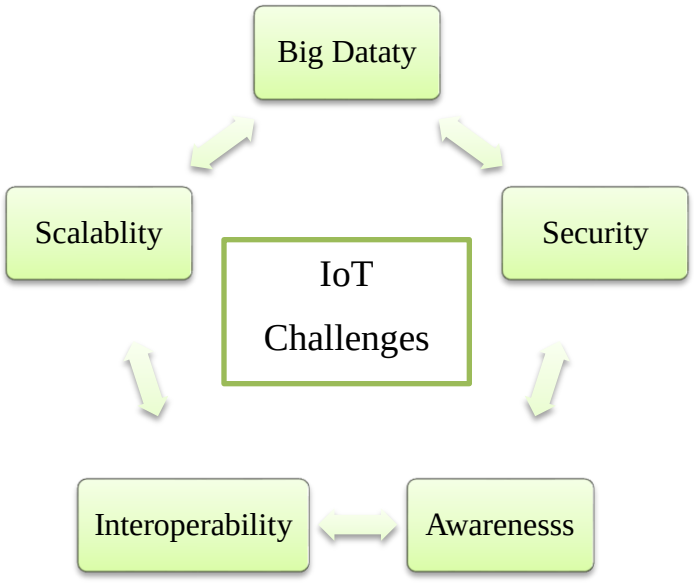


Figure 2.8: The main challenges of the IoT system

A. Big Data

Big Data is the massive amounts of data that are simultaneously produced by the sensors of different devices. IoT devices produce a variety of data kinds because they generate a lot of data. These vast amounts of data require some sort of technology to collect, clean, and analyze them. This makes it challenging to deploy an IoT system with Big Data[43].

B. Security

IoT security is among the most important issues and difficult problems because of numerous threats, cyber-attacks, hazards, and vulnerabilities. Therefore, Security should be a top priority while developing IoT system architectures, protocols, and other techniques. IoT security breaches are the only thing stopping users from adopting and utilizing the technology. Consequently, security features should be strategically positioned to facilitate widespread IoT deployment[50].

C. 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[51].

D. Interoperability

interoperability is a primary challenge in the development of IoT systems. With a vast array of devices and sensors available from various manufacturers, each with different communication protocols and technologies, ensuring interoperability can be a significant challenge. To achieve interoperability in an IoT system, it is important to establish a common set of communication protocols and standards that all devices can adhere to. This allows devices from different manufacturers to communicate and share data seamlessly[52].

E. Scalability

scalability is a critical challenge for the IoT, as the concept of the IoT is much broader than the traditional internet of computers. The IoT encompasses a vast array of devices and sensors that operate in an open environment, which presents unique scalability challenges. achieving scalability is an important consideration in designing IoT systems, as the number of devices and sensors connected to the network can grow exponentially, potentially leading to issues with network congestion, latency, and security. To support the vast number of devices and sensors that are part of the IoT, so It's crucial to create systems that can function effectively in both small- and large-scale environments[53].

The main factor that this thesis focuses on is security. Section (2.6) provides the Security Requirements for IoT and section (2.7) provides security challenges.

2.6 Security Requirements for IoT

Security is a crucial consideration when it comes to IoT devices, which are connected to the internet and can potentially be vulnerable to cyber-attacks. so 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 (2.9) [54].

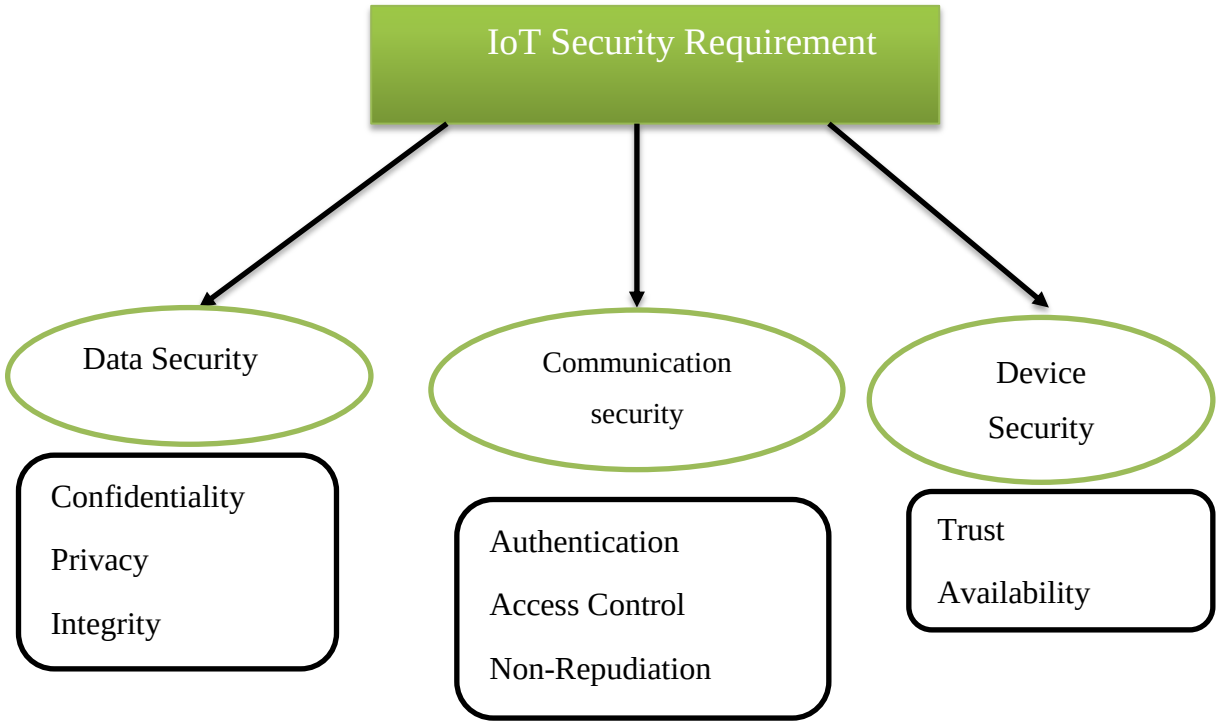


Figure 2.9: The Security Requirement for Internet of Things

2.6.1 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[54].

2.6.2 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 [54].

2.6.3 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 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[54].

2.6.4 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 the authentication process are by applying multifactor methods such as (password, PIN, One Time Password, SMS text, and biometrics) [55].

2.6.5 Authorization

The authorization has become a major concern in the IoT system as the number of items that are linked to the network continues to rise. 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 [56]. Figure (2.10) showed the difference between the authentication and authorization processes.

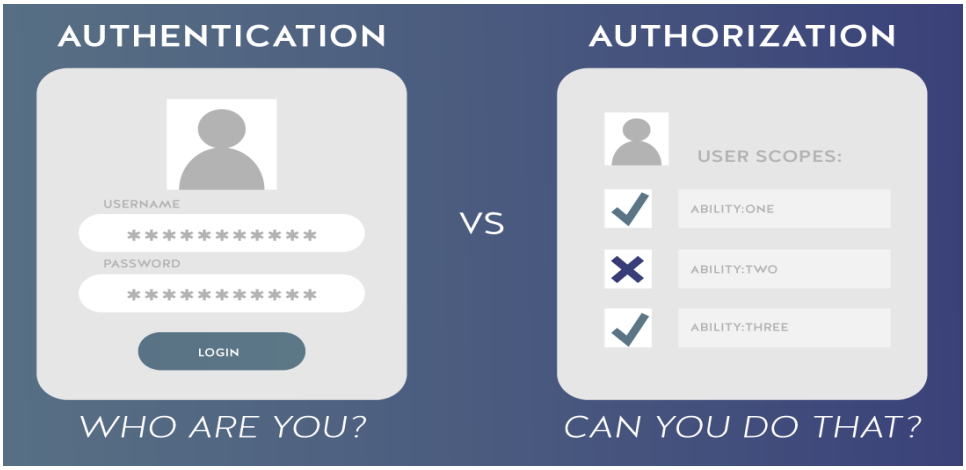


Figure 2.10: The difference between the authentication and authorization process

2.6.6 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 guarantees to a receiving node that it will verify that the data it has received matches the source of the data and promises to a source node that it will share its data [57].

2.7 Security challenges in IoT Architecture

Among the key components of the Internet of things is security. By their very nature, IoT devices bring the offline world online and frequently capture extremely sensitive data. Large amounts of data are produced by connected devices; these data must be sent securely and kept safe from cybercriminals. the central processing and control unit, which could be considered the "brain" of an IoT system, is a potential target for cybercriminals, who can use it to seize control. To prevent such a problem, it makes it logical to record, examine, and preserve all of the commands that control applications send to objects while also keeping an eye on user behavior. Also, it is feasible to spot suspicious behavior patterns, save samples of them, and compare them to the logs produced by an IoT system to stop prospective intrusions and decrease their effects on an IoT system[58].

This section discusses the many security vulnerabilities that could be present in IoT applications for these three layers. Figure (2.11) provides a summary of the main IoT layer threats.

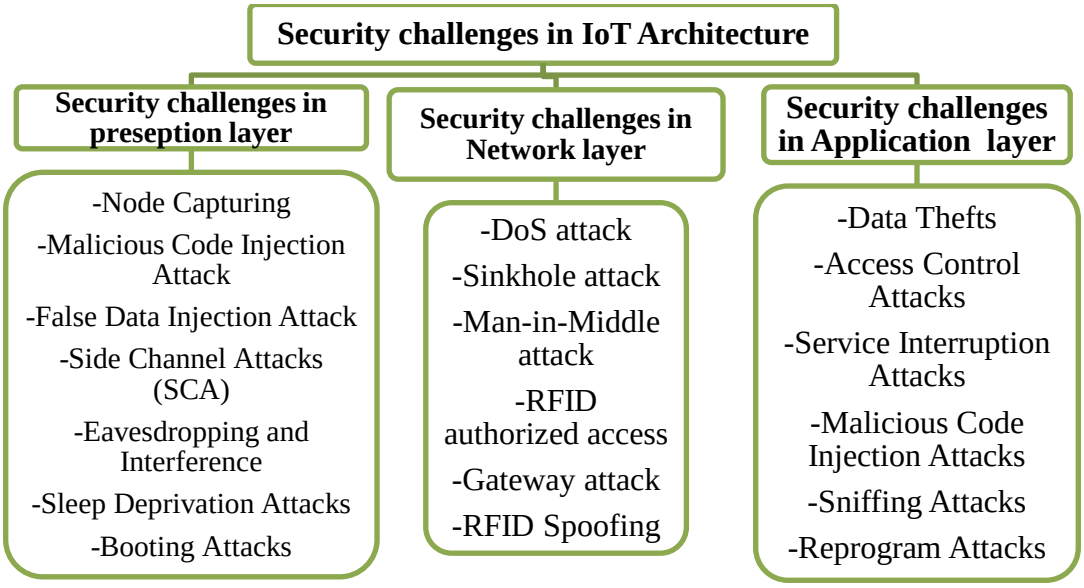


Figure 2.11: Threats on IoT layers

2.7.1 perceptron layer Attacks and Vulnerabilities:

The Perception layer, which mostly works with physical IoT sensors and actuators, is crucial to the Internet of Things security process. Sensors are devices that collect data from the physical world, such as temperature, humidity, pressure, light, sound, motion, and many other forms of data. They can be used to monitor and measure various physical phenomena, such as environmental conditions, human activity, and industrial processes. Actuators, on the other hand, are devices that receive information from sensors and perform a specified action in the physical world. For example, actuators can control the temperature of a room by turning on or off a heating or cooling system[59].

Overall, sensors and actuators play a crucial role in IoT systems by collecting and using data to make decisions and take actions in the physical world.

This layer is exposed to many security threats, some of which can be explained in this section.

A. Node Capturing:

For Internet of Things applications, numerous low-power nodes, including sensors and actuators, are required. These nodes are susceptible to a variety of attacks from the enemy. With the Internet of Things system, attackers can try to seize control of the node or swap it out for a rogue node. The attacker has control over the new node even though it seems to be a component of the system. The security of the entire IoT application may be compromised as a result[59].

B. Malicious Code Injection Attack:

Malicious code might be injected into the node's memory by the attacker to carry out the attack. IoT nodes frequently upgrade their software or firmware over the air, making it possible for hackers to add malicious code. By utilizing such malicious malware, the attackers may attempt to gain access to the whole IoT infrastructure or force the nodes to carry out some undesirable tasks [59].

C. False Data Injection Attack:

The attacker can introduce false data into the IoT system using the node they just took control of. This could lead to inaccurate results and IoT application malfunction. This technique could be used by the attacker to launch a DDoS assault.[59].

D. Side-Channel Attacks (SCA):

Private information may be leaked through a variety of side-channel attacks in addition to direct assaults on the nodes. Enemies can access critical data due to processor microarchitectures, electromagnetic radiation, and power consumption. The side channels may be attacked via electromagnetic, laser, timing, or power consumption techniques. Modern processors take care of a lot of protections while developing cryptography modules to stop these side-channel attacks.[59].

E. Eavesdropping and Interference:

IoT applications usually include a large number of nodes positioned outside. Hence, eavesdroppers have access to these IoT applications. Attackers may intercept and seize the data at several stages, such as during data transmission or authentication[59].

F. Sleep Deprivation Attacks:

In this kind of attack, attackers try to completely drain the battery of IoT edge devices with little power. Due to a dead battery, causes a denial of service from the nodes in the IoT application. This can be accomplished by intentionally raising the power consumption of the edge devices or by using malicious malware to make those devices run in an infinite loop[59].

G. Booting Attacks:

As they are booting up, the edge devices are exposed to several attacks. This is because the built-in security procedures are not functioning at the time. Attackers may attempt to exploit this weakness to attack the devices of the restarting node. Due to their frequent low power consumption and potential for sleep-wake cycles, edge devices must have a safe startup process [59].

2.7.2 Network Layer Attacks and Vulnerabilities:

The network layer sends the data acquired from the sensing layer to the processing unit to be processed[60]. But this layer faces many security threats. We will explain part of these security threats.

A. DoS attack:

A denial of service attack prevents servers from providing services to users. A DoS attack entirely halts the flow of data between devices[60].

B. Sinkhole attack:

In addition to destroying data security, this attack delivers malformed packets rather than the intended recipients[60].

C. Man-in-Middle attack:

The attacker can remain undetected within the network and only use a specific protocol to speak with the Internet of Things. The protocol used to

send data between the two sensor nodes allows for the gathering of private information[60].

D. RFID-authorized access:

Anyone can access RFID-certified access tags since RFID systems lack security authentication. This implies that these tags are simple to alter and update[60].

E. Gateway attack:

The routing attack is a component of this attack that stops data from the Internet from being routed to sensors and nodes. Additionally, it functions to break the link between the sensors and the Internet infrastructure[60].

F. RFID Spoofing:

By focusing on the RFID signal, the data imprinted on the RFID card is acquired. As a result, the attacker can transfer his data and get total access to the IoT system through the original identity [60].

2.7.3 Application Layer Attacks and Vulnerabilities:

The application layer works directly with and for end users. Smart grids, smart cities, smart homes, and smart meters are examples of IoT applications that are present on this layer. Two different security flaws, data theft, and privacy problems, are exclusive to this layer and are not present in other layers. The security flaws in this layer are also specific to different applications[61].

The primary security difficulties that the application layer must overcome are covered in the discussion that follows:-

A. Data Thefts:

Several delicate pieces of data are used by IoT applications. IoT applications move a lot of data, making them considerably more vulnerable to assaults than data that is static. Customers will be hesitant to register their personal information on IoT applications if there is a risk of data theft. Data encryption, data isolation, user and network authentication, privacy management, and other protocols are used to safeguard IoT applications from data theft[61].

B. Access Control Attacks:

Access control is a tool for controlling access so that only authorized people and processes have access to the data or account. A key vulnerability to IoT systems is access control attacks since, once access is compromised, the entire IoT application is susceptible to attacks [61].

C. Service Interruption Attacks:

Sometimes these attacks are referred to as DDoS or illegal interruption attacks. These assaults on IoT applications have happened often. These assaults intentionally overwhelm servers or networks, preventing legitimate consumers from using IoT applications[61].

D. Malicious Code Injection Attacks:

Attackers frequently choose the simplest or easiest way to enter a system or network. That would be a hacker's first choice of access point if the system is susceptible to malicious applications and misdirection as a result of inadequate code inspections. Normally, attackers insert harmful scripts into a reliable website via XSS (cross-site scripting). In the event of a successful XSS attack, an IoT account could be taken over and the IoT system could stop working [61].

E. Sniffing Attacks:

Network traffic in IoT applications may be monitored by attackers using sniffer software. The attacker might be able to get sensitive user data if there are insufficient security measures in place to stop it [61].

F. Reprogram Attacks:

Attackers may attempt to remotely reprogram the object if the IoT object programming process is insecure. Thus, there is a possibility of IoT network hijacking[61].

2.8 Cloud Computing

Cloud computing is a strategy for providing IT resources and services online. It gives consumers the option to acquire and use computer resources on-demand and as needed, such as processing power, storage, and applications, without having to purchase and maintain their IT infrastructure. By leveraging virtualization technology, cloud service providers can create multiple virtual instances of computing resources, such as virtual machines, and allocate them to users on-demand. As a result, clients may quickly scale up or down their resource usage in response to changes in demand or business requirements without having to make substantial upfront investments in new hardware or software[62].

Furthermore, cloud computing is service-oriented, which means that users can choose the specific services they need and pay for them on a usage basis, rather than having to purchase and manage entire software applications or hardware systems. This makes cloud computing a cost-effective and efficient solution for businesses and organizations of all sizes, as they can easily access the IT resources they need, when they need them, without having to worry about maintenance, upgrades, and security [63]. The basic cloud environment is shown in figure (2.12).

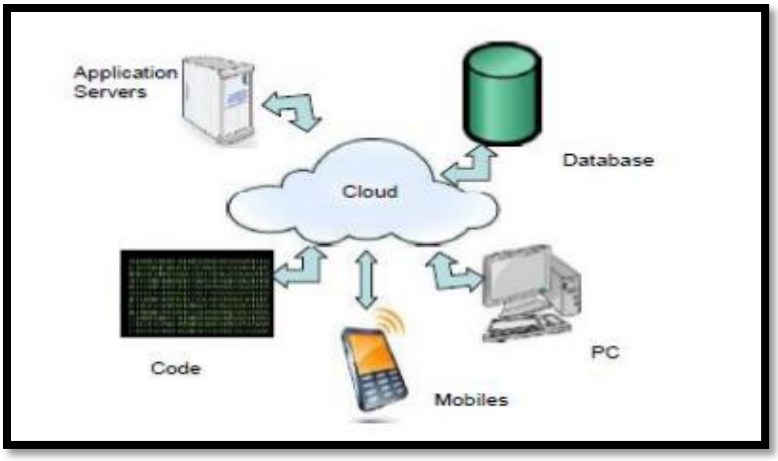


Figure 2.12: A Cloud Computing Environment[63]

2.8.1 Cloud Computing characteristic

The four main characteristics of cloud computing can be listed:

A. On-demand self-service:

On-demand One of the fundamental elements of cloud computing is self-service, which enables customers to provision computer resources—such as servers, storage, and applications—on-demand without requiring direct communication with the cloud service provider. This means that users can quickly and easily deploy and configure the resources they need, without having to wait for approval or assistance from the provider. It also allows for a more flexible and agile approach to resource allocation, as users can scale their usage up or down as needed, based on their changing requirements[64].

B. Resource pooling:

resource pooling is another important characteristic of cloud computing, It refers to the ability to combine and allocate computing resources, such as hardware, software, processing power, and network bandwidth, in a shared pool that can be dynamically assigned and reassigned to multiple users or applications based on demand. This pooling of resources enables cloud

providers to deliver computing services at a lower cost, as they can achieve economies of scale by serving multiple users with the same infrastructure [64].

C. Rapid elasticity and scalability

This means that cloud service providers can quickly and easily allocate additional resources, such as processing speed, storage capacity, and memory, to suit the changing needs of their users. As a result, consumers can scale their usage up or down as needed, depending on factors such as increased workload, seasonal demand, or unexpected spikes in traffic. Rapid elasticity and scalability are enabled through the use of automation and orchestration tools, which allow for the dynamic provisioning and allocation of resources. This allows cloud providers to deliver computing services that are both flexible and cost-effective, as users only pay for the resources they use, and can swiftly and easily increase or decrease their usage as necessary[64].

D. Measured Service:

Cloud service providers analyze each user's consumption and more effectively distribute resources by using metering tools to measure and monitor the use of computing resources including processing power, storage, and bandwidth. customers may now just pay for the resources they utilize, Instead of needing to buy and maintain their infrastructure. Measured service also allows for better cost management, as users can monitor their usage and adjust their resource allocation accordingly. This may assist in lowering the cost of further resource provisioning, as users can identify areas where they may be overprovisioning or underutilizing resources and make adjustments accordingly [64].

The cloud is an essential component of the Internet of Things (IoT) ecosystem. The increasing number of connected devices and the exponential growth in data they generate have made it increasingly difficult for organizations to store, process, and analyze all of this data using traditional on-premises IT infrastructure. Cloud computing provides a scalable and cost-effective solution for dealing with this challenge, it provides a centralized platform for managing IoT devices and their data, enabling real-time monitoring and control of connected devices, and facilitating data-driven decision-making. Cloud-based services can also provide advanced analytics and machine learning capabilities, allowing for predictive maintenance, automated decision-making, and other intelligent applications. Moreover, the cloud allows for the seamless integration of IoT devices with other systems and applications, providing a unified view of the entire ecosystem. This enables businesses to streamline their operations and enhance their customer experiences, while also improving their overall efficiency and productivity. As IoT continues to grow and evolve, the cloud's role is likely to become even more critical. with the development of new technologies such as edge computing and 5G networks, the cloud will need to adapt and evolve to support these new use cases and requirements[65].

Overall, cloud computing is an essential enabler of the Internet of Things, providing the necessary infrastructure, services, and capabilities to support the massive scale and complexity of this rapidly growing ecosystem. The integration between IoT devices and Cloud is shown in figure (2.13) [66].

Figure 2.13: IoT and Cloud Computing Integration[66]

2.8.2 Cloud Deployment Models

Clouds can be classified in terms of who owns and manages the cloud; a common distinction is Public Clouds, Private Clouds, Hybrid Clouds:

A. public cloud

A public cloud, or external cloud, is the most common form of cloud computing, in which services are made available to the general public in a pay-as-you-go manner. Customers – individual users or enterprises – access these services over the internet from a third-party provider who may share computing resources with many customers. The public cloud model is widely accepted and adopted by many enterprises because the leading public cloud vendors as Amazon, Microsoft and Google, have equipped their infrastructure with a vast amount of data centers, enabling users to freely scale and shrink their rented resources with low cost and little management burden. Security and data governance are the main concern with this approach[67].

B. Private Cloud

A Private Cloud, or internal cloud, is used when the cloud infrastructure, proprietary network or data center, is operated solely for a business or organization, and serves customers within the business fire-wall. Most of the private clouds are large company or government departments who prefer to keep their data in a more controlled and secure environment[67].

C. Hybrid Cloud

A composition of the two types (private and public) is called a Hybrid Cloud, where a private cloud is able to maintain high services availability by scaling up their system with externally provisioned resources from a public cloud when there are rapid workload fluctuations or hardware failures. In the Hybrid cloud, an enterprise can keep their critical data and applications within their firewall, while hosting the less critical ones on a public cloud[67].

2.9 Data Security of the Internet of Things

The transfer of large amounts of data over wireless networks to public cloud computing platforms in the context of the Internet of Things (IoT) presents several security challenges. IoT devices are vulnerable to security threats such as malicious attacks and data theft because they often transfer data over public networks and use wireless channels to do so. Information security in IoT is therefore a critical concern and requires sophisticated technology to secure the system [66]. The best solution for data security is encryption.

2.9.1 IoT Encryption

Encryption is a crucial technique for securing information and data in various formats, including files, photos, and documents. By transforming plain text using a mathematical formula into an incomprehensible format, encryption can protect sensitive information from unauthorized access and modification[68].

Encryption algorithms can be resource-intensive and may require significant processing power and memory to operate effectively. This can be a challenge in the context of the Internet of Things (IoT), where devices often have limited resources and may not be able to handle complex encryption algorithms. To address this issue, lightweight cryptography solutions have been developed that can provide robust security while using fewer resources. These solutions typically use simpler algorithms that are optimized for use in resource-constrained environments. By using lightweight cryptography, IoT devices can benefit from strong security measures without sacrificing performance or efficiency[69].

Overall, encryption is an essential tool for securing IoT devices and networks. However, it is important to choose encryption algorithms that are optimized for use in the context of the IoT to ensure that they are both effective and efficient[69].

Cryptography primitives, which are the basic building blocks of cryptographic systems, can be broadly categorized into two main categories: symmetric key cryptography and asymmetric key cryptography[70]. These categories are shown in figure (2.14).

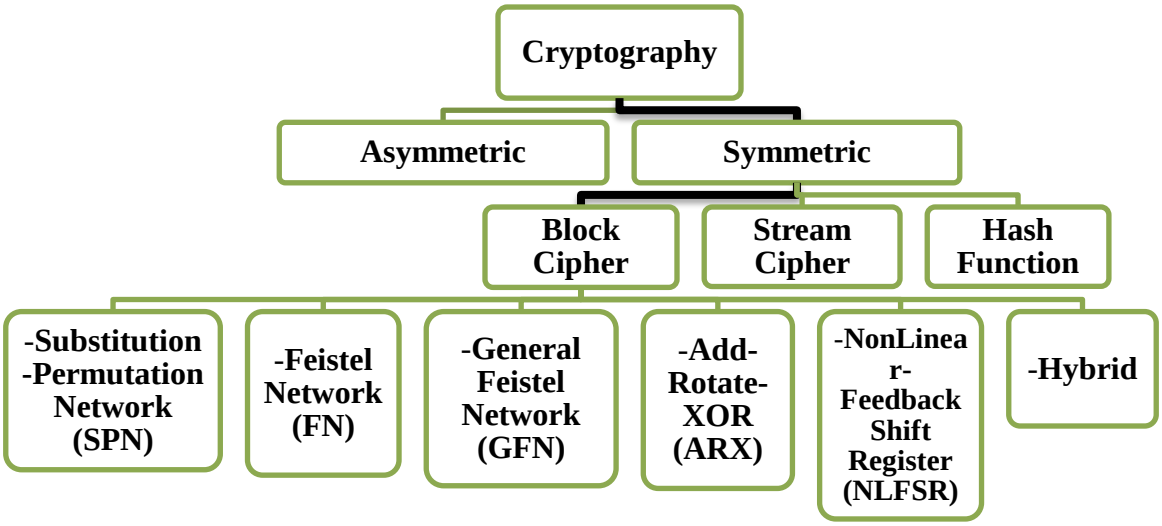


Figure 2.14: classification of lightweight encryption algorithm[70]

2.9.1.1 Lightweight symmetric cryptographic algorithms

Symmetric key cryptography is also referred to as secret key or shared key cryptography. This type of cryptography uses a single key for both encryption and decryption, which is shared between the sender and the receiver through a secure communication channel. Symmetric key cryptography is often preferred for IoT applications due to its speed and efficiency. The operations involved in symmetric key cryptography, such as XOR and permutations, are typically faster than those used in asymmetric key cryptography, making it a more suitable option for devices with limited resources. In addition to its speed and efficiency, symmetric key cryptography is also often used for its simplicity and ease of implementation. However, a potential drawback of symmetric key cryptography is the need for secure key distribution between the sender and the receiver, as any compromise of the shared key could compromise the security of the entire system[71].

Overall, symmetric key cryptography is a useful tool for securing IoT devices and networks, particularly when speed and efficiency are a priority. symmetric Key Cryptography is shown in figure (2.15) [70].

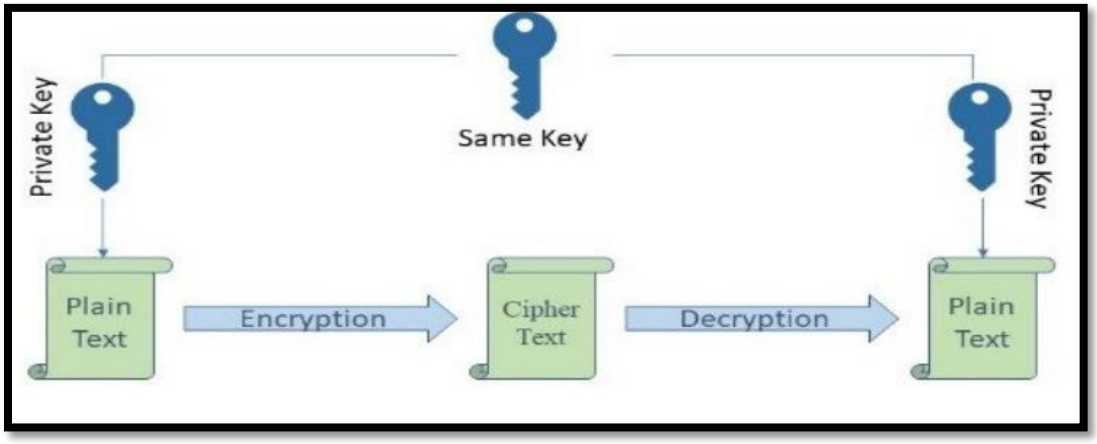


Figure 2.15: Symmetric Key Cryptography[70]

There are various Symmetric encryption algorithms, including the following.

A. Block Ciphers Algorithms
B. Stream Ciphers Algorithms
C. Hash Functions Algorithms

A. Block Ciphers Algorithms

Block ciphers are a type of symmetric-key cryptography that operate on fixed-size blocks of data. They convert plaintext into ciphertext and vice versa using a secret key. Claude Shannon, a pioneer in cryptography, proposed two fundamental properties of a secure cipher: confusion and diffusion. Confusion is achieved through the use of substitution operations such as S-boxes, which make it difficult to determine the relationship between the plaintext and the ciphertext without knowing the key. Diffusion, on the other hand, spreads the influence of each plaintext bit or block over many ciphertext bits or blocks, making it difficult to identify patterns in the

plaintext or ciphertext. Together, confusion and diffusion serve to make the ciphertext as different as possible from the plaintext while maintaining the security of the key[70].

B. Stream Ciphers Algorithms

Symmetric stream ciphers are a type of encryption algorithm that operates on a stream of data, typically one byte or bit at a time, using a related key stream to produce cipher text. To encrypt a plain text stream, the stream cipher generates a key stream using a secret key, and then XORs each byte of the key stream with each byte of the plain text to produce the cipher text. To decrypt the cipher text, the same process is used in reverse, with each byte of the key stream XORed with each byte of the cipher text to recover the plain text. Stream ciphers are often used in resource-constrained devices and IoT contexts, such as low-power RFID tags, because they are typically faster, smaller, and use less power than block ciphers. However, they are generally considered less secure than block ciphers because they do not provide the same level of diffusion and confusion as block ciphers[72].

C. Hash Functions Algorithms

A hash function, also known as a hash value or digest, is a mathematical procedure that accepts input data of any size and outputs a fixed-size result. The output is often a fixed-length bit string with a predetermined amount of bits. As hash functions are one-way by design, it is computationally impractical to obtain the input data from the hash value. They are frequently utilized for digital signatures, authentication, and data integrity checks[73].

As a result of the foregoing, block cipher is chosen over stream cipher in IoT devices with limited resources since block ciphers are primarily made to provide security, integrity, and authenticity. This is because block ciphers use a combination of confusion and diffusion properties to transform the plaintext into ciphertext.

Confusion refers to Linking the plaintext and the ciphertext as complicated as possible, typically through the use of substitution (S-boxes). S-boxes are used to substitute each input bit with a corresponding output bit, based on a predefined table. This process helps to make it difficult for an attacker to deduce the relationship between the plaintext and the ciphertext[70].

Diffusion refers to the process of spreading the influence of a single plaintext input bit over many parts of the ciphertext output. This is usually achieved through the use of operations that mix the input bits in a complex way, such as permutation (P-boxes) and substitution (S-boxes). The goal of diffusion is to ensure that a small change in the input (i.e., flipping a single bit) results in a large change in the output, making it difficult for an attacker to determine the relationship between the input and output. [70].

Our focus will be on block ciphers, namely symmetric lightweight block ciphers. It employs one of the structures listed below:

i. Substitution-Permutation network (SPN):
   prepares the input for the next round by modifying it using a series of replacement boxes and permutation tables[70].
ii. A Feistel network (FN):
   A Feistel network is a type of symmetric key cipher that uses a series of rounds to encrypt plaintext into ciphertext. The basic structure of a Feistel

network involves splitting the plaintext into two equal parts, and applying a function to one half (diffusion) while leaving the other half unchanged. The two halves are then swapped, and the process is repeated for several rounds until the final ciphertext is produced [70].

iii. Generalized Feistel Network (GFN):

The Generalized Feistel Network (GFN) is a modification of the conventional Feistel network that allows for the input block to be split into multiple smaller blocks, which can be of different sizes. The Feistel functions are then applied to each pair of these smaller blocks in a round-robin fashion.[70].

iv. Add-Rotate-XOR (ARX):

ARX (Addition, Rotation, XOR) is a class of cryptographic algorithms that use only three basic operations: addition, rotation (also known as bitwise shift), and exclusive OR (XOR) operation. These operations are considered to be simple and efficient, which makes them useful in situations where resources such as processing power, memory, or storage are limited[70].

v. Nonlinear Feedback Shift Register (NLFSR) :

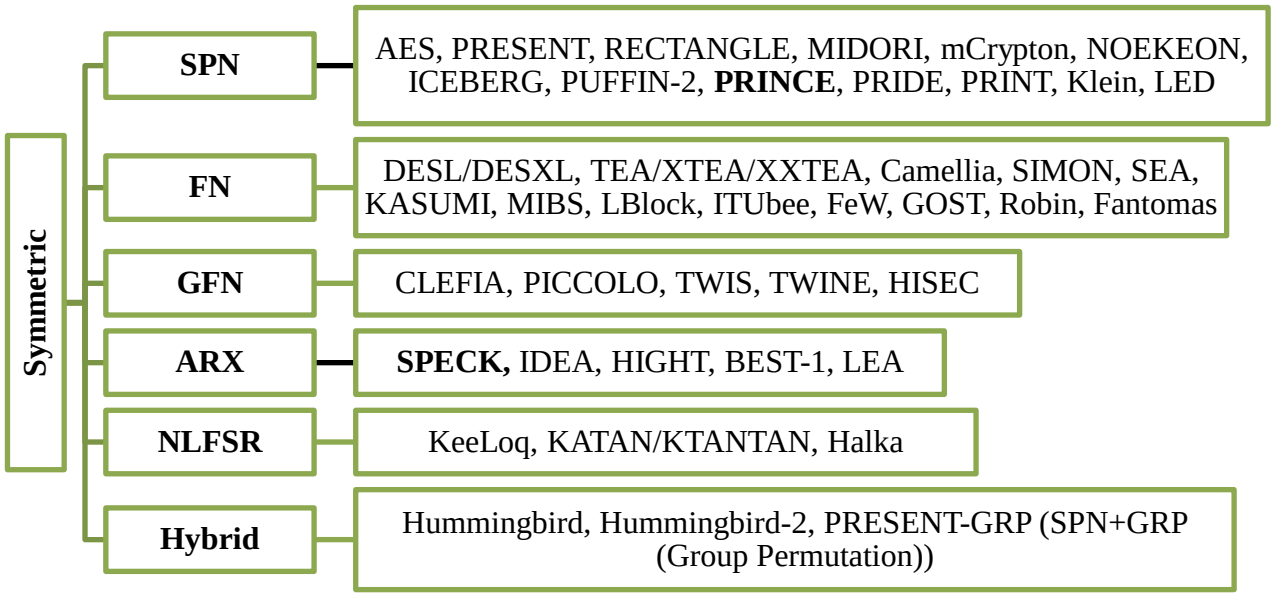
Is a kind of cryptographic algorithm that may be applied to block and stream ciphers. Like other types of shift registers, an NLFSR generates a sequence of bits by shifting the register's contents to the right and then computing the value of the new bit based on the current state of the register[70].

vi. Hybrid:

A cipher may mix any three categories (SPN, FN, GFN, ARX, NLFSR), or it may even combine block and stream attributes to raise many specialized qualities, depending on the requirements of the application (such as throughput, energy, GE, etc.)[70].

Table (2.1) shows the classification structure of lightweight encryption algorithms.

Table 2.1: Structure of lightweight Block cipher algorithms[70]



2.10 The used Encryption algorithms

2.10.1 Speck Algorithm

which is a family of thin, symmetrical block blades designed by the National Security Agency (NSA) in 2013. The algorithm was designed to provide secure cryptography in situations with limited resources, where standard cryptography methods would not function effectively. The Speak method has been extensively tested and has been proven to be secure. It is also more flexible compared to other lightweight ciphers. This algorithm relies on basic operations like AND, Rotation, and XOR, which can be executed even on devices with limited resources, making it adaptable and effective in the future[74].

The main goal of the Speak method is to provide security in constrained devices. It has a strong reputation for having quick execution times, security, and using simple operations[75]. Table (2.2) showed the main properties of the SPECK algorithm.

Table 2.2:The SPECK algorithm properties

Designers	Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, Louis Wingers NSA
First published	2013
Key sizes	64, 72, 96, 128, 144, 192, or 256 bits
Block sizes	32, 48, 64, 96, or 128 bits
Structure	ARX

Speck algorithm, which offers various block and key sizes to accommodate the user's hardware resources. This is in contrast to other lightweight ciphers that operate with fixed-size keys and blocks. The algorithm supports 10 different sizes depending on block size (2n) and key size (mn) resulting in 20 different sizes 10 for encryption and 10 for decryption, this flexibility allows the algorithm to be applied in both small embedded devices and sophisticated computer systems. Additionally, the Speck scheme includes a counter that is built into the algorithm. This improves the efficiency of the algorithm and guards against rotation and slide assaults. This means that the Speck algorithm is not only flexible but also provides strong security features that make it a reliable choice for various applications[76]. The Parameters of the SPECK algorithm are shown in Table (2.3).

Table 2.3:The SPECK algorithm properties[77]

block size $2n$	key size mn	word size n	key words m	rot α	rot β	rounds T
32	**64**	16	4	7	2	22
48	**72**	24	3	8	3	22
	96		4			23
64	**96**	32	3	8	3	26
	128		4			27
96	**96**	48	2	8	3	28
	144		3			29
128	**128**	64	2	8	3	32
	192		3			33
	256		4			34

Each time the Speck round function is used, the following actions are performed on n-bit words:

- Bitwise XOR,⊕,
- Addition modulo 2^n ,+,and
- Left and right circular shifts, S^j and S^{-j}, respectively, by j bits

Where j is the number of bits being shifted. For both the equations, the rotation amount of α and β are 2 and 7 respectively, for the block size is 32bits. And for all other block sizes, these two rotation amounts are 3 and 8, respectively.

The encryption functions of the speck algorithm is equation (2.1) denoted by R as follows [77]:

$$\boldsymbol{R(x,y)} = ((\boldsymbol{S^{-\alpha}}\ \text{x} + \text{y}) \oplus\ \text{k},\ \boldsymbol{S^{\beta}}\ \text{y} \oplus (\boldsymbol{S^{-\alpha}}\ \text{x} + \text{y}) \oplus \text{k}) \quad \ldots\ldots\ldots (\mathbf{2-1})$$

For decryption, the inverse of the round function utilizes modular subtraction rather than modular addition and is equation (2-2) [77]

$$R^{-1}(x,y) = (S^{\alpha}((x \oplus k) - S^{-\beta}(x \oplus y)), S^{-\beta}(x \oplus y)) \quad \ldots\ldots\ldots (2-2)$$

The Encryption process of Speck:

There are different encryption methods used in the Speck algorithm, including ECB, CTR, CBC, PCBC, CFB, and OFB. These are different modes of operation that are used to provide different levels of security and efficiency, depending on the application. The Speck algorithm can be used to encrypt any type of data, including text, images, and other digital content. The various cipher modes used in the algorithm are designed to obscure the patterns in the encrypted data by using the output sequence from the first cipher block or other globally deterministic variables in the second cipher block. This helps to provide additional security by making it harder for a hacker to read the plaintext content from the encrypted data[78]. The various encryption modes that the Speck algorithm supports are displayed in table (2.4).

Table 2.4: Modes of encryption[78]

Mode		Formulas	Ciphertext
Electronic codebook	(ECB)	Y_i – F(PlainText$_i$, Key)	Y_i
Cipher block chaining	(CBC)	Y_i = PlainText$_i$ XOR Ciphertext$_{i-1}$	F(Y, Key); Ciphertext$_0$ = IV
Propagating CBC	(PCBC)	Y_i = PlainText$_i$ XOR (Ciphertext$_{i-1}$ XOR PlainText$_{i-1}$)	F(Y, Key); Ciphertext$_0$ = IV
Cipher feedback	(CFB)	Y_i = Ciphertext$_{i-1}$	Plaintext XOR F(Y, Key); Ciphertext$_0$ = IV
Output feedback	(OFB)	Y_i = F(Y_{i-1}, Key); Y_0 = F(IV, Key)	Plaintext XOR Y_i
Counter	(CTR)	Y_i = F(IV + $g(i)$, Key); IV = token()	Plaintext XOR Y_i

Figure (2.16) and the equations represent the circular function of the Speck and contain two parts: the block on the left x (Xi), and the block on the right y (Yi). k represents one of set of round Keys (K0,K1,…,KT-1) where T is number of rounds. $S^{-\alpha}$ and $S^{-\beta}$ Denotes a left and right circular offset by α or β bits, the products the i_{th} round are Xi + 1 and Yi + 1 [76].

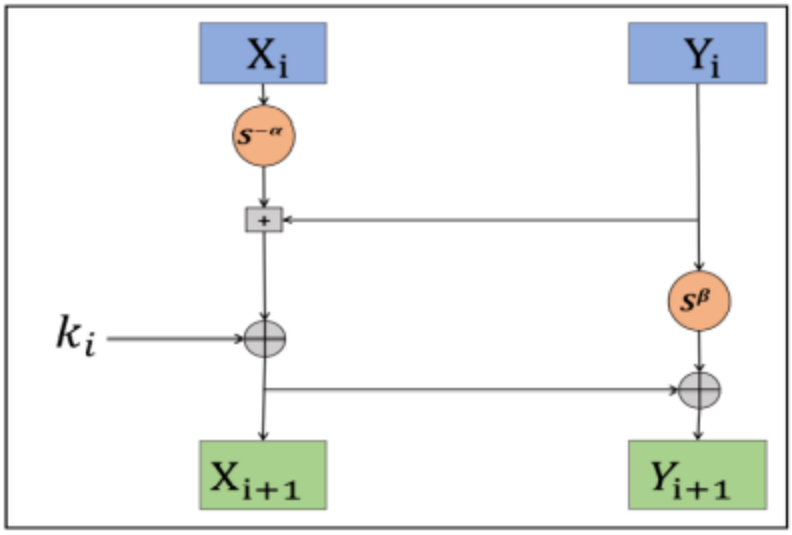


Figure 2.16:Structure of SPECK round function[76].

The original k key must be used to generate round keys for Speck's key generation feature. Using the key generation function of the round function given above, the round key is generated for each round[77]. The key generation of the Speck key generation is shown in figure (2.17).

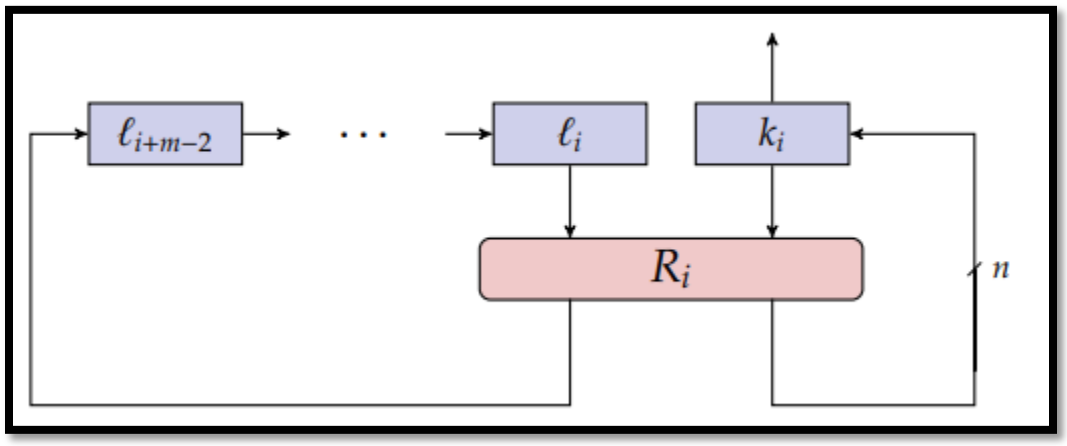


Figure 2.17: Structure of SPECK key generation function[76] [79]

K is a key for a SPECK 2n block cipher. We can write K= $(\boldsymbol{l_{m-2}}, \dots, \boldsymbol{l_0}, \boldsymbol{k_0})$, where K_i is the i^{th} round key, for 0 < i < T:

$\boldsymbol{l_i, k_0} \in \boldsymbol{GF(2)^n}$, for a value of m in {2,3,4}. Sequences $\boldsymbol{k_i}$ and $\boldsymbol{l_i}$ are defined by

$$\boldsymbol{l_{i+m-1}} = (\boldsymbol{k_i} + \boldsymbol{S^{-\alpha}}\, \boldsymbol{l_i}) \oplus \mathbf{i} \quad \ldots\ldots (\mathbf{2-3}) \quad \text{and}$$

$$\boldsymbol{k_{i+1}} = \boldsymbol{S^{\beta}}\, \boldsymbol{k_i} \oplus \boldsymbol{l_{i+m-1}} \quad \ldots\ldots (\mathbf{2-4})$$

The figure illustrates the speck algorithm's encryption procedure (2.18) Additionally, Algorithm (2.1) displays the SPECK encryption algorithm pseudo code, and Algorithm (2.2) displays the scheduling of the SPECK encryption keys[79].

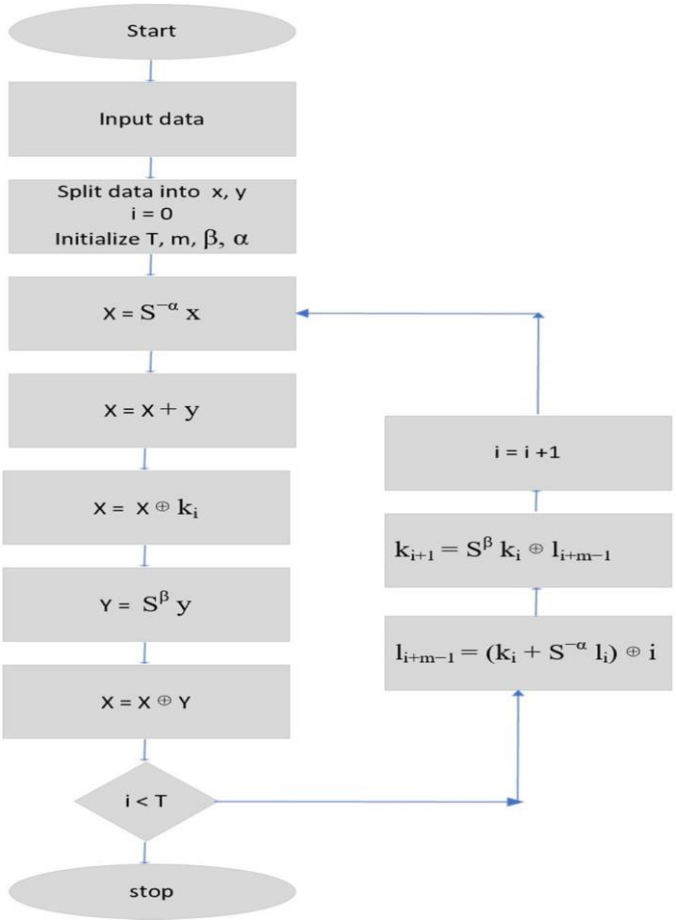


Figure 2.18: The speck encryption flow[79].

Algorithm 2.1: pseudo code of the SPECK encryption

Input: sensor data
Output: Encrypted data

Begin:
Step 1: Generate a key with a 128-bit key length
Step 2: input data with size 64-bit.
Step 3: input values for α=8, β=3.
Step 4: Split the 64-bit input data block into two 32-bit words: X, Y.
Step 5: For each round, from 0 to 25, apply the equation of encryption with the SPECK algorithm.

$\boldsymbol{R}(\boldsymbol{x},\boldsymbol{y}) = ((\boldsymbol{S}^{-\boldsymbol{\alpha}}$ x + y) ⊕ k, $\boldsymbol{S}^{\boldsymbol{\beta}}$ y ⊕ ($\boldsymbol{S}^{-\boldsymbol{\alpha}}$ x + y) ⊕ k)

can be described the equation of encryption as the following:

Step 5.1: Apply the shifting α to X(left side) input data.
Step 5.2: Apply the Anding operation between the output of shifting X with Y(right side), the result of this operation can be called X1
Step 5.3: XOR the X1 bit state with the round key Ki, the result of this operation will be X2
Step 5.4: Apply the shifting β to Y input data.
Step 5.5: Apply the XOR operation between the output of shifting Y with X2

Step 6: The 64-bit state is the output
End

Algorithm 2.2: pseudo code of the SPECK algorithm key generation

Input: Encrypted data
Output: Sequence of Key

Initialize m;
Generate $\boldsymbol{l_{m-2}}, \dots, \boldsymbol{l_0}, \boldsymbol{k_0}$
For I = 0 to T-2
$\boldsymbol{l_{i+m-1}}$ = ($\boldsymbol{k_i} + \boldsymbol{S^{-\alpha}}\ \boldsymbol{l_i}$) ⊕ **i**
$\boldsymbol{k_{i+1}} = \boldsymbol{S^{\beta}}\ \boldsymbol{k_i} \oplus \boldsymbol{l_{i+m-1}}$
End for loop

2.10.2 PRINCE algorithm

PRINCE is a lightweight block cipher designed for limited resource environments, such as IoT devices. It is developed by researchers at the University of Bordeaux in France and is resistant to various types of attacks, including differential and linear cryptanalysis. It is referred to as the first lightweight block cipher to take the latency characteristic into account. PRINCE utilizes a 128-bit key and a 64-bit Substitution-Permutation Network (SPN) with 12 rounds [80]. Table (2.5) showed the main properties of the PRINCE algorithm.

Table 2.5:The PRINCE properties

Designers	Technical University of Denmark, INRIA, Ruhr University Bochum, and NXP Semiconductors
First published	2012
Key sizes	128 bits
Block sizes	64 bit
Structure	SPN

A. Key Schedule

The master key k is divided into two 64-bit keys, K_0 and K_1, such that K= ($K_0 \| K_1$), where || denotes the concatenation. k is extended to 192 bits according to the following equation, where L represents a simple linear transformation.

$$K= (K_0 \| K_1) \rightarrow (K_0 \| K'_0 \| K_1) = (K_0 \| L(K_0) \| K_1);\ L(K_0) = (K_0 >>> 1) \oplus (K_0 >> 63) \quad \ldots.(2.5)$$

The third generated subkey K'_0, together with K_0 are used as whitening keys. The key K_1 of 64 bits is used as an internal key for the core of the block

cipher (12 rounds) referred to as the PRINCE core[81]. The general structure of the PRINCE block cipher is depicted in Figure (2.19).

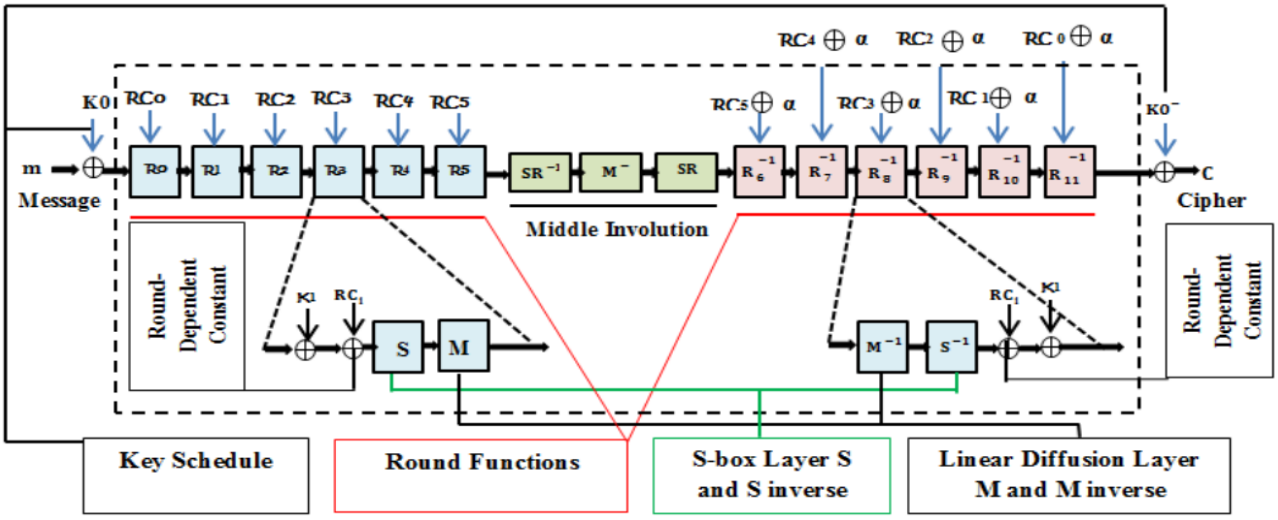


Figure 2.19: The PRINCE Core Cipher components[82]

The 12 rounds of the cipher core each contain a fixed key, 16 parallel 4x4 S-boxes, a linear diffusion, and a round-dependent constant addition[80].

The substitution layer S applies a 4-bit S-box S to every nibble of the internal state. The S-box is shown in Table (2.4) in hexadecimal notation

Table 2.6:S-box of PRINCE block cipher[81]

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S[x]	B	F	3	2	A	C	9	1	6	7	8	0	E	5	D	4

The linear layer M includes, as described in equation (2.6), the multiplication M' by a 64 x 64 matrix and a Shift Rows (SR) transformation.

$$M = SR * M' \quad \ldots\ldots\ldots (2.6)$$

Equation (2.7) is explained how the multiplication with the matrix M' is performed, where $M_0$and M_1 are two different matrixes of 16x16 that have the

following property: after the multiplication, each output bit depends only on three bits of the input value. ($X_0||X_1||X_2||X_3$) represents the 64-bit state.

$$M'.(X_0||X_1||X_2||X_3) = M_0.(X_0)||M_1.(X_1)||M_1.(X_2)||M_0.(X_3)\ldots..(2.7)$$

SR permutes the 16 nibbles according to Table (2.7).

Table 2.7:SR permutation of PRINCE block cipher

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
↓															
0	5	10	15	4	9	14	3	8	13	2	7	12	1	6	11

Key addition represents the bitwise addition between the current state $b_{63}... b_0$ and the 64-bit round key $k_1 = k_{63}... k_0$. The operation performed is defined in equation (2.8).

$$b_j \rightarrow b_j \oplus k_j \ ,0 \leq j \leq 63 \ldots\ldots (2.8)$$

Round constant addition can be expressed as in equation (2.9). It is the XOR operation between the 64-bit state $b_{63}... b_0$ with the 64-bit round constant $RC_i = v_{63}^i \ldots. v_0^i \ ,0 \leq \mathrm{i} \leq 11$.

$$b_j \rightarrow b_j \oplus v_j^i \ ,0 \leq j \leq 63 \ldots\ldots (2.9)$$

B. PRINCE core encryption and decryption processes

The encryption process includes three types of rounds: forward rounds, middle rounds, and backward rounds. The equations for each type of round are described below.

Forward :$R_i(x) = M\big(S(x)\big) \oplus k_1 \oplus RC_i \quad , i = 0 \dots 4$ ….(2.10)

Middle 2 rounds: :$R_i(x) = S^{-1}(M'.S(x))$ ……(2.11)

Backward: R_i^{-1} (x) = $S^{-1}\big(M^{-1}(x \oplus k_1 \oplus RC_i)\big) , i = 7 \dots 11$ …(2.12)

The decryption can be performed by reusing the encryption process with a different key. This functionality is successful due to the α-reflection property of the PRINCE block cipher. This means that the round constants satisfy equation (2.13), where α is the constant hexadecimal value 0xc0ac29b7c97c50dd.

$$RC_i \oplus RC_{11-i} = \alpha \quad ,0 \le i \le 11 \text{……..….}(2.13)$$

Overall, PRINCE is designed to be effective in both hardware and software implementation, making it a suitable choice for IoT devices and other resource-constrained environments. Additionally, it provides strong resistance to various types of attacks, making it a secure choice for encryption.

The pseudo-code of the PRINCE algorithm is shown in *Algorithm (2.3).*

Algorithm 2.3:Pseudo code of the PRINCE algorithm

Input: sensor data
Output: Encrypted data

Begin:
Step 1: Generate a key with a 128-bit key length, and split the key into two 64-bit halves K0 and K1.
Step 2: Apply the XOR operation between the input data block and K0.
Step 3: Split the 64-bit input data block into four 16-bit words: X0, X1, X2, and X3.
Step 4: For each round, I from 0 to 11, perform the following operations:
  Step 4.1: Apply the S-box operation to each 16-bit word Xi.
  Step 4.2: Apply the linear diffusion layer to the 64-bit state.
  Step 4.3: XOR the 64-bit state with the round key Ki.
Step 5: XOR the 64-bit state with K1.
Step 6: The 64-bit state is the output
End

2.10.3 Key Agreement

Establishing a shared secret between two parties, sometimes referred to as a key agreement or key exchange, is one of the most crucial components of cryptography. A key agreement is a procedure or process used to create a secret key that is accessible to two parties. A key exchange system's goal is to securely exchange cryptographic keys between two parties, preventing anyone else from obtaining the keys[83].

There are many different key exchange techniques, but some of the most common ones are Diffie-Hellman key exchange (DHE) and elliptic-curve (ECDH), FHMQV (fully hashed Menezes-Qu-Vanstone), RSA-OAEP, RSA-KEM, PSK (pre-shared key), SRP (secure remote password protocol), and ECMQV (elliptic-curve (quantum-safe key agreement)[84].

A- The Elliptic Curve Diffie-Hellman (ECDH):

The elliptic curve Diffie Hellman (ECDH) is a Diffie-Hellman protocol variation that uses elliptic-curve cryptography. It is based on the elliptic curve discrete logarithm problem (ECDLP), not the discrete logarithm problem(DLP), in contrast to the generic Diffie Hellman (DH). With the use of the anonymous key agreement protocol ECDH, two parties A and B can construct a shared secret key through an unprotected channel. A public-private key pair with an elliptic curve is available to each party. A symmetric-key cipher can then be used to encrypt subsequent communications using the key or a key that is derivable from it [84][85].

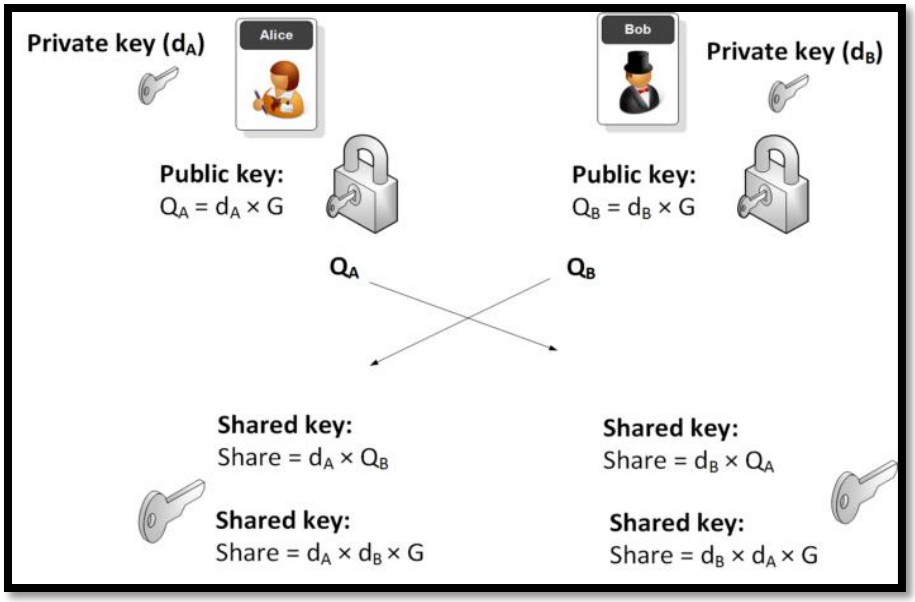


Figure 2.20: EllipticCurveDiffieHellman (ECDH) work

2.10.4 Message Authentication codes

Authenticated encryption is a technique used to ensure the confidentiality, integrity, and authenticity of encrypted data. It combines encryption and message authentication to provide a secure communication channel. The message authentication code (MAC) is one of the techniques used in authenticated encryption to ensure the authenticity and integrity of the message. To use MAC, a secret key is generated, and the message is signed with this key. The signature is then sent along with the message to the receiver. The receiver verifies the signature using the same key and checks if the message has been tampered with or not. If the message has not been tampered with, the receiver accepts it, and if it has been tampered with, the receiver rejects it[86]. Sequence and timing can also be used to achieve authentication. This involves verifying the sequence and timing of messages to ensure they are authentic[87].

The Message Authentication Code (MAC) is a technique used in authenticated encryption to ensure the authenticity and integrity of the message. It uses a key generation algorithm, a signature method, and a verification algorithm to ensure that the message has not been tampered with during transmission[87].

1. Hash Function

This mathematical function or procedure uses a small number as the big data's hash table index to break it down into smaller chunks. A fixed-size encrypted text known as the hash value is produced from random data as the input [88]. The original content can then be replaced with this ciphertext, which can be used to confirm the user's identity. The hash function is different from data compression in that the latter can decompress and restore data while the former cannot because it is a one-way function. Digital signatures can be encrypted and decrypted using the hash function. The hash value and the digital signature are provided to the receiving device after being converted using the hash function. The receiving device uses the same received hash function to generate a hash value. It then compares the two functions; if they match, the message is from the intended recipient and error-free[89].

Two categories of hash functions exist:

- Keyless hash functions (Hash Message Authentication Code, or HMAC): With this type, a hash result is generated from a single parameter (message contents).
- Keyed hash functions: Using two parameters to generate a hash value (contents of message and key).

The contents of the message are divided into blocks of equal sizes, and in case the last block is not equal to the other blocks, padding is added to it [90]. The process of the hash function is shown in figure (2.21).

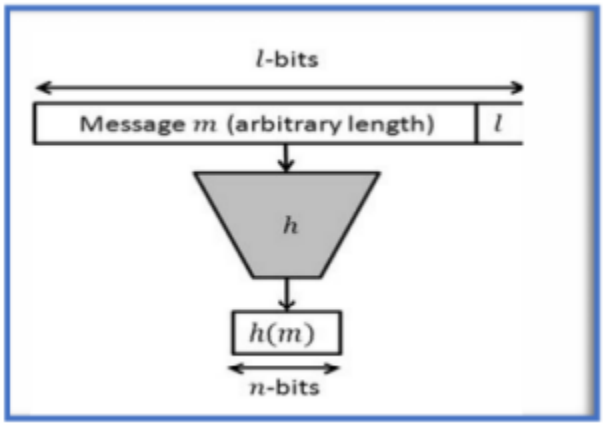


Figure 2.21: the process of generating the hash value[88]

2. Secure Hash Algorithm (SHA)

There are many hash algorithms, the most important being the Secure Hash Algorithm (SHA) used for authentication of transmitted messages between the sender and the recipient. After finding significant weaknesses in cryptanalysis, the National Institute of Standards and Technology developed SHA in 1993 and published it as a Standard for Federal Information Procedures (FIPS 180). After finding weaknesses in SHA, known as (SHA-0), the SHA hash function was developed in 1995 with a NIST version called SHA-1 known as SHA-160 and then produced in 2002. It was called NIST FIPS 180-2 It is an updated and modified version of the old version, then new versions appeared, which are SHA-512, SHA-384, and SHA-256, also known as SHA-2. This company added another version in 2002, which is SHA-224[89][90]. The Standard Hash Functions are shown in table(2.8).

Table 2.8:Standard Hash Functions[91]

Properties	Name of Algorithm				
	MD5	**RIPEMD -160**	**SHA-1**	**SHA-2 256/512**	**SHA-3 256/512**
Block Size	512 bits	512 bits	512 bits	512/1024 bits	1088/576 bits
Word Size	32 bits	32bits	32bits	32/64 bits	320/320bits
Output Size	128bits	160 bits	160 bits	256/512 bits	1600/1600bits
Rounds	18	80	80	64/80	24/24
Operations	ADD,XOR, AND,OR, NOT, SHIFT	ADD,, ROTATE, XOR,AND, OR,NOT	ADD, XOR AND, OR,NOT, ROTATE.	ADD, XOR, OR, AND SHIFT,, ROTATE	-
Construction	Merkle-Damgard	Merkle-Damgard	Merkle-Damgard	Merkle-Damgard	Sponge

A. SHA -3

In addition to being known as Keccak, it is the third iteration of the SHA algorithm. In the most recent NIST Hash Function Competition, Keccak was chosen as the winner. As no substantial SHA-2 attacks have been demonstrated, SHA-3 is not aimed to replace SHA-2. However, it was created in response to the need to discover a different and distinct cryptographic hash construct that is more resistant to attacks and has also relied on hashing growth employing a brand-new method known as "sponge." The sponge function gives flexibility to the algorithm structure by allowing the use of inputs and outputs of variable lengths. The domain extender also helps the sponge used to examine the security effects of increasing the message block size to increase efficiency. Unlike its contemporaries MD5, MD6, and other hash function architectures, Keccak does not use iterative structures. The Keccak implementation suggested for SHA-3 uses a single permutation for all security levels, which lowers the cost of implementation[92]. Hash functions and stream ciphers can both be created using the Keccak construction[88]. Keccak is a reliable, adaptable, and

effective hash algorithm a result. Keccak was created to address the demand for a more secure cryptographic hash architecture[92].

The sponge function divides the message entered into chunks of a predetermined size and is dependent on the following four factors:

f = the internal function used to process each input block.

r = the size in bits of the input blocks, called the bitrate.

C= is referred to as the capacity (complexity of the sponge construction).

Pad = the padding algorithm.

For Keccak, the default parameters are 1024 bits for c, 576 bits for r, and 1600 bits for b[92].

Figure (2.22) depicts the SHA-3 block diagram. Part B of the picture provides more information on the key algorithms and functionalities of SHA-3.

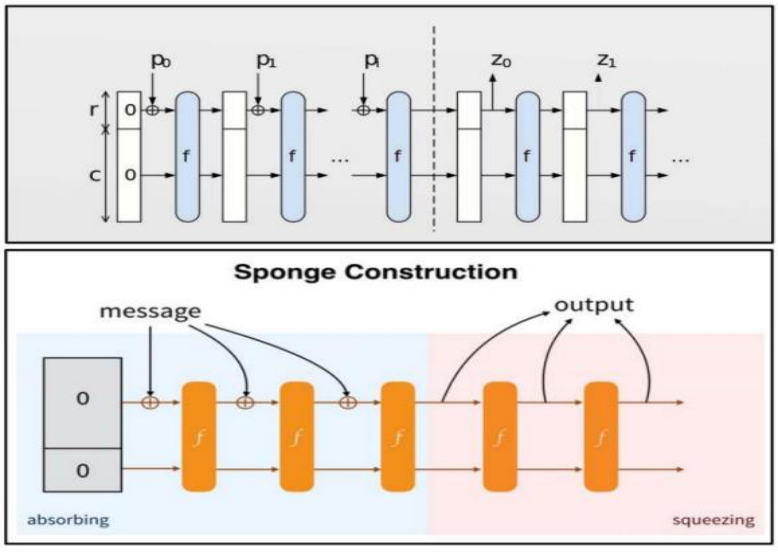


Figure 2.22: Block diagram of the SHA-3[92]

2.11 Implementation Tools

The thesis work is utilizing the Raspberry Pi (RPi) as a key building block due to its small size, affordability, high processing power, and flexibility. The RPi is a programmable computer board that is equipped with various input and output ports, as well as a network connection, making it a suitable choice for processing and controlling sensing devices. The author has used the Raspberry Pi 3 model B specifically in this work, as shown in Figure (2.23).

Figure 2.23: The Raspberry Pi (RPi) 3 model B

Besides, the thesis uses two different types of sensors to collect data. The first sensor is a temperature sensor, which is used to detect body temperature and temperature changes. The second sensor is an oximeter (SPO2) sensor, which is used to determine blood oxygen saturation. The SPO2 sensor provides results in the form of a percentage that represents the amount of hemoglobin that is saturated with oxygen in the blood. Figures (2.24) and (2.25) show the temperature and SPO2 sensors used in the study, respectively.

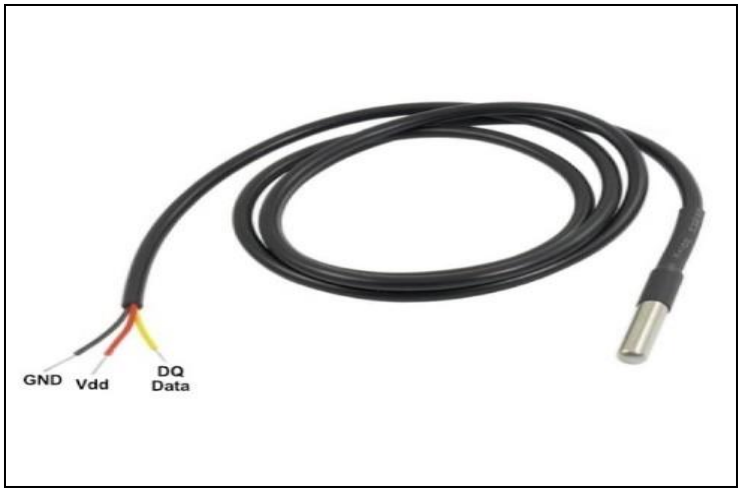


Figure 2.24: The Temperature (DS18B20 Waterproof) sensor.

Figure 2.25: The SPO2 Sensor MAX30100.

And, we use Arduino is a small, complete, and breadboard-friendly board. The Arduino board is connected to the Raspberry Pi. The Raspberry Pi reads digital data, while the data from the SPO2 sensor comes in the form of analog data. To convert the analog data into digital data, the Arduino board is used. The Arduino board is capable of converting the analog signal from the SPO2 sensor into a digital signal that can be read and processed by the Raspberry Pi. This enables the Raspberry Pi to collect and analyze data from the SPO2 sensor effectively.

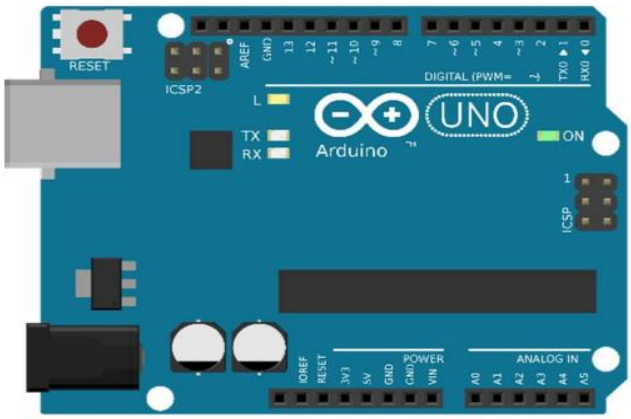


Figure 2.26: Ardunio

Python is the programming language used in this project for both the Raspberry Pi device and the cloud. Python is a commonly and versatile language that works well for many different purposes, including resource management, data analysis, and network data analysis. Python is a fantastic choice for many programming tasks because of its simplicity, readability, and ease of implementation. Its adaptability and power allow it to handle a wide range of tasks effectively.

2.12 Evaluation Metrics

The proposed cryptosystem will be tested and evaluated using some assessment metrics, including:

2.12.1 Throughput

The greatest amount of data that can be delivered across an internet connection from source to destination in a specific amount of time is measured using this unit. Equation (2-14) is utilized to calculate the throughput[93].

$$Throughput = \left(\frac{Number\ of\ send\ data}{Time}\right) \quad \ldots.(2-14)$$

2.12.2 Execution Time

It is employed in this thesis to calculate the execution time (compression and decompression processes, encryption and decryption processes, and transfer data). Equation (2-15) is utilized to calculate the execution time [94].

$$Execution\ time = End\ Time - start\ Time \quad \ldots.(2-15)$$

2.12.3 Entropy

Entropy, which is used to define and measure the unpredictable nature of data, is the most significant indicator of information randomness. Its definition states that it is the projected average scale of information from the data following encryption. Equation (2-16) is utilized to calculate the entropy [95].

$$Entropy = -\sum_{i=0}^{m} pi\ log_2(pi) \quad \ldots..(2-16)$$

Chapter Three

The Proposed Security System

CHAPTER THREE

THE PROPOSED SECURITY SYSTEM

3.1 Introduction

This chapter explains the details of the proposed Hybrid Lightweight Cryptography Algorithm (HLCA) to secure IoT data. The PRINCE and SPECK lightweight encryption algorithm was adopted to design and implement the HLCA approach. In addition, the HLCA system uses ECDH key exchange to generate a shared secret key. The main steps of the proposed HLCA system consist of the following: (1) generating and collecting sensor data; (2) the key generation phase ; (3) data encryption; (4) the Authentication phase (this stage between the sensing devices and Cloud Computing); (5) send the encrypted data and the digital signature to the cloud.

3.2 The proposed Implementation Requirements

The proposed system installation is based on the IoT devices such as a temperature sensor (DS1820 Waterproof Digital Thermal Probe 1m), and oxygen sensor (Pulse Oximeter (SPO2) Sensor MAX30100), and the management device as a (Raspberry Pi (RPi 3 model B+)), and server (Cloud Computing)

The main steps of implementation requirements in the proposed system are sorted as follows:

1. Configuration network devices (Raspberry Pi and IoT Devices)
2. Building security system (PRINCE, SPECK algorithms

and call ECDiffieHellman protocol)

3. Configuration of the Server (connecting sockets, ports, and addresses translation).

3.2.1 System Installation Requirements

The proposed system can be easily installed in hospitals, houses and can serve as a large data source to collect data. The results can be integrated with the mobile by developing an application so that it can be easily accessed at all times and in all locations. As well as the proposed system required the:

1- Computer system with Windows operating system.

2- Raspberry Pi device with RaspbianOS.

3- IoT sensor devices for temperature and SPO2 (sensor of oxygen)

3.2.2 Configure Raspberry Pi and IoT Devices

The proposed system components are Raspberry Pi (RPi), a Temperature Sensor, and a sensor of Oxygen.

- Raspberry Pi (RPi) we choose it as a key building element because of its performance: small, cheap, powerful, and fully customizable. Also, it is a programmable small computer board, The Raspberry Pi 3B+ has 40 pins, including 26 GPIO pins. The pins used for UART are GPIO 14 (TX) and GPIO 15 (RX). These pins are used to transmit and receive data, respectively. They can be used to connect the Raspberry Pi to other devices using a serial communication protocol.
- Temperature Sensor: detection of body temperature and its changes.
- A sensor of oxygen: it measures the level of oxygen

The configuration (Supply voltage, Product Size, and Output Interface) of the used devices are described in Table (3.1).

Table 3.1: Specification Details of Devices Nodes.

Device Type	**Device Model**	**Supply voltage**	**Product Size**	**Output Interface**
Raspberry Pi	B+ (ARD_000049).	DC 5V/2.5A	82mm x 56mm x 19.5mm, 50g	4 USB 2.0 ports, WiFi
Temperature Sensor	DS18B20 Waterproof Digital Thermal Probe 1m	3.0V to 5.5V	6*50mm	RJ11/RJ12,3P-2510,USB.
A sensor of oxygen	Pulse Oximeter (SPO2) Sensor MAX30100	3.3V	5.6mmx 2.8mm x 1.2mm	14-Pin Optically Enhanced System-in-Package

3.3 The proposed Methodology

The proposed system is based on a lightweight cryptographic technique called HLCA, which stands for "Hybrid Lightweight Cryptographic Approach". The HLCA algorithm uses two popular lightweight encryption algorithms PRINCE and SPECK for encrypting and decrypting data. In addition to encryption, the proposed system includes an authentication phase between the sensing devices and the Cloud. This phase allows only authenticated sensors to communicate with the Cloud. The authentication process uses the signature algorithm Hash Message Authentication Code (HMAC), which ensures the authenticity and integrity of the data transmitted between the sensors and the Cloud. To generate a shared secret key for encryption and decryption, the system uses the Elliptic-curve Diffie–Hellman (ECDH) key exchange protocol. This protocol enables two parties to establish a shared secret key over an insecure communication channel, which can then be used for the encryption and decryption of data. The proposed system works at the client-

side, which means that the sensing devices encrypt their readings efficiently and protect IoT data before transmitting it to the Cloud. This approach ensures the confidentiality and privacy of the data, even if the communication channel between the sensors and the Cloud is compromised. The general view of the proposed system is shown in Figure (3-1).

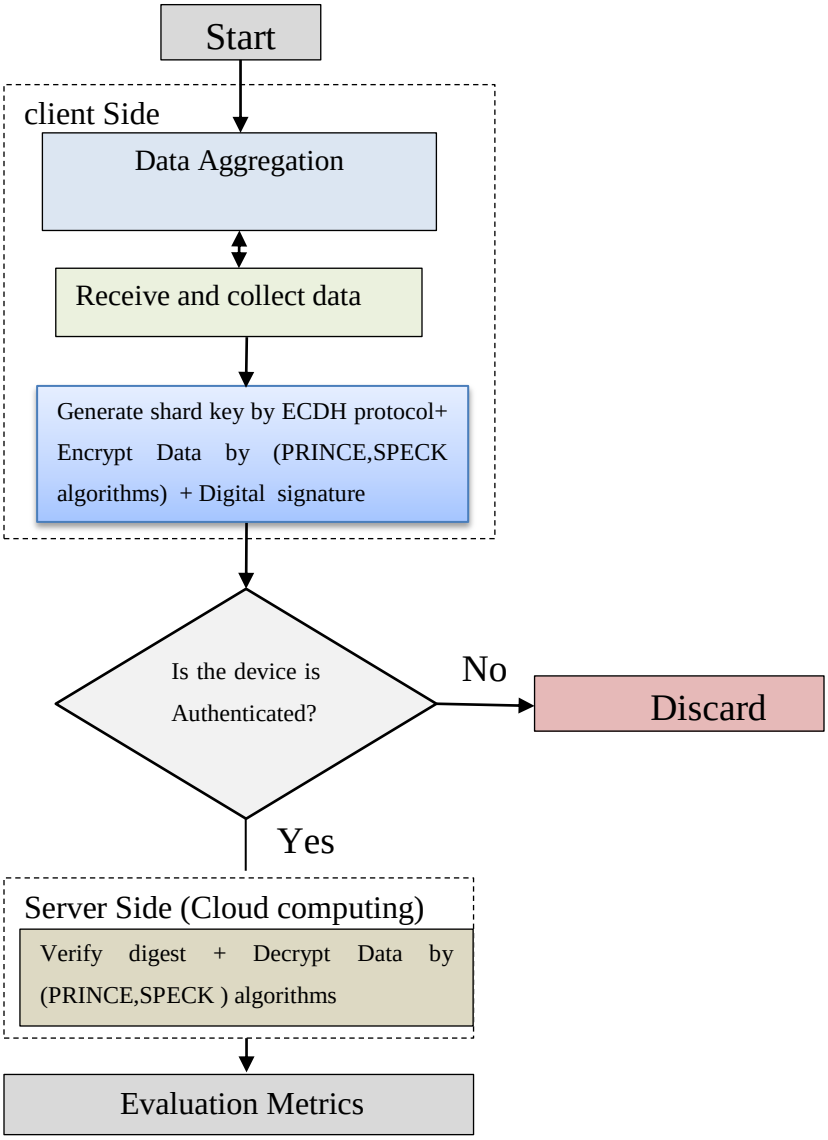


Figure 3.1: The general view of the proposed system.

The proposed system has several main steps, it will be described as follows:

3.3.1 Data Aggregation

The sensors are initialized to read a stream of data from the patient's body. Two IoT sensors, including Temperature, High-Sensitivity Pulse Oximeter Sensors, are connected to the Raspberry Pi, which receives the data from them and collects sensing information as data messages. The model of sensors used is shown in table (3.2) :

Table 3.2: The used Sensor type and Models

Sensor Type	**Sensor Model**
Temperature Sensor	DS18B20 Waterproof Digital Thermal Probe 1m
A sensor of oxygen	Pulse Oximeter (SPO2) Sensor MAX30100

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

The proposed system is based on the 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 (PRINCE, SPECK) lightweight encryption algorithms. The reason for using this protocol is based on the Elliptic Curve Cryptography (ECC) is a 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, Because the number of bits is much less.

The process of ECDH consists of the following steps: First, each client (Raspberry pi) and Cloud has the private key and an ECC elliptic curve with generator point G. Secondly, both the client and the Cloud generate the public key by multiplying the private key with G, and then exchange their public key over an insecure channel. Thirdly, then both the client and server can derive a shared secret key by multiplying the received public key with your private key.

Figure (3.2) and algorithm (3.1) showed the process of the 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.

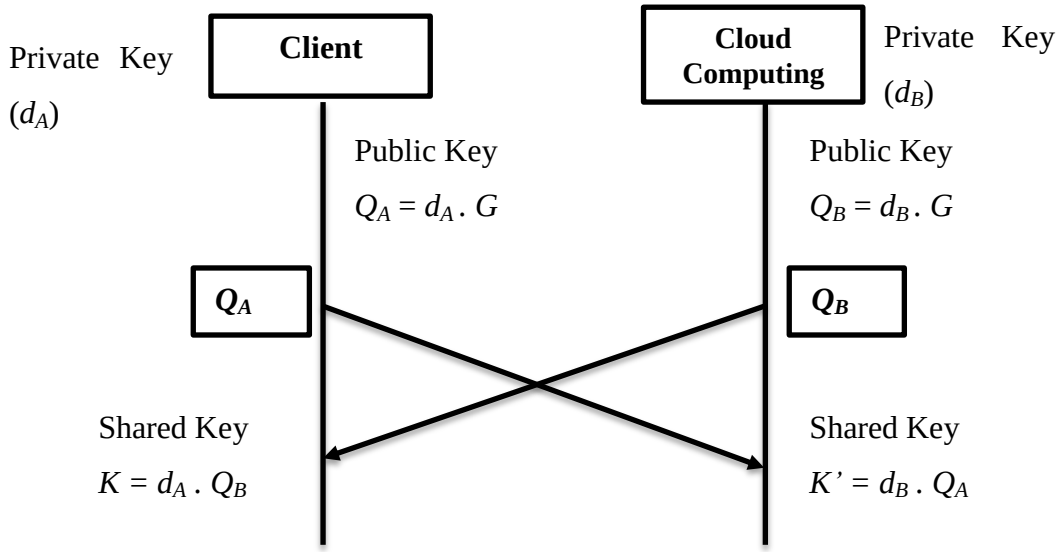


Figure 3.2:The process of Elliptic-curve Diffie–Hellman protocol.

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 $Q_A = d_A . G$ *sends* to server B
4. Server B computes $Q_B = d_B . G$ *sends* to client A
5. Client A calculate $K = d_A . Q_B = d_A (d_B . G)$
6. Server B calculates $K' = d_B . Q_A = d_B (d_A . G)$

End of algorithm

3.3.3 Encryption data using hybrid lightweight cryptography algorithms PRINCE, SPECK (HLCA)

The HLCA algorithm is a hybrid lightweight cryptography algorithm that provides a high level of security for data encryption in constrained environments such as IoT environments. By combining the PRINCE algorithm with the SPECK algorithm, the HLCA algorithm can provide a high level of randomness and security while still being efficient enough to be used on resource-constrained devices. The use of the ECDH protocol for key generation also ensures that strong and secure random numbers are used to generate the key.

The HLCA algorithm is designed with 12 rounds, using the PRINCE algorithm for 10 rounds, and the speck algorithm for two rounds (round one and twelve). The SPECK algorithm is inserted as a layer in the PRINCE round layers to increase the complexity of the encrypting results.

The suggested hybridization of the PRINCE algorithm with the SPECK algorithm increases the randomness strength of the whole PRINCE structure processing, even if its key generation is weak.

Figure (3.3) likely shows the encryption state of the data using the hybrid lightweight cryptography algorithms PRINCE and SPECK. Algorithm (3.3) is the used HLCA Encryption Algorithm.

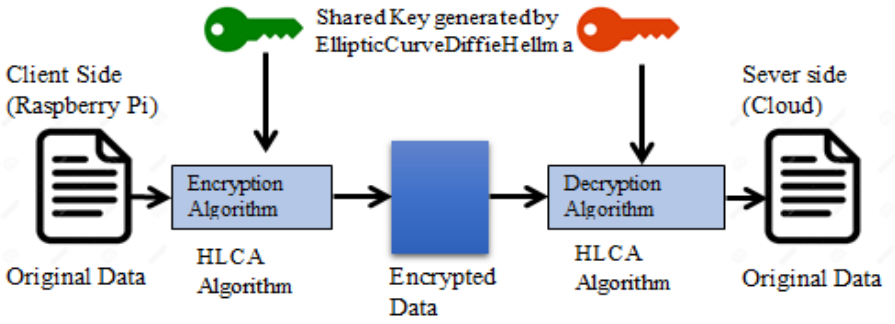


Figure 3.3: The used HLCA encryption and decryption algorithm

Algorithm 3.2: pseudo code of the HLCA

Input: Sensor Data +shared secret key

Output: Encrypted data (Ciphertext)

1. Initialize system parameters and constants for PRINCE and SPECK
2. Receive data packets from sensors (Temperature Sensor, A sensor of oxygen);
3. Generate a random 128-bit key using the ECDH protocol
4. For each block of data apply the following:
   4.1.Encrypt the block of data using the PRINCE algorithm with the initialized round key, applying 10 rounds of encryption
   4.2. Take the output of the 10 rounds of PRINCE encryption and encrypt it using the SPECK algorithm, applying 2 rounds of encryption.
5. Concatenate all encrypted blocks to obtain the encrypted data.
6. make a digital signature on the encrypted data.
7. send the encrypted data with the digital signature message to the server(cloud).

End of algorithm

3.3.4 Authentication process

This section of the implementation focuses on the authentication process between the IoT sensors (specifically, the IoT healthcare sensors for temperature and SPO2 in the Raspberry Pi) and the cloud server. The authentication process involves the exchange of an authentication digest from the sensor device to the cloud server. This digest contains two keys generated using the RSA algorithm, namely Pk (Public Key) and Prk (Private Key)

It is necessary to do the following process for authentication IoT device (A) with cloud layer (B): -

Message= temperature + SPO2 … (3.1)

Digest = SHA-256 (message) … (3.2)

SignatureA $=\{Digest\}_{PrK}$ … (3.3)

where: P_K is the public key of the IoT device in which each device includes public and private keys, and SHA256 is the hash function.

Figure 3.4 provides a schematic view of the second section, illustrating the authentication process between the IoT sensors (Temperature and SPO2) in the Raspberry Pi and the cloud server. This diagram helps to visually explain the steps involved in the authentication process and the flow of information between the sensor device and the cloud server.

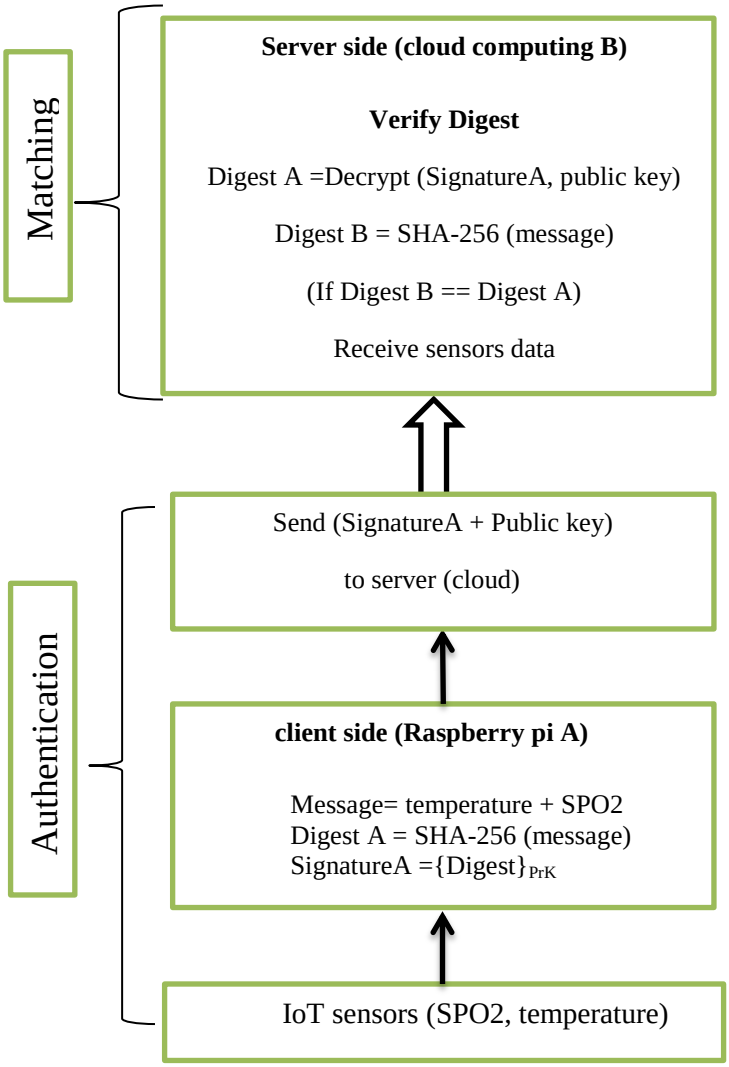


Figure 3.4:The Authentication process between IoT sensors and Cloud

The authentication process in an algorithm (3.3) is implemented in this step between the IoT devices and the cloud layer.

Algorithm 3.3: Authentication Process for IoT device (A) and cloud layer (B)

Input: IoT public Key (PK), private Key (PrK)
Output: digital signature

Begin
Step 1: Calculate Message = encrypted data(temp + SPO2).
Message = temperature + SPO2;
Step 2: Digest = SHA-256(message).
DigestA = SHA-256(Message) ;
Step 3: SignatureA = {Digest}PrK.
signatureA = Encrypt(DigestA, PrK);
Step 4:before sending signatureA to cloud layer (B) verify the digest and signature.
verify(digestA, signatureA);
Step 5: IoT device sends SignatureA and public key to cloud layer (B).
Cloud Receive(signatureA, PK, Message);
End

Function CloudReceive(signatureA, PK, Message)
Step 6: Cloud decrypts signatureA using public key (PK).
Step 6: Cloud computes DigestB in the same way in A.
DigestB = SHA-256(Message) ;
Step 7: Cloud compare SHA-256 with SHA-256 comes from the client side
If (DigestB== DigestA) {
Continue sending data
} **else** {
Print("Authentication failed. Denying data transmission from IoT device (A) to cloud Server (B).");
}

End of Algorithm

It is clearly shown from algorithm (3.3), the PK_A and PrK_A are equivalent to public and private keys which are generated for only one session using the RSA algorithm as shown in an algorithm (3.4).

Algorithm 3.4: IoT Device Key Generation Approach

Input: Two prime numbers for IoT device (Sp, Sq)
Output: PkA, PrKA // public and private keys for IoT device (A)
Begin
Step 1: Select pA, qA where pA ≠ qA are both primes.
pA = Sp;
qA = Sq;
Step 2: Calculate n
n = pA * qA;
Step 3: Calculate Φ(n)
Φ(n) = (pA - 1) * (qA - 1);
Step 4: Select integer (e), where 1 < e < Φ(n) and gcd(Φ(n), e) = 1.
e = SelectInteger(Φ(n));
while (gcd(Φ(n), e) != 1) {
e = SelectInteger(Φ(n));
}
Step 5: Calculate d where d * e mod Φ(n) = 1.
d = CalculateModInverse(e, Φ(n));
Step 6: Public Key for IoT device (A) = PkA.
PkA = (e, n);
Step 7: Private Key for IoT device (A) = PrKA.
PrKA = (d, n);
End

End of Algorithm

Table (3.3) shows the public and private keys used in the authentication process which is implemented in the real environment.

Table 3.3: The main parameters of the authentication process between the sensor and cloud

Item	Description
Public Key(P_K)	MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAk m6ugVcHuqQbzhUCEWwa X+VyM6ffNtz+6+EkHh8MDLToTd/F6qu9De0M83jX2A+0l28n1+ vPkCj2Abfwyje4 66PYlOVVmcb06UorN0U0Zi2Wiu1hrjy9gjg64w5hk1l+oGXTkBD w+yg/Y2TMqFn/ KdaeF6kZh3qmNeRCpgBvrrSG9TKWNpMRVu0+NWHrRHyIsK YNR+CF5rsEA9WXImRU K+jd3ZER7oa1OT5HWuYOO0ZQ+D8W6v5kSubPWYsvBuS8fSb

	tBPfg5QxjzNAemRMJ QTLTVG8Ru4Kx0rVjEOl6DRbpiUql8tEvZCJzEz/zMyPLWwI3eY AujLnBm+z5Y0ZE IwIDAQAB
Private Key(Prk)	MIIEpAIBAAKCAQEAkm6ugVcHuqQbzhUCEWwaX+VyM6ffNt z+6+EkHh8MDLToTd/F 6qu9De0M83jX2A+0l28n1+vPkCj2Abfwyje466PYlOVVmcb06Uo rN0U0Zi2Wiu1h rjy9gjg64w5hk1l+oGXTkBDw+yg/Y2TMqFn/KdaeF6kZh3qmNeR CpgBvrrSG9TKW NpMRVu0+NWHrRHyIsKYNR+CF5rsEA9WXImRUK+jd3ZER7 oa1OT5HWuYOO0ZQ+D8W 6v5kSubPWYsvBuS8fSbtBPfg5QxjzNAemRMJQTLTVG8Ru4Kx0 rVjEOl6DRbpiUql 8tEvZCJzEz/zMyPLWwI3eYAujLnBm+z5Y0ZEIwIDAQABAoIB AHU1BJbge35IJz1g EWSym7XQaQWxw/gdEirp7cfyX+m30A2tJLT+u/dum6QyQMpf X9Sqk0DALkackrSz b5fc3DY40n3zxv2sg5JL3KWDzjpPGSWgdkRJuAKIUZI2i6Guy61 Fc/7noIvYHkzH fQzv5E6vy6DyqE2JvYp7Z2GKDdrOzRfZkKNJjXA+Dx/Zy5X5xJ H+9pODwUjT7wzF WS2Y9EnNk3kFv9c0FYS9BUhr6uihvPo5CLxDkxLNFlxf4lmPWr eTCQ8q0Y9s1OE+ a3Jynq0GktWrTNom+Or24p/I/hNHOvQR6RiIxMCFmGZyfyUjGe G6nCYf58HrDu5X cmQpwQECgYEAwicTn1N+dnj5QuyPtdqCCV/Q45HemAX7x5Q/ RGzMR5+UzmKeUPL5 QwsyQc7LHGud82RfevasX6tIApP0kGTvQ6zeL8JsxJbWS+YKt6y liQ1hlOSxAXzN B3uM3uyWJU5YxW0TeMVj6TtdtIRql2Ci0c9AwdzqjpG1oSkMBk

	vLJIECgYEAwRQS Ao+oWp2Oeg16evKwuwpncZqCzSJG8KckLP2a98n9fU5T3sgCIz WkjUSdSbNec75e vkA5rc2bBFccBpuWOBVJTCQds/YGPxvHOo7zyc6gHaA0+LhiI YFUNKPvPTOlaXnJ 9kIu1eYc/QEufg7/Vl9vsoXHcoR/LTGftmiTBqMCgYEAiApbfHtD lH3lolghxnnq PelOc/bE8t7WchzVS/u0E8ekvAxsBCqML8cLmwLsXOinT2EYag +n7o0UswG7DuY6 pL/fG7XvArzyQVJaViL6BpNCudKYmIM7IrAWoClEd7VzDDsGt swQ2t8HJLLYAWPs EnlXVb+W9vjwPJGHZG39VIECgYEAm9S1DHghqrELMxr5xY1 mci9deeT4uNN3afec TOuKTX8rJh03t9C5WO2Ml6/POpKYItZ85RQ0+T++Y2xi5DpirX UQqXFkv8BLHlgv HeYOgMxRklehKTS3js8wxSOijUTHti9jf2u/HUgSEWkIuxkjn/R6 1B0A8j5Z3oH0 o2YXc/0CgYAxxlN33P4WBnTcfS8pqCtZMMEMMN2s4U6GUPZ LPz+hjCAwCMubRvYY +3T15jlNK1GFmRAMRpdQoD/3g4qKuv+QTqmNzA6q9sf1meIgn vweq9BUF7YklxkI LOkEyorSOQ1AFompeV50HI9V/S5LmEQYnrn/69yP76R36TBliS aQ+g==

Table (3.3) includes the following items:

Public Key (PK): The public key is a long string of characters represented in Base64 encoding. It is used for encrypting data that can only be decrypted with the corresponding private key.

Private Key (Prk): The private key is also a long string of characters represented in Base64 encoding. It is kept secret and used for decrypting data that has been encrypted with the associated public key.

The entropy measurements indicate the level of randomness or unpredictability of the keys. In this case, the entropy value of the public key (PK) is reported as 5.15 and the number of characters in the public key was 328 , while the entropy of the private key (Prk) is reported as 5.21 and the number of characters in the private key was 1592. Higher entropy values generally indicate stronger keys with a higher level of randomness. Entropy is a crucial factor in cryptographic systems as it ensures the difficulty for an attacker to guess or deduce the key through brute-force or other methods. Therefore, higher entropy values indicate increased robustness and security of the RSA keys used in the authentication process described in the table.

After the data is encrypted and signed, it is sent to the public cloud computing platform type virtual private server (VPS) with CPU 2*2.6 GHz, RAM 4GB, Bandwidth 8TB, SSD, and internet speed is 100Mbps. which acts as the central management and storage.

3.3.5 Decryption data using HLCA Algorithm

In the decryption process, the encrypted data (ciphertext) received from the Raspberry Pi is used as input along with the shared secret key generated through the Elliptic Curve Diffie-Hellman (ECDH) key exchange protocol. The same key used in the encryption process is used for decryption. The decryption algorithm then performs the inverse process of the encryption algorithm to obtain the original plaintext data.

Figure (3.5) showed the decryption of the HLCA system in the server node. Algorithm 3.5 showed the used HLCA Decryption Algorithm in the server(cloud).

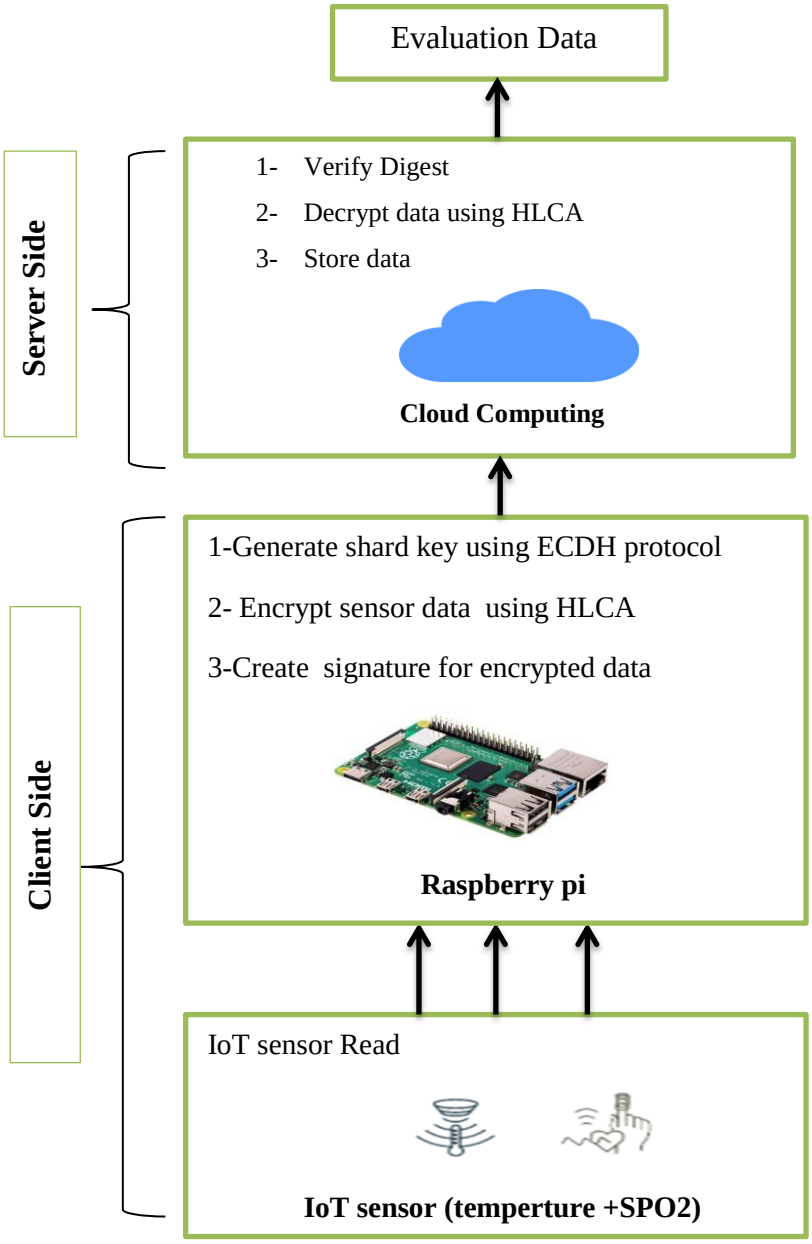


Figure 3.5:Decryption Data with HLCA

Algorithm 3.5: HLCA Decryption Algorithm

Input: Encrypted(ciphertext) data + shared secret key
Output: Decrypted data

1. System Initialization;
2. Receive encrypted data packets with the signature from the client node;
3. Verify Digest
4. **if** the Digest of the client == Digest of the server, then:
5. Get keyvalue
6. Set decryptvalue

```
Plane1 = speck.dec(key, plane)        //first decrypt data with SPECK algorithm
Plane1 = prince.dec(key, plane)  // finally decrypt data with PRINCE algorithm
print (plane1)
```

7. While still receiving encrypted data, do:
8. Build decrypt data;
9. **End;**
10. Store the decrypted data

End of algorithm

3.4 Summary

This chapter serves as a guide for implementing a secure IoT-cloud system based on IoT healthcare sensors. It covers the necessary steps, installation requirements, and execution methods. Additionally, it introduces authentication and encryption algorithms to ensure the system's security and protect sensitive healthcare data.

CHAPTER FOUR

IMPLEMENTATION AND RESULTS

4.1 Introduction

This chapter focuses on the implementation details of an HLCA (Hybrid Lightweight Cryptography Approach) system with cloud computing. The implementation is divided into two main sections: the Authentication section and the Encryption section. The Authentication section describes how the sensor layers connect with the cloud using an authentication approach based on the RSA public key algorithm. The Encryption section explains the main encryption process between IoT devices and the cloud using a private shared key protocol(ECDH).

4.2 The Proposed System Implementation and Results

The proposed system is based on the utilization of a Raspberry Pi microcontroller and healthcare sensors, specifically for measuring SPO2 (Blood Oxygen Saturation) and temperature. These sensors are crucial for monitoring the health status of individuals. The system enables the exchange of sensing information between the sensor layer and the cloud layer, facilitating real-time data analysis and remote access to healthcare data. Figure 4.1 provides a general view of the proposed system, illustrating the overall architecture and the flow of data between the different layers.

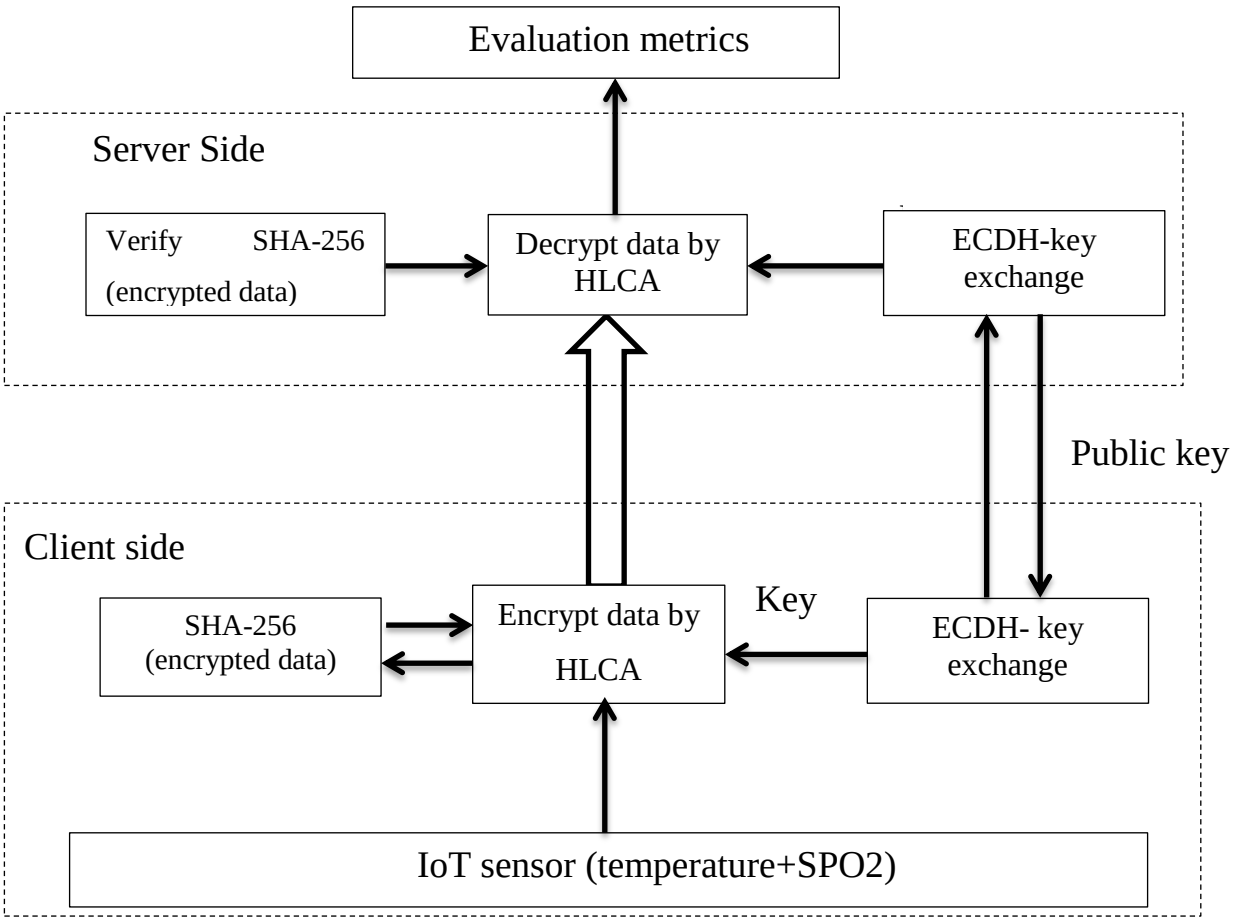


Figure 4.1: general implementation approach of the proposed System.

Figure 4.2 shows the encrypted data with 10 reads, shared key, and digital signature in the raspberry pi device.

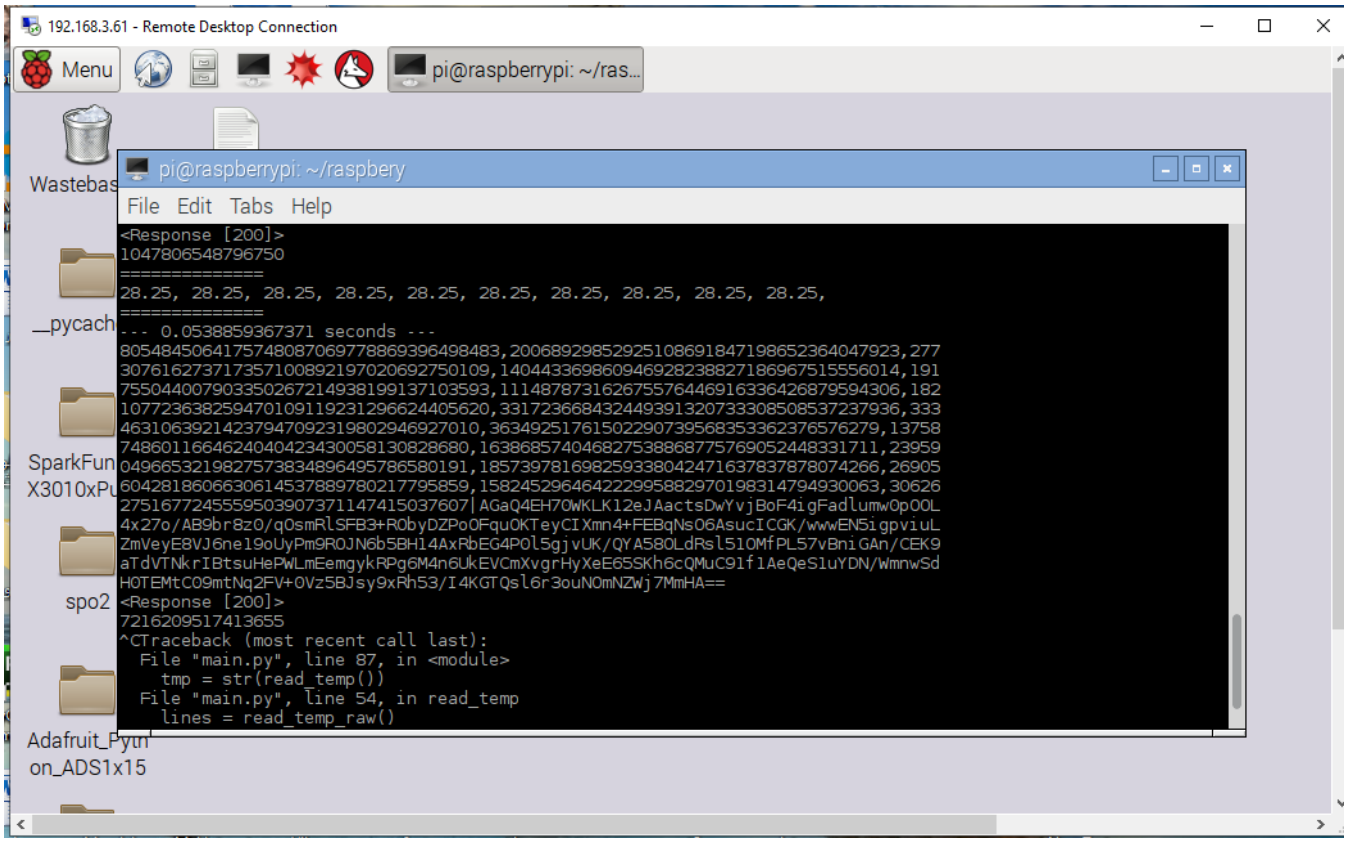
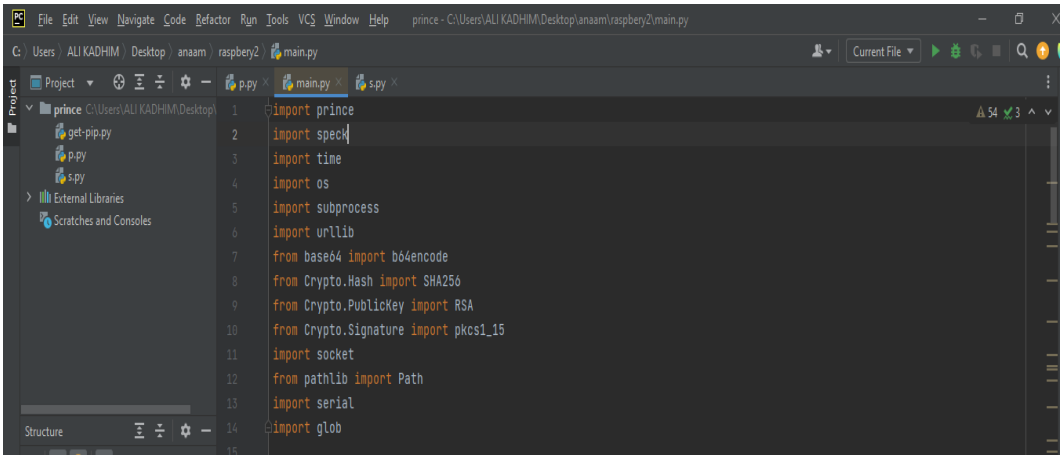


Figure 4.2: Data Results from Raspberry Pi.

Application programming interface (API) that included in client side are shown in figure (4.3).



In figure 4.4 The encrypted data and digital signature message arrive to the server side (cloud). The following steps are applied:

1. Calculate the message digest of the encrypted data using the same hashing algorithm that is used to generate the original message digest. the SHA-256 is used to generate the message digest, then apply SHA-256 to the received encrypted data.
2. Compare the calculated message digest with the received message digest. If they match, it indicates that the encrypted data has not been tampered with during transmission.
3. If the message digests match, proceed to decrypt the encrypted data using the HLCA decryption algorithm and the private key.

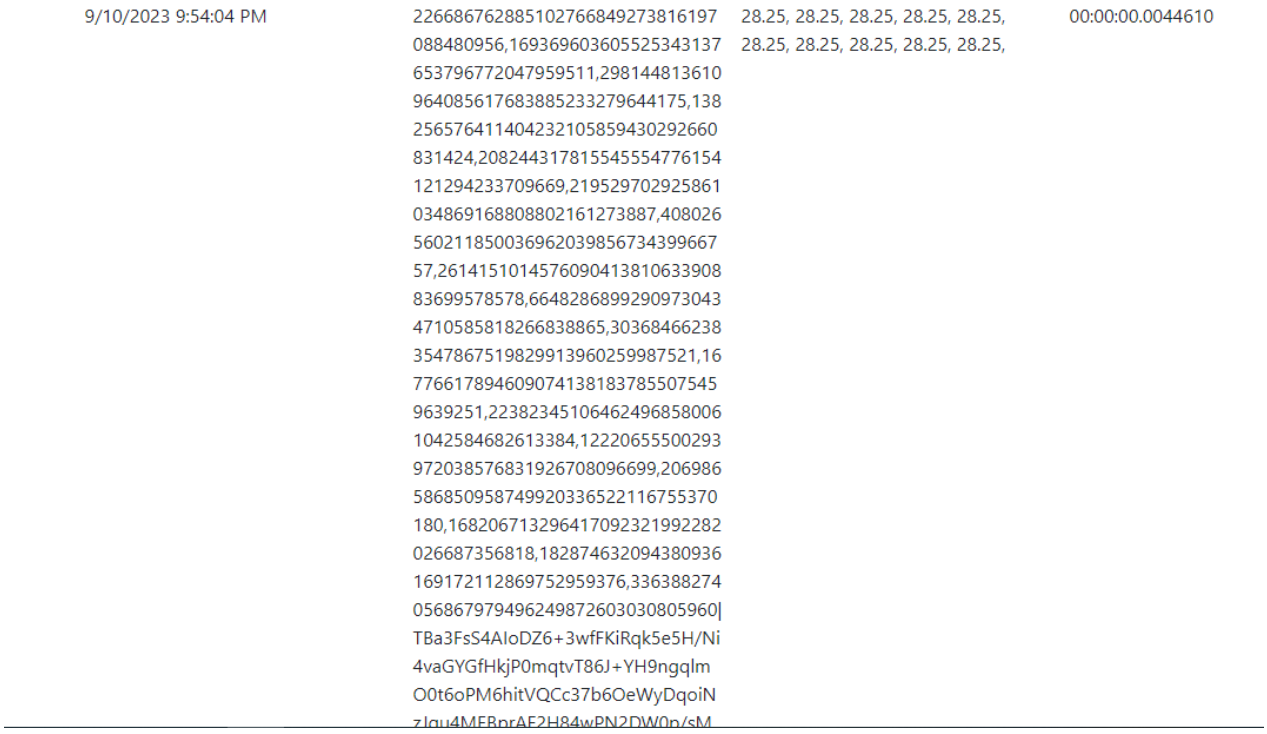


Figure 4.4: Data Results from Cloud.

4.3 The proposed System Results

The following results are based on three case studies the 1st case study is based on data encryption with a lightweight encryption algorithm (PRINCE algorithm), the 2nd is based on data encryption with a lightweight encryption algorithm (SPECK algorithm), the 3rd is based on data encryption with a hybrid lightweight encryption algorithm (HLCA algorithm).

4.3.1 The 1st case study Results

In this case, we show the results of a security system based on a PRINCE lightweight algorithm to encrypt and decrypt data traffic depending on different numbers of reads (10 to 10000 reads) and different sizes of plaintext from 90 to 91998 bytes.

Table (4.1) shows the results of encryption with the PRINCE algorithm.

Table 4.1: the result of the PRINCE algorithm

No. of reads	Data Size in Byte	Encryption Time (ms)	Throughput (Byte/ ms)
10	90	9	10
100	918	75	12.24
1000	9198	197	46.69
5000	45998	310	148.38
10000	91998	482	190.86

In Figure (4.5) We notice that the number of reads increases, and the encryption time also increases. This makes sense because with more reads, there is a larger volume of data that needs to be processed and encrypted, which takes

additional time.

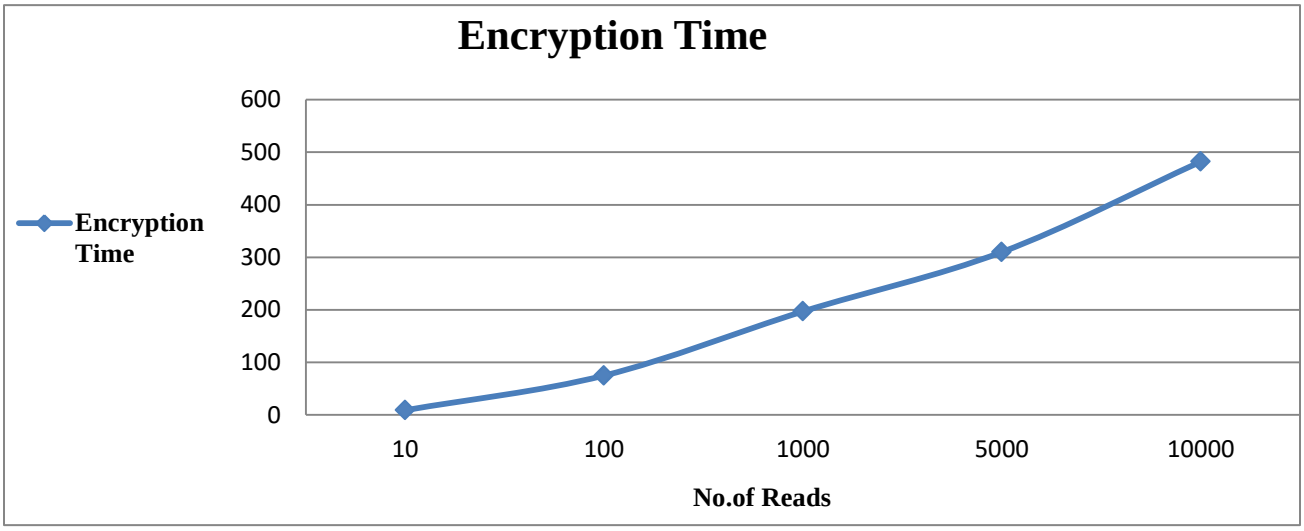


Figure 4.5: Encryption Time of PRINCE algorithm

Figure (4.6) showed that the throughput value is increase due to the number of reads increases. In this case, the increase in throughput could be attributed to the increased encryption time as the number of reads increases.

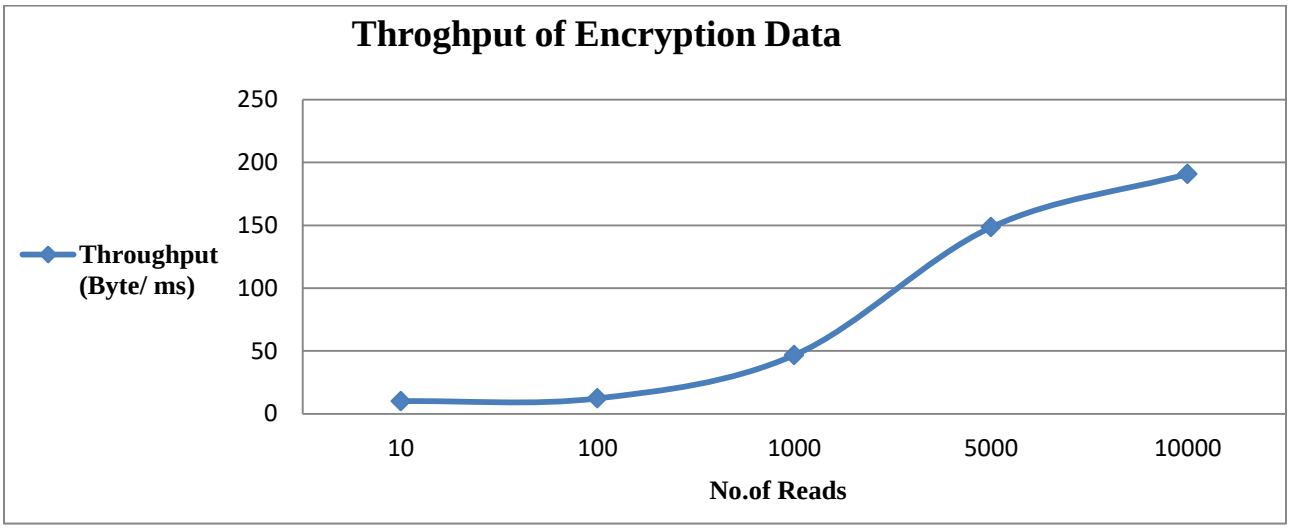


Figure 4.6: Throughput Encryption Value of PRINCE algorithm

Furthermore, the decryption results showed that the number of reads increased affects the required time for decryption and the throughput. table (4.2) shows the result of decryption and throughput for the PRINCE algorithm.

Table 4.2: The results of Decryption Data using the PRINCE algorithm.

No. of reads	**Data Size in Byte**	**Decryption Time (ms)**	**Throughput (Byte/ ms)**
10	90	7	12.85
100	918	59	15.5
1000	9198	175	52.56
5000	45998	293	157
10000	91998	461	199.5

In figure (4.7) and figure (4.8), we notice that the decryption process is impacted by the number of reads. As the number of reads increases, the decryption time also increases, resulting increase in throughput.

As the number of reads increases from 10 to 100, the decryption time increases from 7 ms to 59 ms. The throughput also increases slightly from 12.85 Byte/ms to 15.5 Byte/ms. With a further increase in the number of reads to 1000, the decryption time increases to 175 ms, and the throughput improves to 52.56 Byte/ms. When the number of reads is 5000, the decryption time increases to 293 ms, while the throughput increases to 157 Byte/ms. Lastly, with 10,000 reads, the decryption time further increases to 461 ms, and the throughput increases to 199.5 Byte/ms.

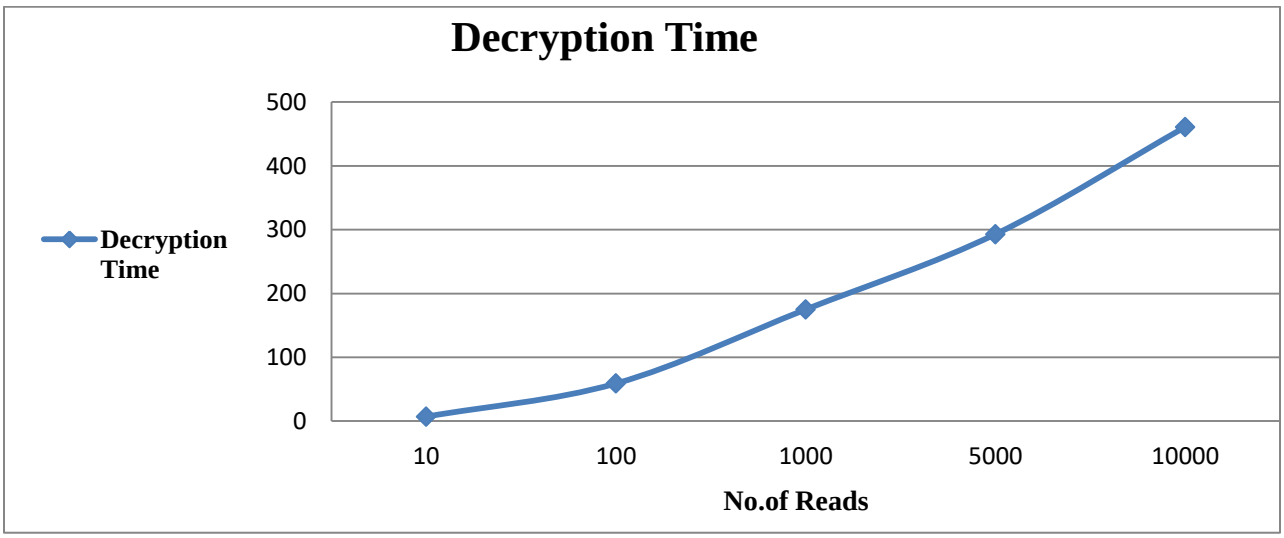


Figure 4.7: Decryption time of the PRINCE algorithm

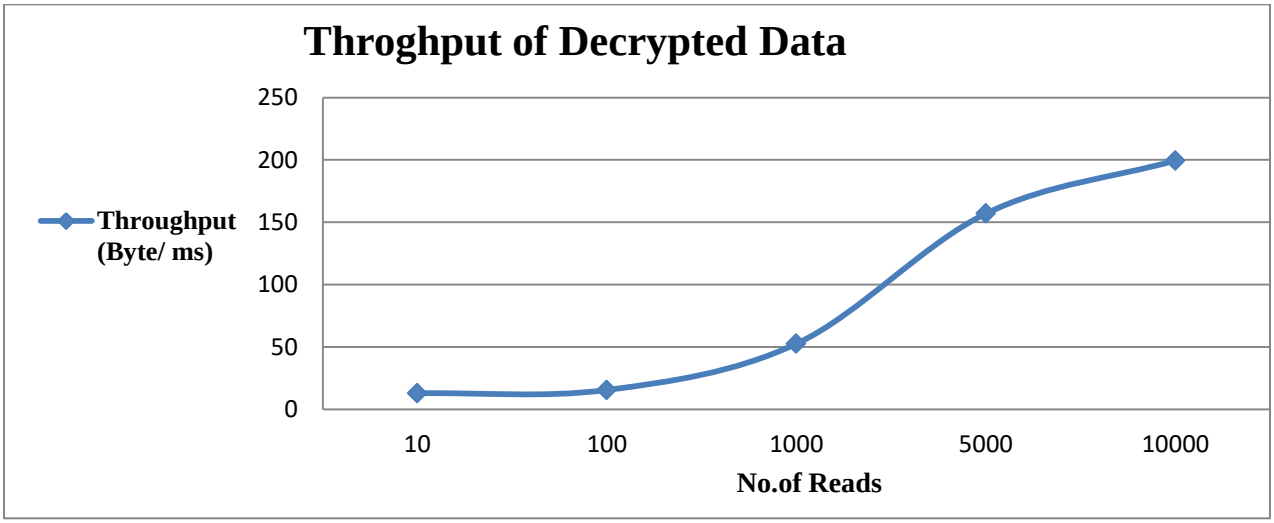


Figure 4.8:Throughput decryption value of the PRINCE algorithm

Table (4.3) showed the value of delay (latency) values in milliseconds for the PRINCE lightweight encryption algorithm. The value is increased due to the increased number of sensors and the size of input data.

Table 4.3:Delay value in PRINCE algorithm

No. of reads	**Delay in ms (Encryption time + Decryption time)**
10	16
100	134
1000	372
5000	603
10000	943

From the table, we can observe that as the number of reads increases, the delay in the encryption process also increases. This suggests that the encryption algorithm takes more time to process a larger number of sensor inputs or a larger size of input data.

Table (4.4) and Figure (4.9), show the result of Entropy which is an important parameter for security analysis that measures the randomness of data. The result shows the entropy value are increase when increase number of data.

Table 4.4: Entropy result of PRINCE algorithm

No. of reads	**Data Size in Byte**	**Entropy**
10	90	7.4051651499166
100	918	7.4912531832244
1000	9198	7.6204257149501
5000	45998	7.7812784792104
10000	91998	7.8678321053291

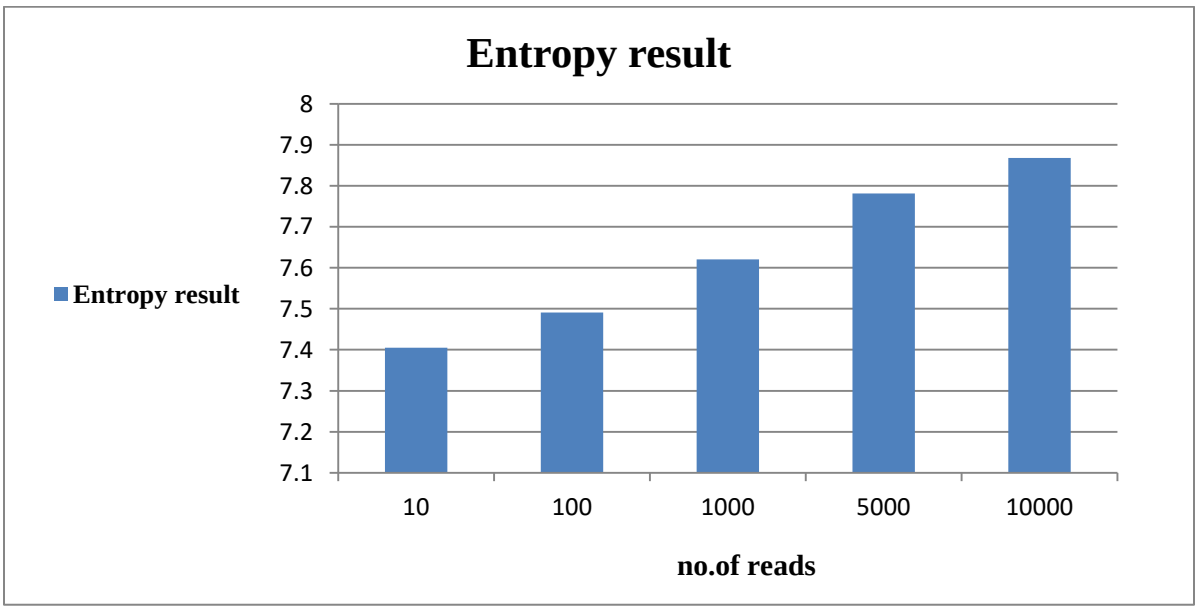


Figure 4.9: Entropy result of PRINCE algorithm

From figure (4.9) and table (4.4), notice that the entropy value with 10 reads is slightly lower compared to the entropy values with higher numbers of reads. A lower entropy value indicates a lower level of randomness or unpredictability in the encrypted data.

In general, it is desirable to have a high entropy value regardless of the number of reads or data size. A high entropy value signifies a higher level of randomness and provides stronger security. Algorithms with higher entropy values are generally considered more secure, as they generate ciphertext that is harder to decrypt or predict.

Therefore, it is important to choose an encryption algorithm that maintains a high level of entropy even when dealing with a smaller number of reads or a limited amount of data. This helps ensure the security and confidentiality of the encrypted messages, even in scenarios with fewer observations or smaller data sizes.

4.3.2 The 2nd case study Results

In this case, we have shown the results of a security system based on a SPECK lightweight algorithm to encrypt and decrypt data traffic depending on different numbers of reads (10 to 10000 reads) and different sizes of plaintext from 90 to 91998 bytes.

Table (4.5) displays the results of the SPECK algorithm in terms of encryption time for different data sizes. It provides information on the number of reads, data size in bytes, encryption time in milliseconds, and throughput in bytes per millisecond.

Table 4.5:the result of the SPECK algorithm

No. of reads	**Data Size in Byte**	**Encryption Time (ms)**	**Throughput (Byte/ ms)**
10	90	7	12.85
100	918	52	17.65
1000	9198	173	53.16
5000	45998	267	172.27
10000	91998	407	226.03

In Figure (4.10) notice that, As the data size increases, the encryption time generally increases. For example, with 10 reads and a data size of 90 bytes, the encryption time is 7 milliseconds. However, with 10000 reads and a data size of 91998 bytes, the encryption time increases to 407 milliseconds.

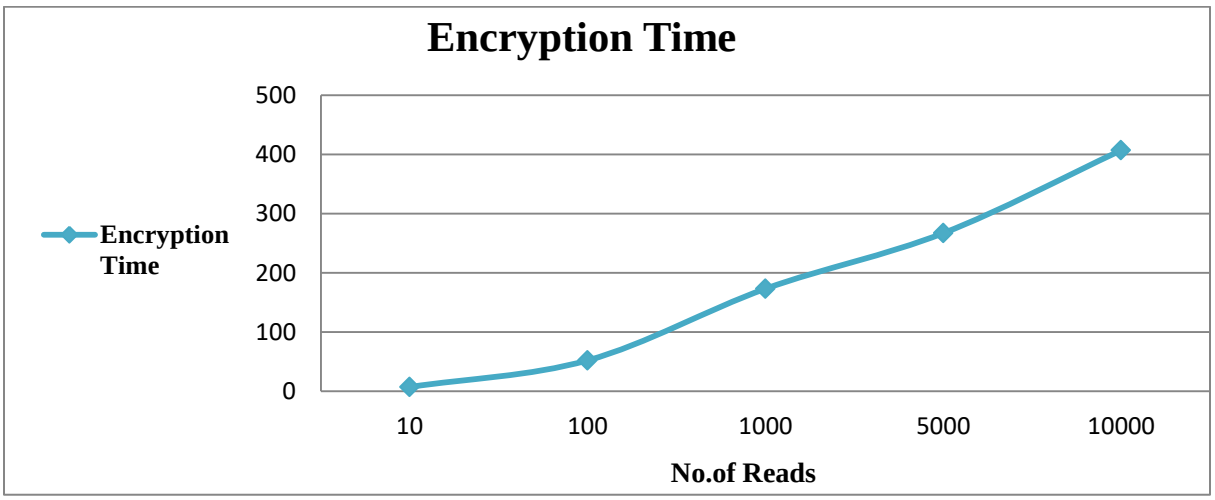


Figure 4.10: The results of Encryption Data using the SPECK algorithm

In Figure (4.11) notice that These results indicate the performance of the SPECK algorithm in terms of throughput for different data sizes.

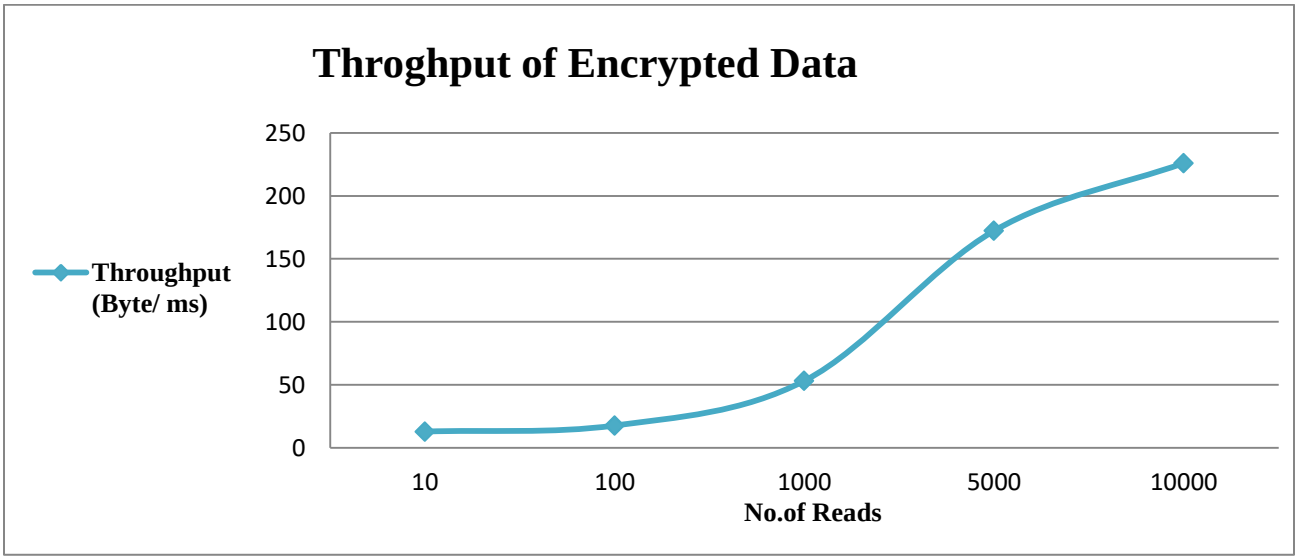


Figure 4.11:Throughput decryption value of the SPECK algorithm

From figure (4.11) notice that The throughput varies with the data size. In general, as the data size increases, the throughput tends to increase. For instance, with 10 reads and a data size of 90 bytes, the throughput is calculated as 12.85 bytes per millisecond. On the other hand, with 10000 reads and a data size of 91998 bytes, the throughput increases to 226.03 bytes per millisecond.

Table (4.6) presents the results of decryption using the SPECK algorithm. The table includes different sets of data sizes and the corresponding decryption times and throughput rates. The impact of the number of reads on the decryption time and throughput is observed.

Table 4.6: The results of Decryption Data using the SPECK algorithm.

No. of reads	Data Size in Byte	Decryption Time (ms)	Throughput (Byte/ ms)
10	90	5	18
100	918	48	19.125
1000	9198	151	60.91
5000	45998	241	190.86
10000	91998	382	240.83

Figure (4.12) displays the result of decryption time with the SPECK algorithm, it can be observed that as the number of reads increases, the required decryption time generally increases, indicating a longer time to decrypt larger amounts of data.

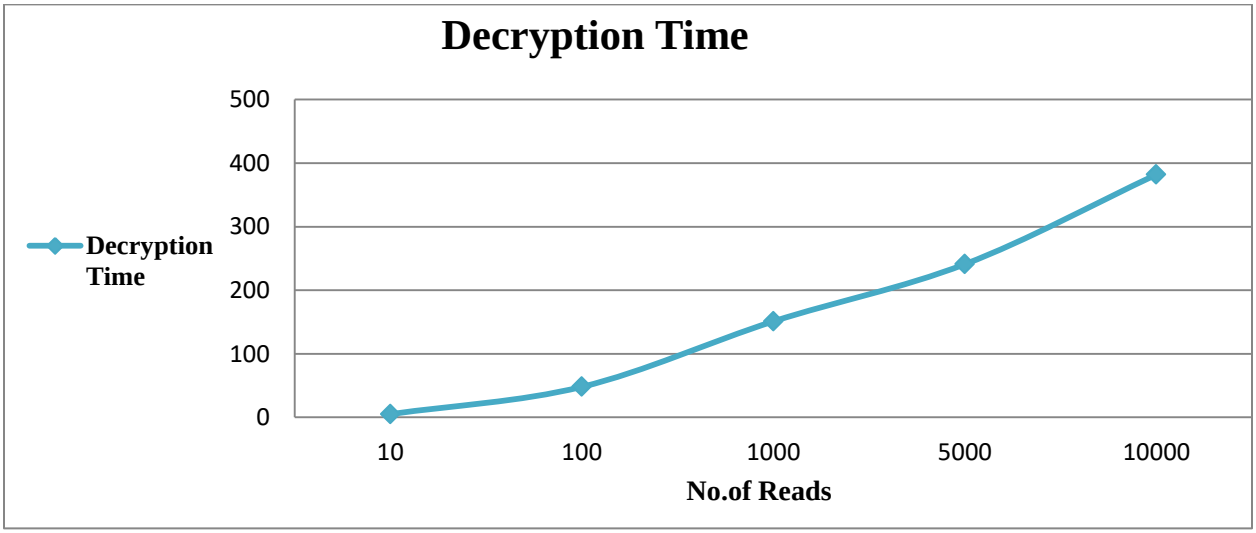


Figure 4.12: The results of Decryption Time using the SPECK algorithm.

From figure (4.12) notice that, As the number of reads increases, the decryption time generally increases. For instance, with 10 reads and a data size of 90 bytes, the decryption time is 5 milliseconds. However, with 10000 reads and a data size of 91998 bytes, the decryption time increases to 382 milliseconds

Figure (4.13) display the throughput of decrypted data. The result shows a slight increase, indicating a higher rate of data processed per unit of time.

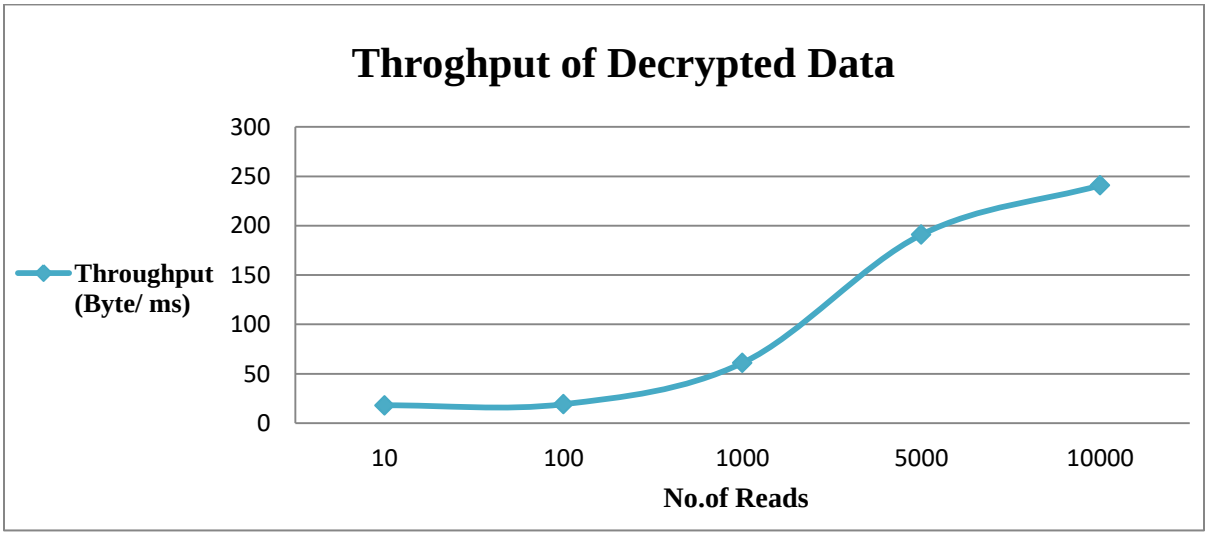


Figure 4.13:Throughput of decrypted data using SPECK algorithm

From figure (4.13) notice that The throughput also varies with the number of reads and data size. It shows a slight increase as the number of reads and data size increase. For example, with 10 reads and a data size of 90 bytes, the throughput is 18 bytes per millisecond. As the number of reads and data size increase, the throughput gradually increases, reaching 240.83 bytes per millisecond with 10000 reads and a data size of 91998 bytes.

Table (4.7) presents the values of delay (latency) in the SPECK lightweight encryption algorithm. The delay values increase as the number of sensors and the size of the input data increase.

Table 4.7:Delay result with SPECK encryption algorithm

No. of reads	**Delay in ms (Encryption time + Decryption time)**
10	12
100	100
1000	324
5000	508
10000	789

From table(4.7) notice that, As the number of reads increases, the delay also increases. For example, with 10 reads, the delay is 12 ms. However, with 10000 reads, the delay increases to 789 ms.

These results suggest that the delay in the SPECK lightweight encryption algorithm is influenced by both the number of reads and the size of the input data. As these factors increase, the delay in processing the data also increases.

Table (4.8) and Figure (4.14), show the result of Entropy. The result shows the entropy value are increase when increase number of data.

Table 4.8:Entropy result of SPECK algorithm

No. of reads	**Data Size in Byte**	**Entropy**
10	90	7.35384213996737
100	918	7.41944153544838
1000	9198	7.59468291465212
5000	45998	7.71127684773115
10000	91998	7.80446892214571

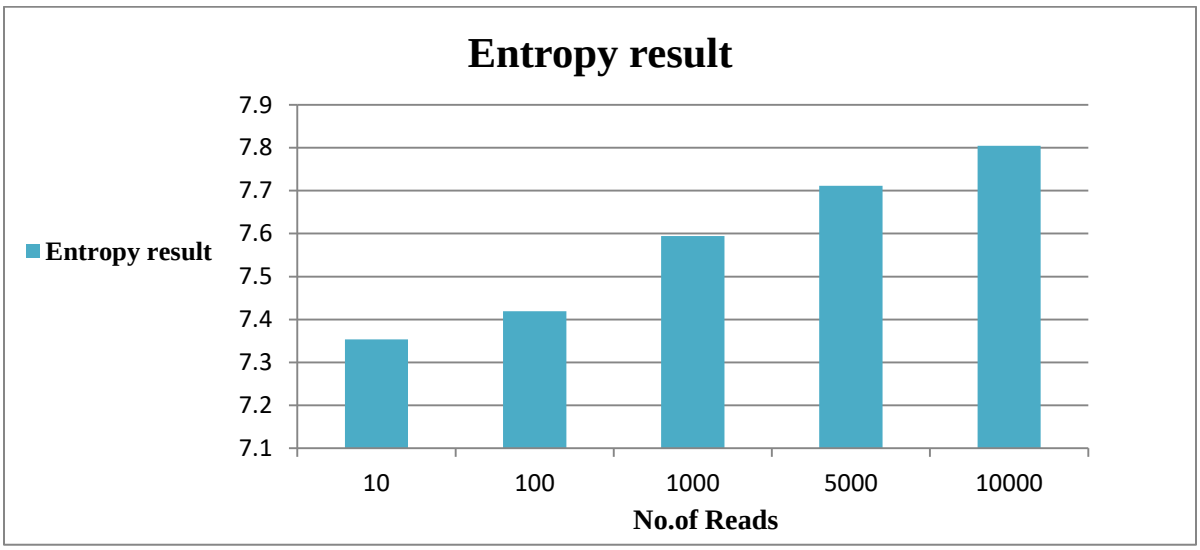


Figure 4.14: the Entropy result of the SPECK algorithm

From figure (2.14) notice that The entropy values generally increase as the number of reads and the size of the data increase. For example, with 10 reads and a data size of 90 bytes, the entropy is 7.35384213996737. However, with 10000 reads and a data size of 91998 bytes, the entropy increases to 7.8044689221457.

The entropy value with 10 reads is slightly lower Therefore, it is important to choose an encryption algorithm that maintains a high level of entropy even when dealing with a smaller number of reads or a limited amount of data. This helps ensure the security and confidentiality of the encrypted messages, even in scenarios with smaller data sizes.

4.3.3 The 3rd case study Results

In this case, we have shown the results of a security system based on the Hybrid Lightweight Cryptography Approach HLCA (PRINCE + SPECK) to encrypt and decrypt data traffic depending on different numbers of reads (10 to 10000 reads).

Table (4.9) displays the results of encryption using the HLCA. The table includes various sets of data sizes and the corresponding encryption times and throughput rates. It is evident from the table that as the number of data increases, the encryption time also increases

Table 4.9: the result of the HLCA algorithm

No. of reads	Data Size in Byte	Encryption Time (ms)	Throughput (Byte/ ms)
10	90	14	6.42
100	918	89	10.31
1000	9198	213	43.18
5000	45998	348	132.17
10000	91998	512	179.68

In figure (4.15) we observed, when the number of reads (data) increases, the encryption time also increases. This indicates that the HLCA algorithm requires more time to process larger amounts of data.

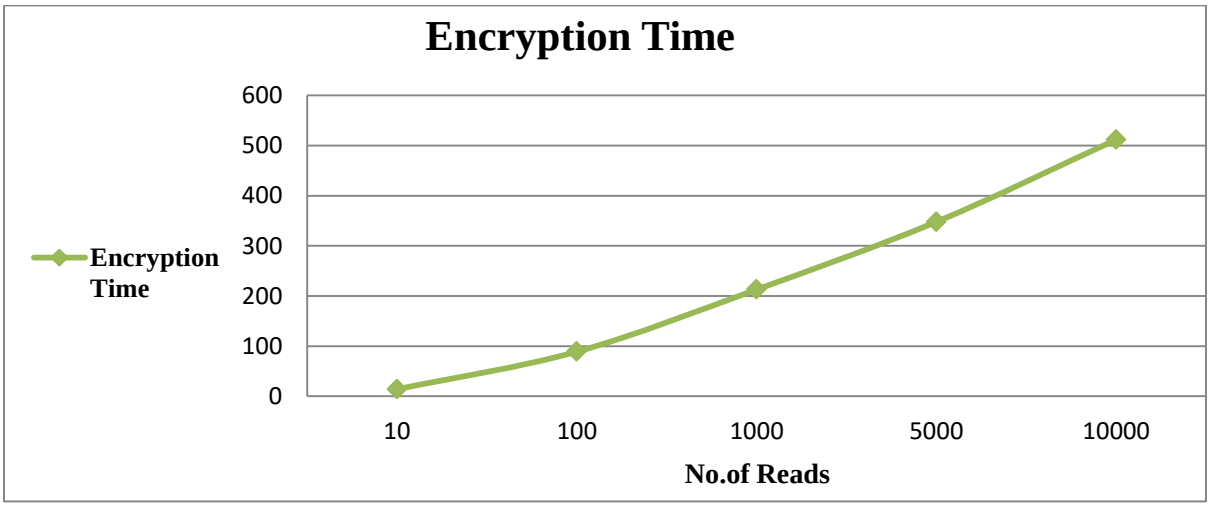


Figure 4.15: the result of Encryption time for HLCA

From figure (4.15) notice that, As the number of reads (data) increases, the encryption time also increases. For instance, with 10 reads and a data size of 90

bytes, the encryption time is 14 milliseconds. However, with 10000 reads and a data size of 91998 bytes, the encryption time increases to 512 milliseconds. This indicates that the HLCA algorithm requires more time to process larger amounts of data.

Figure (4.16) showed that the throughput increases as the data size increases because the encryption process takes more time for larger amounts of data.

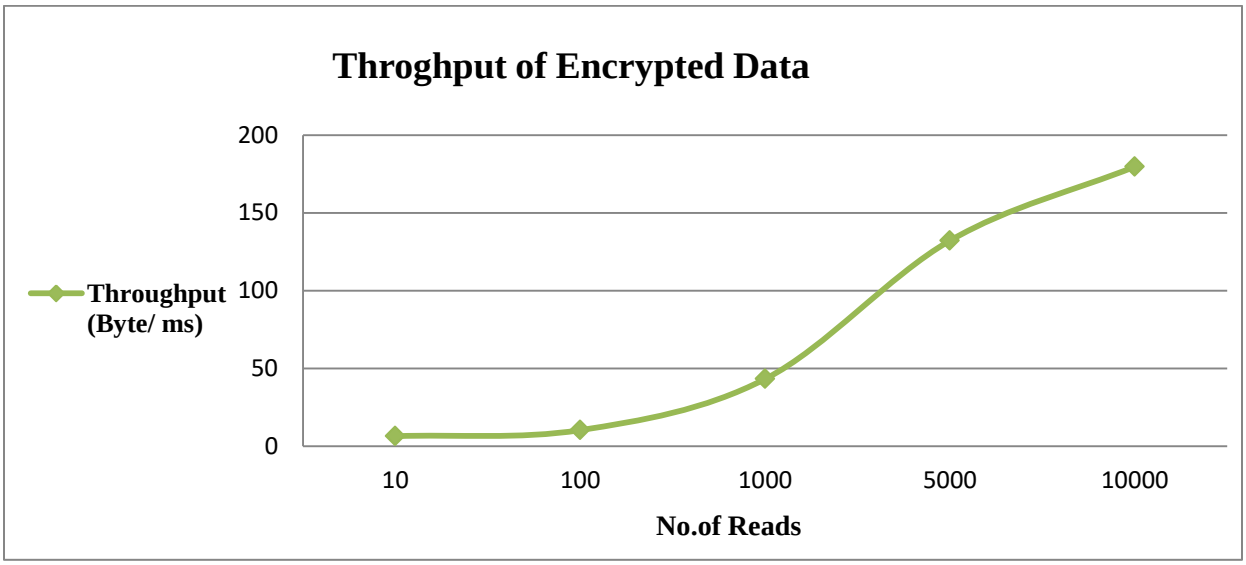


Figure 4.16: the result Throughput of Encrypted data for HLCA

From figure (4.16) notice that The throughput varies with the number of reads and data size. In general, as the number of reads increases, the throughput tends to increase. For example, with 10 reads and a data size of 90 bytes, the throughput is 6.42 bytes per millisecond. However, with 10000 reads and a data size of 91998 bytes, the throughput increases to 179.68 bytes per millisecond.

Table (4.10) presents the results of decryption using the HLCA. The table includes different sets of data sizes and the corresponding decryption times and

throughput rates. The impact of the number of reads on the decryption time and throughput is observed.

Table 4.10: The results of Decryption Data using the HLCA algorithm.

No. of reads	Data Size in Byte	Decryption Time (ms)	Throughput (Byte/ ms)
10	90	9	10
100	918	72	12.75
1000	9198	196	47
5000	45998	320	143.74
10000	91998	481	191.26

In figure (4.17) we observed that as the number of reads (data) increases, the decryption time generally increases. This suggests that the HLCA algorithm requires more time to decrypt larger amounts of data.

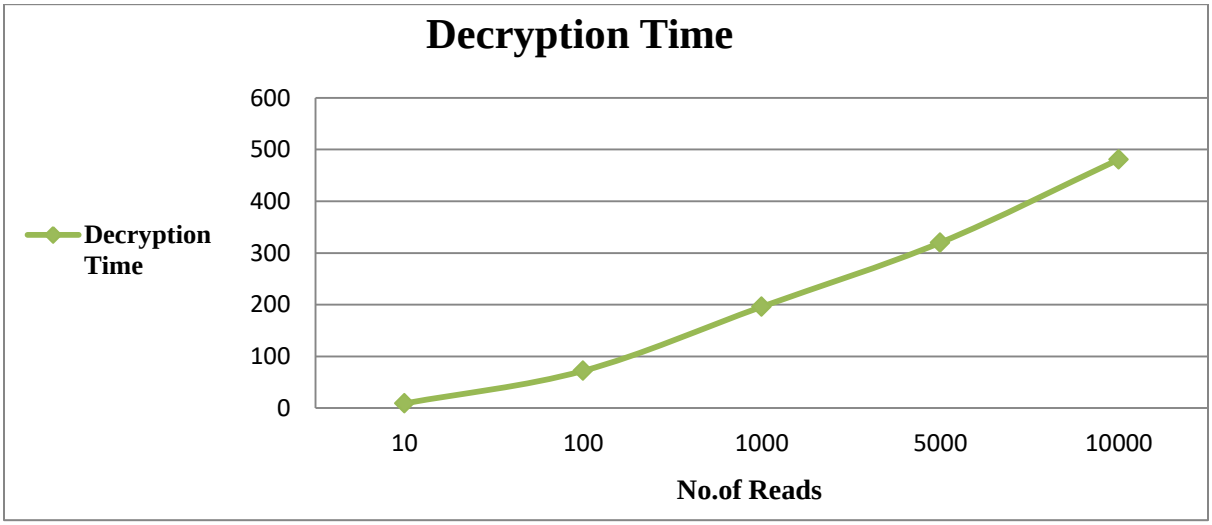


Figure 4.17: the result of Decryption time for HLCA

Figure (4.18) showed that the throughput of decrypted data.

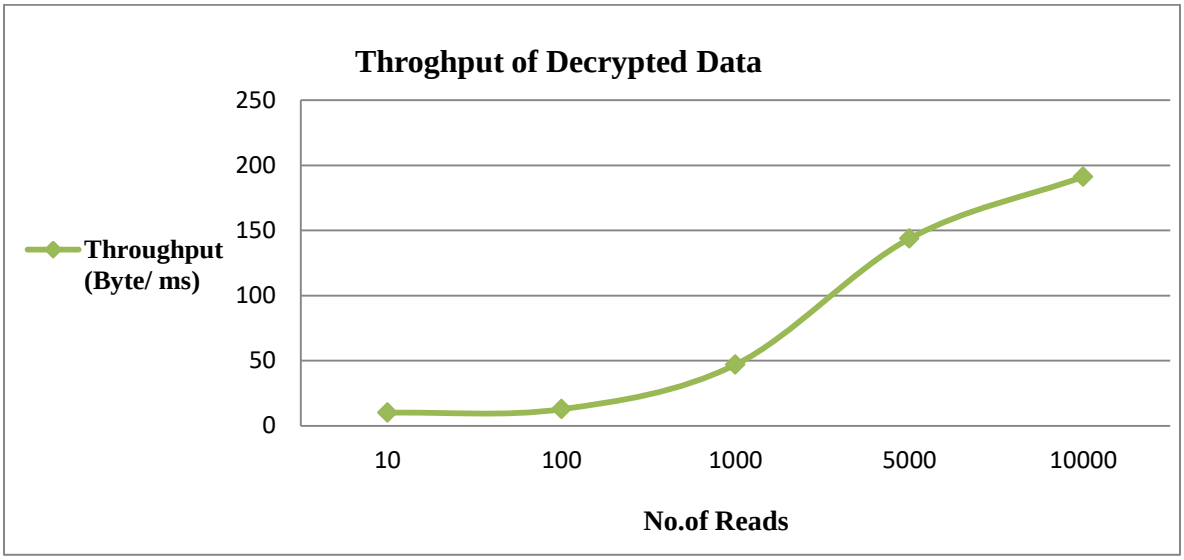


Figure 4.18: Throughput of Decrypted data with HLCA

From figure 4.18, The throughput, measured in bytes per millisecond, varies with the number of reads and data size. It increases as the data size increases. For instance, with 10 reads and a data size of 90 bytes, the throughput is 10 bytes per millisecond. As the number of reads and data size increase, the throughput gradually increases, reaching 191.26 bytes per millisecond with 10000 reads and a data size of 91998 bytes.

Table (4.11) showed the delay of HLCA. As the number of reads increases, the delay also increases. This indicates that the HLCA algorithm takes more time to process larger amounts of data, resulting in higher delays.

Table 4.11:Delay of HLCA

No. of reads	**Delay in ms (Encryption time + Decryption time)**
10	23
100	161
1000	409
5000	668
10000	993

From table (4.11), notice that As the number of reads increases, the delay in the HLCA algorithm also increases. For example, with 10 reads, the delay is 23 milliseconds. However, with 10000 reads, the delay increases to 993 milliseconds.

Based on these results, it can be observed that the HLCA algorithm increased processing time as the number of reads grows. This indicates that the HLCA may be computationally intensive which makes it provide more secure data compare with other lightweight algorithms.

Table (4.12) and Figure (4.19), present the results of entropy calculations in the HLCA algorithm. The entropy values increase as the number of data increases

Table 4.12: Entropy result of the HLCA algorithm

No. of reads	Entropy
10	7.8947989093792
100	7.9136759943012
1000	7.9687732193135
5000	7.9805568943211
10000	7.9991861963155

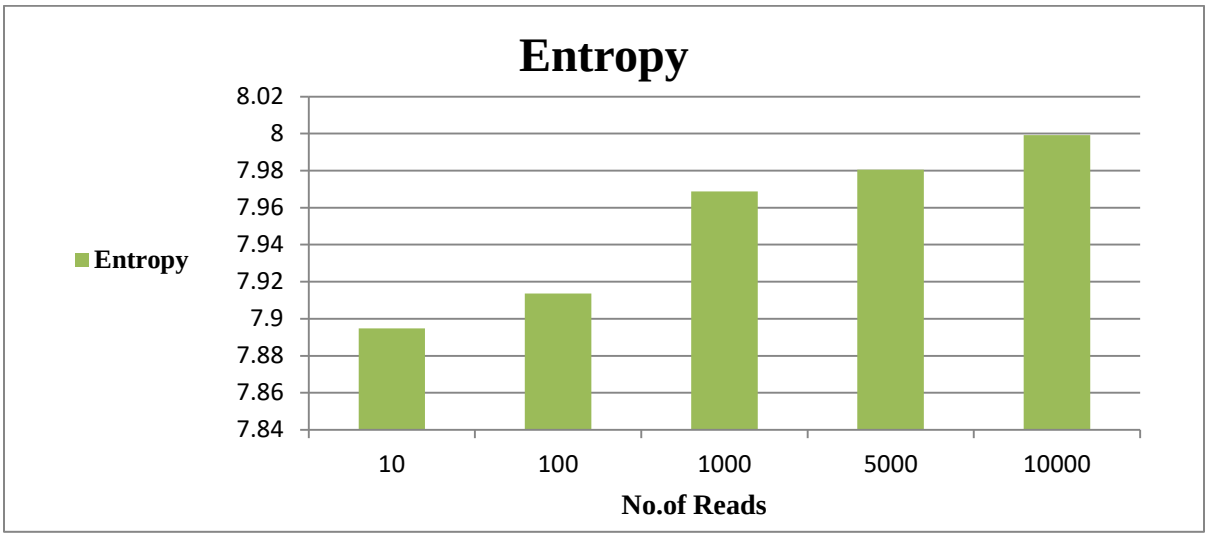


Figure 4.19: the result of Entropy for HLCA

From figure (4.19) notice that the HLCA algorithm demonstrates a high level of security even with a smaller number of reads. This finding highlights the strength of the HLCA algorithm in maintaining a high level of entropy and ensuring data security, even in scenarios with fewer observations. This makes it a reliable choice for applications where data security is a top priority.

4.4 Discussion of the Results

Relying on the results obtained from testing the system and the evaluations presented in the previous sections, the proposed system (HLCA) has several notable features that contribute to its high encryption strength and data integrity:

1. The study focused on addressing security issues in IoT environments.
2. The system ensures data integrity by employing digital signatures .
3. The authentication process utilized public key and privet key of the RSA algorithm for one session and HMAC, known for their robustness in security.
4. The proposed system involved healthcare data sensors such as temperature and blood oxygen level sensors in a real environment.
5. Digest computation is employed between the sensor and server layers for authentication and encrypted data transmission to the cloud.
6. Encryption between IoT devices and cloud servers utilized the HLCA (PRINCE+SPECK) algorithm and a shared key generated by the ECDiffieHellmanCng protocol, eliminating the need for a key management server and enabling data access from anywhere.
7. The proposed system demonstrated a high entropy value, indicating strong security.
8. The proposed system exhibited improved latency for encrypted data exchange compared to other works.

Figure (4.20) showed the main comparison of different cases and explained the proposed system (HLCA) is better results in security based on entropy results.

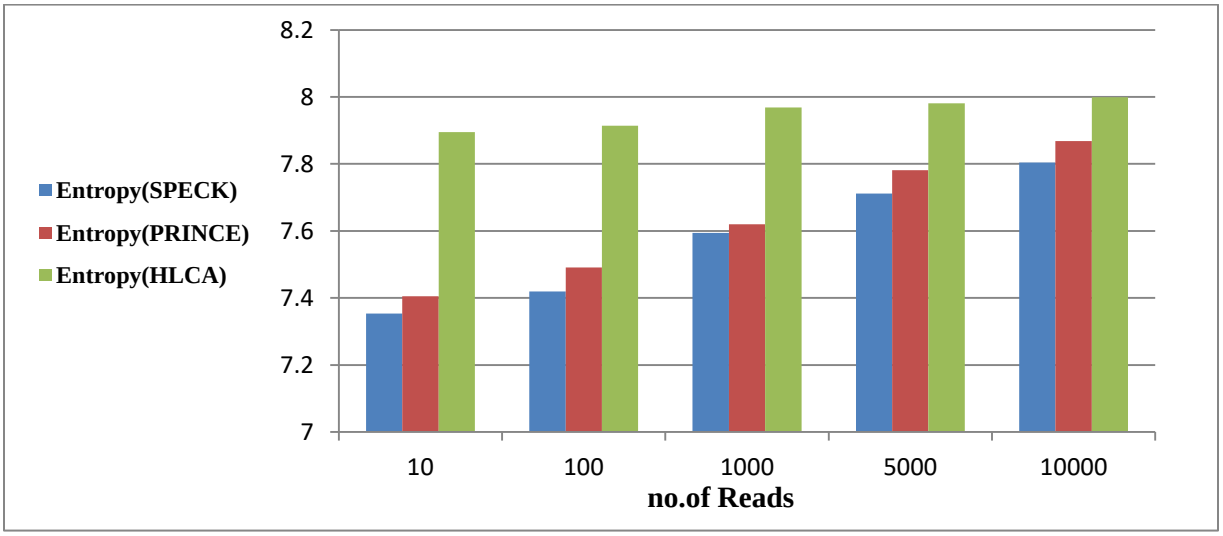


Figure 4.20: The main comparison of 1st, 2nd, and 3rd cases.

4.5 System Comparison

The proposed system was compared with different related work based on different evaluation parameters. In Table 4.13, the proposed system has an average latency of **2417** ms and has a higher throughput (8.473 Bytes/ms) for a data size of 20 KB. Compared to the other algorithms listed, such as RC4+ ECC+ SHA-256, AES+ Serpent+ ECC, LWC-ABE, LWC-AES, and PRESENT+TEA+ECC, the proposed system demonstrates lower latency values and higher throughput, indicating faster encryption and decryption of data.

In Table 4.14, the proposed system using the PRINCE+SPECK algorithm achieves an entropy value of 7.913 for a data size of 918 bytes. This entropy value is higher compared to the AES +Huffman and TEA

algorithms, demonstrating a higher degree of randomness and complexity in the encrypted data generated by the proposed system.

Table 4.13: comparison with other related work based on Average latency and throughput

Ref. No, Year	**Algorithms**	**data Size**	**Avg. latency**	**Throughput Byte/ms**
[96],2021	RC4, ECC, and SHA-256	20 KB	3194ms	6.412
[25],2022	AES+ Serpent + ECC	1 KB	134.96 ms.	7.587
[26],2022	PRESENT+TEA+ECC	20 KB	3274ms	6.255
[27],2022	LWC-ABE	100 Byte	257ms	3.891
[97],2022	LWC-AES	481 Byte	539ms	8.923
Proposed System	**PRINCE+SPECK**	**20 KB**	**2417ms**	**8.473**

Table 4.14: comparison with other related work based on entropy value

Ref. no	**algorithms**	**Data size**	**Entropy value**
[98],2019	AES+ Huffman	1868 Byte	7.864
[27],2022	TEA	1228 Byte	7.830
proposed	**PRINCE+SPECK**	**918 Byte**	**7.913**

Therefore, based on these comparisons, it can be concluded that the proposed system utilizing the PRINCE+SPECK algorithm offers improved performance in terms of both average latency for encrypting and decrypting data and higher entropy value for enhanced security.

4.6 Summary

This chapter demonstrates the implementation of the proposed system in an IoT healthcare application. There are two sensors used namely the temperature sensor, and the SPo2 sensor for the number of suspected patients, the management device (Raspberry pi), and the server for data evaluation. Firstly, the sensing data is generated from the sensor and collected in the Raspberry Pi. Secondly, the collected data is encrypted by the (PRINCE, SPECK) algorithms with the EllipticCurveDiffieHellman key exchange protocol and authentication process between IoT devices and server(cloud), and then sent to the server (cloud). In addition, this chapter explained the evaluation data after encryption and decryption data. The obtained results of the proposed system are a good value in terms of the performance evaluation parameters such as encryption and decryption time, throughput, and entropy.

CHAPTER FIVE

CONCLUSION AND SUGGESTIONS FOR FUTURE WORK

5.1 Conclusions

This chapter explains the proposed system conclusions and the main suggestions for future works they can be summarized as :

1. Increased Randomness: By increasing the randomness in the ciphertext, the system enhances its encryption strength. This makes it more challenging for attackers to extract the original text without the decryption key.
2. Suitable Coding Time: The system boasts a coding time that is deemed suitable compared to other hybrid coding methods. This implies that it strikes a balance between encryption strength and computational efficiency, ensuring a reasonable amount of time is required for encryption and decryption processes.
3. Data Integrity through Digital Signature: The system ensures data integrity by employing digital signatures. Digital signatures use cryptographic techniques to verify the authenticity and integrity of the data. By applying digital signatures, the system can detect any unauthorized modifications or tampering attempts.
4. Authorization Process via ECDH Protocol: The system employs the Elliptic Curve Diffie-Hellman (ECDH) protocol to establish a shared key during the authorization process. ECDH is a key exchange protocol that allows two parties to securely agree upon a shared secret key over an insecure channel. This shared key can then be used for symmetric encryption, ensuring confidentiality during data transmission.

These features collectively contribute to the proposed system's overall security and protection of sensitive health data.

5.2 Future Works

1. Evaluate the proposed system's resilience against various types of attacks. This assessment will help identify potential vulnerabilities and strengthen the system's security measures accordingly.
2. Explore the integration of lightweight encryption algorithms as a hybrid solution to further enhance the system's security. Evaluating different lightweight encryption algorithms and their effectiveness in the IoT environment can provide valuable insights for improving the overall security of the system.
3. Investigate alternative authentication and security approaches to complement the existing model. This could involve exploring advanced techniques such as biometric authentication, multi-factor authentication, or blockchain-based security mechanisms to strengthen the overall security posture of the system.
4. Implement compression techniques for IoT healthcare data to reduce data size and increase throughput. Compression algorithms can help optimize data transmission and storage, improving overall system efficiency and scalability. Evaluating different compression techniques and their impact on security and performance would be valuable for future enhancements.

These suggestions aim to expand the scope of the research and address potential areas of improvement for the proposed system.

References:

[1] 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.

[2] A. Whitmore, A. Agarwal, and L. Da Xu, “The Internet of Things—A survey of topics and trends,” *Inf. Syst. Front.*, vol. 17, no. 2, pp. 261–274, 2015, doi: 10.1007/s10796-014-9489-2.

[3] S. Surendran, A. Nassef, and B. D. Beheshti, “A survey of cryptographic algorithms for IoT devices,” *2018 IEEE Long Isl. Syst. Appl. Technol. Conf. LISAT 2018*, pp. 1–8, 2018, doi: 10.1109/LISAT.2018.8378034.

[4] A. H. Aly, A. Ghalwash, M. M. Nasr, and A. A. A. El-Hafez, “Formal security analysis of lightweight authenticated key agreement protocol for IoT in cloud computing,” *Indones. J. Electr. Eng. Comput. Sci.*, vol. 24, no. 1, pp. 621–636, 2021, doi: 10.11591/ijeecs.v24.i1.pp621-636.

[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] 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.

[7] D. N. Gupta and R. Kumar, “Lightweight cryptography: An IoT perspective,” *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 8, pp. 700–706, 2019.

[8] M. Alam, K. A. Shakil, and S. Khan, *Internet of things (IoT): Concepts and applications*, no. May 2020. 2020. doi: 10.1007/978-3-030-37468-6.

[9] S. Shakya, “A Perspective Review of Security Issues in IoT with Cloud Environment,” *J. ISMAC*, vol. 4, no. 2, pp. 84–93, 2022, doi: 10.36548/jismac.2022.2.002.

[10] P. Mell and T. Grance, “The NIST-National Institute of Standars and Technology- Definition of Cloud Computing,” *NIST Spec. Publ. 800-145*, pp. 1–3, 2011.

[11] S. Shilpashree, R. R. Patil, and C. Parvathi, “‘Cloud computing an overview,’” *Int. J. Eng. Technol.*, vol. 7, no. 4, pp. 2743–2746, 2018, doi: 10.14419/ijet.v7i4.10904.

[12] M. Manna and M. Ali Mohammed A, “Data Encryption Scheme for Large Data Scale in Cloud Computing,” *J. Telecommun. Electron. Comput. Eng.*, vol. 9, no. 2–12, pp. 1–5, 2017.

[13] 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.

[14] N. M. Abdulkareem, S. R. Zeebaree, M. A. M. Sadeeq, Di. M. Ahmed, A. S. Sami, and R. R. Zebari, “IoT and Cloud Computing Issues, Challenges and Opportunities: A Review,” *Qubahan Acad. J.*, vol. 1, no. 2, pp. 1–7, 2021, doi: 10.48161/issn.2709-8206.

[15] Y. L. Huang, C. R. Dai, F. Y. Leu, and L. You, “A secure data encryption Method employing a sequential-logic style mechanism for a cloud system,” *Int. J. Web Grid Serv.*, vol. 11, no. 1, pp. 102–124, 2015, doi: 10.1504/IJWGS.2015.067158.

[16] J. Yu, S. Liu, S. Wang, Y. Xiao, and B. Yan, “LH-ABSC: A Lightweight

Hybrid Attribute-Based Signcryption Scheme for Cloud-Fog-Assisted IoT," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 7949–7966, 2020, doi: 10.1109/JIOT.2020.2992288.

[17] F. Thabit, A. P. S. Alhomdy, A. H. A. Al-Ahdal, and P. D. S. Jagtap, "A new lightweight cryptographic algorithm for enhancing data security in cloud computing," *Glob. Transitions Proc.*, vol. 2, no. 1, pp. 91–99, 2021, doi: 10.1016/j.gltp.2021.01.013.

[18] A. I. Siam *et al.*, "Secure Health Monitoring Communication Systems Based on IoT and Cloud Computing for Medical Emergency Applications," *Comput. Intell. Neurosci.*, vol. 2021, 2021, doi: 10.1155/2021/8016525.

[19] M. Hussam, G. H. Abdul-majeed, and H. K. Hooomod, "New Lightweight Hybrid Encryption Algorithm for Cloud Computing (LMGHA-128bit) by using new 5-D hyperchaos system," *Turkish J. Comput. Math. Educ.*, vol. 12, no. 10, pp. 2531–2540, 2021.

[20] H. Abroshan, "A Hybrid Encryption Solution to Improve Cloud Computing Security using Symmetric and Asymmetric Cryptography Algorithms," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 6, pp. 31–37, 2021, doi: 10.14569/IJACSA.2021.0120604.

[21] P. Karuppusamy, I. Perikos, F. Shi, and T. N. Nguyen Editors, *Sustainable Communication Networks and Application Proceedings of ICSCN 2020*. 2021. [Online]. Available: http://www.springer.com/series/15362

[22] A. B. Jaber and M. E. Manaa, "A Robust Fog-Computing Security Approach for IoT Healthcare Applications," *Int. Conf. Commun. Inf. Technol. ICICT 2021*, pp. 174–179, 2021, doi: 10.1109/ICICT52195.2021.9568425.

[23] F. Thabit, O. Can, S. Alhomdy, G. H. Al-Gaphari, and S. Jagtap, “A Novel Effective Lightweight Homomorphic Cryptographic Algorithm for data security in cloud computing,” *Int. J. Intell. Networks*, vol. 3, no. April, pp. 16–30, 2022, doi: 10.1016/j.ijin.2022.04.001.

[24] S. Alexander Suresh and R. Jemima Priyadarsini, “Design of Maintaining Data Security on IoT Data Transferred Through IoT Gateway System to Cloud Storage,” *Int. J. Comput. Networks Appl.*, vol. 9, no. 1, pp. 135–149, 2022, doi: 10.22247/ijcna/2022/211632.

[25] S. Das and S. Namasudra, “A Novel Hybrid Encryption Method to Secure Healthcare Data in IoT-enabled Healthcare Infrastructure,” *Comput. Electr. Eng.*, vol. 101, no. July, p. 107991, 2022, doi: 10.1016/j.compeleceng.2022.107991.

[26] A. S. Kadhim, A. H. Alazam, and N. F. Sahib, “A hybrid lightweight security approach in internet of things for healthcare application,” *Bull. Electr. Eng. Informatics*, vol. 11, no. 6, pp. 3562–3569, 2022, doi: 10.11591/eei.v11i6.4417.

[27] M. Jammula, V. M. Vakamulla, and S. K. Kondoju, “Hybrid lightweight cryptography with attribute-based encryption standard for secure and scalable IoT system,” *Conn. Sci.*, vol. 34, no. 1, pp. 2431–2447, 2022, doi: 10.1080/09540091.2022.2124957.

[28] N. A. Khalil, “Proposed Encryption Algorithm for IoT Video Framing Stream Security in Building Monitoring,” 2022.

[29] A. Najah and K. Kareem, “Improving IoT Security using Data Compression and Lightweight Cryptographic Technique,” 2022.

[30] Y. Perwej, M. Ahmed, B. Kerim, and H. Ali, “An Extended Review on Internet of Things (IoT) and its Promising Applications,” *Commun. Appl. Electron.*, vol. 7, no. 26, pp. 8–22, 2019, doi: 10.5120/cae2019652812.

[31] 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.

[32] P. Sethi and S. R. Sarangi, “Internet of Things: Architectures, Protocols, and Applications,” *J. Electr. Comput. Eng.*, vol. 2017, 2017, doi: 10.1155/2017/9324035.

[33] T. Kramp, R. van Kranenburg, and S. Lange, *Introduction to the internet of things*. 2013. doi: 10.1007/978-3-642-40403-0_1.

[34] M. A. Ezechina, K. K. Okwara, and C. A. U. Ugboaja, “The Internet of Things (Iot): A Scalable Approach to Connecting Everything,” no. 2014, pp. 9–12, 2015, [Online]. Available: https://figshare.com/articles/The_Internet_of_Things_Iot_A_Scalable_Approach_to_Connecting_Everything/1329665

[35] N. M. Kumar and P. K. Mallick, “The Internet of Things: Insights into the building blocks, component interactions, and architecture layers,” *Procedia Comput. Sci.*, vol. 132, pp. 109–117, 2018, doi: 10.1016/j.procs.2018.05.170.

[36] P. R. Kumar, A. T. Wan, and W. S. H. Suhaili, “Exploring data security and privacy issues in internet of things based on five-layer architecture,” *Int. J. Commun. Networks Inf. Secur.*, vol. 12, no. 1, pp. 108–121, 2020, doi: 10.17762/ijcnis.v12i1.4345.

[37] A. Patel, P. Swaminarayan, and M. Patel, *Identification of Nutrition’s*

Deficiency in Plant and Prediction of Nutrition Requirement Using Image Processing, vol. 166. 2021. doi: 10.1007/978-981-15-9689-6_50.

[38] M. Burhan, R. A. Rehman, B. Khan, and B. S. Kim, "IoT elements, layered architectures and security issues: A comprehensive survey," *Sensors (Switzerland)*, vol. 18, no. 9, pp. 1–37, 2018, doi: 10.3390/s18092796.

[39] M. Kavre, A. Gadekar, and Y. Gadhade, "Internet of Things (IoT): A Survey," *2019 IEEE Pune Sect. Int. Conf. PuneCon 2019*, no. December, 2019, doi: 10.1109/PuneCon46936.2019.9105831.

[40] N. Almrezeq, L. Almadhoor, T. Alrasheed, A. A. A. El-Aziz, and S. Nashwan, "Design a secure IoT Architecture using Smart Wireless Networks," *Int. J. Commun. Networks Inf. Secur.*, vol. 12, no. 3, pp. 401–410, 2020, doi: 10.17762/ijcnis.v12i3.4877.

[41] A. J. Showail, "Internet of Things Security and Privacy," *Internet of Things*, vol. 2018, no. August, pp. 16–17, 2021, doi: 10.5772/intechopen.96669.

[42] V. Bhuvaneswari and R. Porkodi, "The internet of things (IOT) applications and communication enabling technology standards: An overview," *Proc. - 2014 Int. Conf. Intell. Comput. Appl. ICICA 2014*, no. March, pp. 324–329, 2014, doi: 10.1109/ICICA.2014.73.

[43] 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.

[44] 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-January, pp. 1347–1352, 2017, doi:

10.1109/CCAA.2017.8230007.

[45] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, “A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures,” *IEEE Access*, vol. 7, pp. 82721–82743, 2019, doi: 10.1109/ACCESS.2019.2924045.

[46] G. Lampropoulos, K. Siakas, and T. Anastasiadis, “Internet of Things in the Context of Industry 4.0: An Overview,” *Int. J. Entrep. Knowl.*, vol. 7, no. 1, pp. 4–19, 2019, doi: 10.2478/ijek-2019-0001.

[47] H. Rajab and T. Cinkelr, “IoT based Smart Cities,” *2018 Int. Symp. Networks, Comput. Commun. ISNCC 2018*, no. June, pp. 1–4, 2018, doi: 10.1109/ISNCC.2018.8530997.

[48] S. Pal, M. Hitchens, T. Rabehaja, and S. Mukhopadhyay, “Security requirements for the internet of things: A systematic approach,” *Sensors (Switzerland)*, vol. 20, no. 20, pp. 1–34, 2020, doi: 10.3390/s20205897.

[49] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, and E. K. Markakis, “A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches, and Open Issues,” *IEEE Commun. Surv. Tutorials*, vol. 22, no. 2, pp. 1191–1221, 2020, doi: 10.1109/COMST.2019.2962586.

[50] E. Vasilomanolakis, J. Daubert, M. Luthra, V. Gazis, A. Wiesmaier, and P. Kikiras, “On the Security and Privacy of Internet of Things Architectures and Systems,” *Proc. - 2015 Int. Work. Secur. Internet Things, SIoT 2015*, pp. 49–57, 2016, doi: 10.1109/SIOT.2015.9.

[51] 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.

[52] A. Mehrjou, R. Hosseini, and B. N. Araabi, “Ac Pt Us Cr Ac Pt Us Cr,” *Pattern Recognit. Lett.*, vol. 5, pp. 2–8, 2014, [Online]. Available: http://dx.doi.org/10.1016/j.patrec.2015.10.004

[53] Z. K. A. Mohammed and E. S. A. Ahmed, “World Scientific News WSN.,” *World Sci. News*, vol. 67, no. 2, pp. 126–148, 2017, [Online]. Available: http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.psjd-b638cb4d-d68f-4f4c-afa5-ad309a7c4838%0Ahttps://www.infona.pl/resource/bwmeta1.element.psjd-8c8e8b68-9180-4879-85d8-a7870d5644e9

[54] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan, “Internet of things (IoT) security: Current status, challenges and prospective measures,” *2015 10th Int. Conf. Internet Technol. Secur. Trans. ICITST 2015*, pp. 336–341, 2016, doi: 10.1109/ICITST.2015.7412116.

[55] M. M. Hossain, M. Fotouhi, and R. Hasan, “Towards an Analysis of Security Issues, Challenges, and Open Problems in the Internet of Things,” *Proc. - 2015 IEEE World Congr. Serv. Serv. 2015*, pp. 21–28, 2015, doi: 10.1109/SERVICES.2015.12.

[56] A. Khan, A. Ahmad, M. Ahmed, J. Sessa, and M. Anisetti, “Authorization schemes for internet of things: requirements, weaknesses, future challenges and trends,” *Complex Intell. Syst.*, vol. 8, no. 5, pp. 3919–3941, 2022, doi: 10.1007/s40747-022-00765-y.

[57] A. Mosenia and N. K. Jha, “A comprehensive study of security of internet-of-things,” *IEEE Trans. Emerg. Top. Comput.*, vol. 5, no. 4, pp. 586–602, 2017,

doi: 10.1109/TETC.2016.2606384.

[58] Z. K. Zhang, M. C. Y. Cho, C. W. Wang, C. W. Hsu, C. K. Chen, and S. Shieh, “IoT security: Ongoing challenges and research opportunities,” *Proc. - IEEE 7th Int. Conf. Serv. Comput. Appl. SOCA 2014*, pp. 230–234, 2014, doi: 10.1109/SOCA.2014.58.

[59] I. Ahmad, M. S. Niazy, R. A. Ziar, and S. Khan, “Survey on IoT: Security threats and applications,” *J. Robot. Control*, vol. 2, no. 1, pp. 42–46, 2021, doi: 10.18196/jrc.2150.

[60] H. I. Ahmed, A. A. Nasr, S. Abdel-Mageid, and H. K. Aslan, “A survey of IoT security threats and defenses,” *Int. J. Adv. Comput. Res.*, vol. 9, no. 45, pp. 325–350, 2019, doi: 10.19101/ijacr.2019.940116.

[61] P. Malhotra, Y. Singh, P. Anand, D. K. Bangotra, P. K. Singh, and W. C. Hong, “Internet of things: Evolution, concerns and security challenges,” *Sensors*, vol. 21, no. 5, pp. 1–35, 2021, doi: 10.3390/s21051809.

[62] Y. Jadeja and K. Modi, “Cloud computing - Concepts, architecture and challenges,” *2012 Int. Conf. Comput. Electron. Electr. Technol. ICCEET 2012*, pp. 877–880, 2012, doi: 10.1109/ICCEET.2012.6203873.

[63] V. Kale, “Cloud Computing Basics,” *Creat. Smart Enterp.*, no. June, pp. 141–171, 2017, doi: 10.1201/9781315152455-6.

[64] Z. Mahmood, “Cloud Computing: Characteristics and deployment approaches,” *Proc. - 11th IEEE Int. Conf. Comput. Inf. Technol. CIT 2011*, pp. 121–126, 2011, doi: 10.1109/CIT.2011.75.

[65] A. R. Nimodiya and S. S. Ajankar, “A Review on Internet of Things,” *Int. J.*

Adv. Res. Sci. Commun. Technol., vol. 113, no. 1, pp. 135–144, 2022, doi: 10.48175/ijarsct-2251.

[66] C. Stergiou, K. E. Psannis, B. G. Kim, and B. Gupta, "Secure integration of IoT and Cloud Computing," *Futur. Gener. Comput. Syst.*, vol. 78, pp. 964–975, 2018, doi: 10.1016/j.future.2016.11.031.

[67] A. Wicaksana and T. Rachman, *済無No Title No Title No Title*, vol. 3, no. 1. 2018. [Online]. Available: https://medium.com/@arifwicaksanaa/pengertian-use-case-a7e576e1b6bf

[68] 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.

[69] S. Chander, "Lightweight Cryptography Algorithms for Security of IoT Devices: A Survey," *Int. Res. J. Eng. Technol.*, no. June, pp. 842–850, 2022, [Online]. Available: www.irjet.net

[70] 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.

[71] 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.

[72] N. Arora and Y. Gigras, "Block and Stream Cipher Based Cryptographic

Algorithms: A Survey," *Int. J. Inf. Comput. Technol.*, vol. 4, no. 2, pp. 189–196, 2014.

[73] Z. A. Al-Odat, E. M. Al-Qtiemat, and S. U. Khan, "An efficient lightweight cryptography hash function for big data and IoT applications," *Proc. - 2020 IEEE Cloud Summit, Cloud Summit 2020*, pp. 66–71, 2020, doi: 10.1109/IEEECloudSummit48914.2020.00016.

[74] A. D. Dwivedi, P. Morawiecki, and G. Srivastava, "Differential Cryptanalysis of Round-Reduced SPECK Suitable for Internet of Things Devices," *IEEE Access*, vol. 7, pp. 16476–16486, 2019, doi: 10.1109/ACCESS.2019.2894337.

[75] L. Sleem and R. Couturier, "Speck-R: An ultra light-weight cryptographic scheme for Internet of Things," *Multimed. Tools Appl.*, vol. 80, no. 11, pp. 17067–17102, 2021, doi: 10.1007/s11042-020-09625-8.

[76] A. Alkamil and D. G. Perera, "Towards Dynamic and Partial Reconfigurable Hardware Architectures for Cryptographic Algorithms on Embedded Devices," *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.3043750.

[77] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "Implementation and Performance of the Simon and Speck Lightweight Block Ciphers on ASICs," *Unpubl. Work*, 2014.

[78] Q. Zhang, H. Zhang, X. Cui, X. Fang, and X. Wang, "Side Channel Analysis of SPECK Based on Transfer Learning," *Sensors*, vol. 22, no. 13, 2022, doi: 10.3390/s22134671.

[79] M. AbdulRaheem *et al., An Enhanced Lightweight Speck System for Cloud-Based Smart Healthcare*, vol. 1455 CCIS, no. November. Springer International Publishing, 2021. doi: 10.1007/978-3-030-89654-6_26.

[80] J. Borghoff *et al.*, “PRINCE – A Low-Latency Block Cipher Extended Abstract,” *Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, no. 11061130539, pp. 208–225, 2012.

[81] R. Posteuca, C. L. Duta, and G. Negara, “New approaches for round-reduced prince cipher cryptanalysis,” *Proc. Rom. Acad. Ser. A - Math. Phys. Tech. Sci. Inf. Sci.*, vol. 16, pp. 253–264, 2015.

[82] A. A. Abdullah and N. R. Obeid, “Efficient Implementation for PRINCE Algorithm in FPGA Based on the BB84 Protocol,” *J. Phys. Conf. Ser.*, vol. 1818, no. 1, 2021, doi: 10.1088/1742-6596/1818/1/012216.

[83] A. Taparia, S. K. Panigrahy, and S. K. Jena, “Secure key exchange using enhanced Diffie-Hellman protocol based on string comparison,” *Proc. 2017 Int. Conf. Wirel. Commun. Signal Process. Networking, WiSPNET 2017*, vol. 2018-Janua, pp. 722–726, 2018, doi: 10.1109/WiSPNET.2017.8299856.

[84] 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.

[85] R. Haakegaard and J. Lang, “The elliptic curve diffie-hellman (ECDH),” *Retrieved Febr. 10, 2020, from http//koclab.cs.ucsb.edu/teaching/ecc/project/2015Projects/Haakegaard+Lang.pdf*, no. December, p. 4, 2015.

[86] J. Katz and Y. Lindell, “Message Authentication Codes,” *Introd. to Mod. Cryptogr.*, pp. 127–172, 2020, doi: 10.1201/b17668-9.

[87] E. T. Krovetz, “UMAC: Message authentication code using universal hashing’,” *קתדרה*, vol. 122, no. 1995, pp. 25–27, 2006.

[88] H. Tiwari, “Merkle-Damgård construction method and alternatives: A review,” *J. Inf. Organ. Sci.*, vol. 41, no. 2, pp. 283–304, 2017, doi: 10.31341/jios.41.2.9.

[89] M. A. Kale and P. S. Dhamdhere, “Survey Paper on Different Type of Hashing Algorithm,” *Int. J. Adv. Sci. Res.*, vol. 3, no. 2, pp. 14–16, 2018.

[90] S. Al-Kuwari, J. H. Davenport, and R. J. Bradford, “Cryptographic hash functions: recent design trends and security notions,” *Short Pap. Proc. 6th China Int. Conf. Inf. Secur. Cryptol. (Inscrypt ’10)*, pp. 133–150, 2010, [Online]. Available: https://ia.cr/2011/565%0Ahttps://researchportal.bath.ac.uk/en/publications/cryptographic-hash-functions-recent-design-trends-and-security-no%0Ahttps://purehost.bath.ac.uk/ws/portalfiles/portal/309274/HashFunction_Survey_FINAL_221011-1.pdf

[91] O. ÖzEN, “Design and Analysis of Multi-Block-Length Hash Functions,” vol. 5333, 2012, [Online]. Available: http://infoscience.epfl.ch/record/174698/files/EPFL_TH5333.pdf

[92] U. K. Chowdhury, A., & Ray, “Keccak_Published_RP.pdf.”

[93] A. N. Kadhim and M. E. 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.

[94] D. B. Stewart, “Measuring Execution Time and Real-Time Performance,” *Embed. Syst. Conf. (ESC SF)*, no. September, pp. 1–15, 2002.

[95] G. A. A.-R. Abdulrazzaq H. A. Al-Ahdal and and N. K. Deshmukh, “Sustainable Communication Networks and Application Proceedings of

ICSCN 2020," 2021.

[96] 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.

[97] H. M. Mohammad and A. A. Abdullah, "Enhancement process of AES: a lightweight cryptography algorithm-AES for constrained devices," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 20, no. 3, pp. 551–560, 2022, doi: 10.12928/TELKOMNIKA.v20i3.23297.

[98] 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.

الخلاصة

إن إنترنت الأشياء (IoT) هو مجال سريع النمو يتضمن توصيل مجموعة متنوعة من الأجهزة وأجهزة الاستشعار بالإنترنت لجمع البيانات ومشاركتها. تتطلب الكمية الهائلة من البيانات التي تم إنشاؤها بواسطة هذه الأجهزة تخزينًا ومعالجة آمنة ، وهنا يأتي دور الحوسبة السحابية. تعد الحوسبة السحابية حلاً فعالاً لتخزين بيانات إنترنت الأشياء ، ولكن كمية كبيرة من البيانات معرضة أيضًا للتهديدات الأمنية وزمن انتقال عالٍ عندما تنتقل من أجهزة إنترنت الأشياء إلى السحابة, لذلك يجب توفير مستوى عالٍ من الأمان للبيانات لتأمينها من التهديدات.

يهدف العمل المقترح إلى معالجة المشكلات الأمنية في أنظمة الحوسبة السحابية لإنترنت الأشياء من خلال تنفيذ مرحلة المصادقة وآليات التشفير بين أجهزة إنترنت الأشياء والسحابة. يستخدم النظام خوارزميات تشفير هجينة خفيفة الوزن(PRINCE و SPECK) لتشفير بيانات المستشعر باستخدام مفتاح سري مشترك تم إنشاؤه من خلال بروتوكول Elliptic Curve Diffie-Hellman (ECDH) من جانب العميل. بالإضافة إلى ذلك، يتم استخدام خوارزمية التجزئة لمصادقة البيانات المشفرة، وضمان صحتها وسلامتها. على جانب الخادم، يتم تنفيذ عمليات فك التشفير باستخدام نفس خوارزميات التشفير. قام النظام المقترح بتطبيق مستشعر درجة الحرارة ومستشعر SPO2 المتصلين بجسم الإنسان. يُستخدم مستشعر درجة الحرارة لمراقبة درجة حرارة الجسم، بينما يُستخدم مستشعر SPO2 لمراقبة معدل ضربات القلب ومستويات الأكسجين. ولجعل كل شيء يعمل، استخدمنا نموذج Raspberry Pi 3 B+ كمحور مركزي. بشكل عام، أثبت هذا النظام فعاليته في مساعدتنا على تتبع المقاييس الصحية المهمة.

تشير نتائج النظام المقترح إلى تحسين وقت التشفير، حيث بلغ متوسط وقت التشفير 512 مللي ثانية لحجم بيانات يبلغ 10 كيلو بايت. يعتمد تقييم القوة الأمنية للنظام على الإنتروبيا، التي تقيس العشوائية وتوفر مستوى عالٍ من الأمان للبيانات. يحقق النظام المقترح قيمة إنتروبيا تبلغ 7.999 لحجم بيانات يبلغ 10 كيلو بايت، مما يشير إلى مستوى عالٍ من الأمان.

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

تحسين أسلوب التشفير الخفيف الوزن لدفق البيانات في الوقت الفعلي باستخدام الحوسبة السحابية

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

الحصول على درجة الماجستير في تكنولوجيا المعلومات / شبكات المعلومات

من قبل الطالبة

انعام غانم هلال

باشراف

أ.م.د مهدي عبادي مانع مهدي

هـ 1445 م 2023