Design of an Ultra-Light Weight Cryptographic algorithm for heterogeneous environment in the Internet of Things

Submitted in partial fulfilment of the requirements

for the award of the degree of

Doctor of Philosophy

by

Mounika Jammula

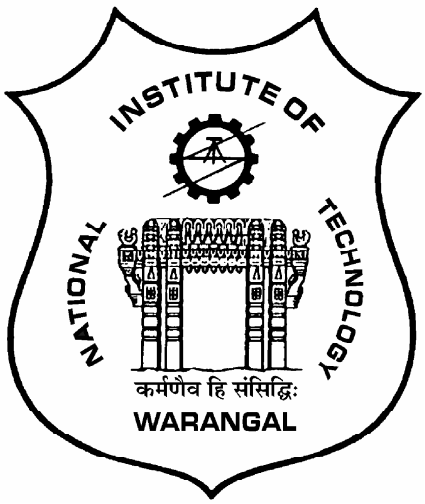
(Roll No: 701946)

Under the supervision of

Prof. V. Venkata Mani

&

Dr.K.Sai Krishna



Department of Electronics & Communication Engineering

National Institute of Technology Warangal

Telangana, India - 506004

2023

Dedicated

To

My Mother

Approval Sheet

This thesis entitled **Design of an Ultra-Light Weight Cryptographic algorithm for heterogeneous environment in the Internet of Things** by **Mounika Jammula** is approved for the degree of **Doctor of Philosophy**.

Examiners

Research Supervisor

Prof. V. Venkata Mani
Department of ECE
NIT Warangal, India-506004

Research Co-Supervisor

Dr. K. Sai Krishna
Department of ECE
CBIT, Hyderabad, India-500075

Chairman & Head

Prof. P. Sreehari Rao
Department of ECE
NIT Warangal, India-506004

Place:

Date:

Declaration

This is to certify that the work presented in this thesis entitled **Design of an Ultra-Light Weight Cryptographic Algorithm for Heterogeneous Environment in the Internet of Things** is a bonafied work done by me under the supervision of **Prof. V. Venkata Mani** & **Dr. K. Sai Krishna** and was not submitted elsewhere for the award of any degree.

I declare that this written submission represents my own ideas and even considered others ideas which are adequately cited and further referenced the original sources. I understand that any violation of the above will cause disciplinary action by the institute and can also evoke panel action from the sources or from whom proper permission has not been taken when needed. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea or data or fact or source in my submission.

Place:

Date:

Mounika Jammula
Research Scholar
Roll No.: 701946

NATIONAL INSTITUTE OF TECHNOLOGY

WARANGAL, INDIA-506004

Department of Electronics & Communication Engineering

CERTIFICATE

This is to certify that the thesis work entitled is a bonafide record of work carried out by **Mounika Jammula** submitted to the faculty of **Electronics & Communication Engineering** department, in partial fulfilment of the requirements for the award of the degree of **Doctor of Philosophy** in **Electronics and Communication Engineering, National Institute of Technology Warangal, India-506004**. The contributions embodied in this thesis have not been submitted to any other university or institute for the award of any degree.

Dr. V. Venkata Mani
Research Supervisor
Professor
Department of ECE
NIT Warangal, India-506 004.

Dr. K. Sai Krishna
Research Co-Supervisor
Assistant Professor
Department of ECE
CBIT, India-500 075.

Place:

Date:

Acknowledgements

Words cannot express my gratitude to my guide, Prof. V. Venkata Mani, for her invaluable patience, feedback, and continuous support during my entire PhD journey. Her guidance has been instrumental in shaping the trajectory of my research and personal growth. I am profoundly thankful to Dr. K. Sai Krishna for his unwavering support, from the moment I embarked on my academic journey at NIT Warangal. His encouragement and guidance have been pivotal in helping me navigate through the challenges and triumphs of this pursuit.

My heartfelt appreciation goes to my esteemed DSC members, Prof. P. Sreehari Rao (Chairman, HOD of ECE), Dr. A Prakasa Rao (Member-Internal-ECE), and Dr. P. Muralidhar (Member-Internal-ECE), for their generous sharing of knowledge and expertise. Their insightful inputs and feedback have significantly enriched my research endeavor.

I extend my gratitude to my co-scholars, whose interactions and camaraderie have been a source of inspiration and support. Their willingness to collaborate, share ideas, and provide moral encouragement have made this academic journey truly fulfilling. My sincere thanks go to the entire office staff of the Department of ECE for their seamless administrative support, which facilitated the smooth execution of my academic pursuits.

Lastly, I would like to express my heartfelt appreciation to my family, especially my parents and spouse. Their unwavering belief in me has been a constant source of strength, motivating me to persevere through challenges and stay focused on my goals.

Mounika Jammula

Abstract

The Internet of Things (IoT) has revolutionized our lives by creating a smart infrastructure using various devices capable of self-organization. However, this interconnected network raises concerns about data privacy and protection. Moreover, the limited resources and battery power of IoT devices necessitate developing resource-optimized and secure solutions. To address these issues, this research proposes an integrated communication protocol using symmetric key-based cryptography and a Deep Learning Convolutional Neural Network (DLCNN) for predicting normal and attacked data. The logical map generates the symmetric keys, ensuring resistance against key reset and device capture attacks, resulting in an Ultra-Lightweight Communication (ULWC) protocol with improved attack detection parameters.

In addition to the communication protocol, this work focuses on enhancing IoT security through Lightweight Cryptography-based Attribute-Based Encryption (LWC-ABE) method. The proposed LWC-ABE method reduces the reliance on multiple trusted authority environments, which can be bottlenecks in IoT servers and devices. It offers high expressiveness, access policy updates, large attribute domains, and white box traceability properties. Simulation results demonstrate that the proposed LWC-ABE method outperforms conventional approaches, with reduced encryption and decryption times for multi-users and different message sizes.

Finally, Lightweight-Medical Image Cryptography (LW-MIC) system was developed using ELWC protocols. The medical image data is first converted into digital format, and then ELWC operations, employing Play-Fair and Cha-Cha based encryption algorithms, are applied to the vector data. This ensures that the secured image data transmitted over the Internet of Medical Things (IoMT) environment remains protected. At the receiver's end (doctor), the ELWC decryption algorithms restore the original image data.

The simulation results for the proposed ULWC, LWC-ABE, and LW-MIC protocols indicate superior security performance compared to state-of-the-art methods. Additionally, these solutions demonstrate reduced time complexity, making them efficient and effective for securing IoT environments.

Contents

Declaration iii

Acknowledgements v

Abstract vi

List of Figures xii

List of Tables xiv

List of Abbreviations xv

1 Introduction 1

1.1 Overview 1

1.2 Research Motivation 2

1.3 Problem Statement 3

1.4 Research Objectives 4

1.5 Organization of the Thesis 5

2 Review of Literature 7

2.1 Introduction 7

2.2 Types of Cryptography 8

2.2.1 Public Key Cryptography 9
2.2.2 Secret Key Cryptography 18
2.2.3 Hash Function Cryptography 39
2.2.4 Other Cryptographic Methods 43
2.3 LWC 51
2.4 Applications of LWC 55
2.5 Performance Metrics 56
2.6 Summary 61

3 ULWC Protocol with DLCNN for Heterogenous IoT Environment 63

3.1 Introduction 63
3.2 DLCNN-based ULWC for IoT Environment 67
3.2.1 DLCNN Model 69
3.3 Protocol Design 71
3.3.1 Initialization 73
3.3.2 Establish Session Key 75
3.3.3 Communication Between Devices 75
3.3.4 Key Delegation 77
3.4 Results and Discussion 78
3.4.1 Dataset 79
3.4.2 Influence on the ADT 80
3.4.3 Impact on Encryption and Decryption time 81
3.4.4 Impact of attack detection performance 82
3.5 Summary 83

4 Hybrid LWC with Attribute-Based Encryption for secure and scalable IoT system 85

4.1 Introduction . . . 85

4.2 LWC-ABE Method . . . 87

4.2.1 ABE . . . 90

4.2.2 ChaCha Encryption . . . 91

4.2.3 Privileged Encryption . . . 95

4.3 Results and Discussion . . . 97

4.3.1 Influence on the Amount of time required for Encryption and Decryption . . . 98

4.4 Summary . . . 101

5 Secure And Scalable IoMT Using Ensemble LWC Model 103

5.1 Introduction . . . 103

5.2 Proposed Method . . . 105

5.2.1 ChaCha Encryption and Decryption . . . 105

5.2.2 Playfair encryption . . . 106

5.3 Results and Discussion . . . 109

5.3.1 Subjective performance . . . 109

5.3.2 Objective performance . . . 109

5.4 Summary . . . 110

6 Conclusion and Future Scope 112

6.1 Conclusion . . . 112

6.2 Future Scope . . . 114

Publications 116

Bibliography **117**

List of Figures

2.1 Types of cryptography 8

2.2 PKC block diagram 10

2.3 RSA encryption block diagram 14

2.4 Diffie Hellman encryption 16

2.5 ECC block diagram 17

2.6 SKC block diagram 19

2.7 SCC block diagram 21

2.8 RC4 encryption block diagram 24

2.9 Salsa20 encryption block diagram 25

2.10 Grain-128 encryption block diagram 27

2.11 BCC block diagram 29

2.12 AES block diagram 34

2.13 3DES block diagram 35

2.14 Blowfish block diagram 37

2.15 CAST-128 block diagram 39

2.16 HFC block diagram 40

2.17 Quantum cryptography block diagram 44

2.18 Homomorphic encryption block diagram 47

2.19 Obfuscation-based Cryptography block diagram 49

2.20 Design trade-offs for LWC 53

3.1 Proposed IoT network model 68

3.2 Proposed DLCNN architecture 71

3.3 Proposed ULWC protocol 72

3.4 Analysis of the encryption time for ten users 82

3.5 Analysis of the decryption times for ten users 82

3.6 Attack detection Performance comparison of various LWC models 84

4.1 Model for the LWC-ABE system 89

4.2 Input forms, (a) Zigzag form, (b) Alternative form 93

4.3 Polybius Square, (a) row-based ciphertext generation, (b) column-based ciphertext generation, (c) horizontal opposite corner-based ciphertext generation 96

4.4 Analysis of encryption time for ten users 99

4.5 Analysis of decryption times for ten users 100

5.1 Proposed LW-MIC block diagram 106

5.2 Subjective performance of the proposed method (a) Source images (b) Decrypted images using existing HSO-KG (c)Decrypted images using proposed LW-MIC. 110

List of Tables

3.1 Initialization algorithm 74

3.2 Session Key configuration Algorithm 76

3.3 Communication Protocol Algorithm 77

3.4 ADT comparision of various LWC methods 81

3.5 Comparison of encryption and decryption times based on message sizes . . 83

4.1 Initialization algorithm 94

4.2 QRF Algorithm 94

4.3 Performance comparison of encryption and decryption times 98

4.4 Performance of encryption time analysis for ten users. 100

4.5 Performance of decryption time analysis for ten users. 101

4.6 Comparison of encryption and decryption timings dependent on message size101

5.1 Proposed ChaCha image encryption algorithm 107

5.2 Image quality performance comparison of various methods 111

5.3 Time complexity (in seconds) performance of various methods 111

List of Abbreviations

ADT Attack detection time

AES Advanced Encryption Standard

ASG Alternating Step Generator

BCC Block Cipher Cryptography

BCDT Block Cipher Decryption Time

CP-ABE Ciphertext-Policy based ABE

CSP Cloud Storage Providers

CP-ABE Ciphertext-Policy based ABE

D2C Device-to-Control-Center

D2D Device-to-Device

DNN Deep Neural Networks

DM-ABE Dual Membership-based ABE

DSA Digital Signature Algorithm

DTVMS Decryption time measured for various message sizes

ECDSA Elliptic Curve Digital Signature Algorithm

ECC Elliptic Curve Cryptography

EEE Encrypt-Encrypt-Encrypt

EDE Encrypt-Decrypt-Encrypt

EHR Electronic Health Record

EHR Electronic Health Records

ELWC Ensemble LWC

ETVMS Encryption time measured for various message sizes

FLDSOP Fractional Lorenz-Duffing Chaotic System

FP False Positives

FN False Negatives

GFS Generalized Feistel Structure

GID User's ID

HFC Hash Function-Based Cryptography

HMAC Hash-Based Message Authentication Codes

HIPAA Health Insurance Portability and Accountability Act

HOTP HMAC-Based One-Time Password

HSO-KG Hybrid Swarm Optimization-Based Key Generation

ICT Information and Communication Technology

ID Identification Number

IEC International Electrotechnical Commission

IoMT Internet of Medical Things

IoT Internet of Things

ISO International Organization for Standardization

KST Key Setup Time

LFSR Linear Feedback Shift Registers

LWC Light Weight Cryptography

LWC-ABE LWC-based Attribute-Based Encryption

LWE Learning with Errors

LW-MIC Lightweight-Medical Image Cryptography

LWSE Light Weight Selective Encryption

MD5 Message Digest 5

MCP-ABE Modified CP-ABE

NIST National Institute of Standards and Technology

PPT Post-Processing Time

PBKDF2 Password-Based Key Derivation Function 2

PERAP Privacy-Enhanced Robust Authentication Protocol

PH-ABKS-DS Policy-Hiding Attribute-Based Keyword Search and Data Sharing Scheme

PKC Public Key Cryptography

PKI Public Key Infrastructure

PPT Post-Processing Time

PRGA Pseudo-Random Generation Algorithm

PRNG Pseudorandom Number Generator

PSNR Peak Signal-to-Noise Ratio

QKD Quantum Key Distribution

QRF Quarter Round Function

RABE-DI Revocable Attribute-based Encryption with Data Integrity

RC5 Rivest Cipher5

RFID Radio Frequency Identification Device

RLWE Ring Learning with Errors

RMA-ABE Reversible Multi-Authority-Based ABE

RSA Rivest-Shamir-Adleman

RSA-PSS RSA Public-key Cryptography Standard

RTL Register Transfer Level

SCC Stream Cipher Cryptography

SDM Secure Decision of Membership

SHA Secure Hash Function

SKC Symmetric Key Cryptography

SPN Substitution-Permutation Network

SSIM Structural Similarity Index

TLS Transport Layer Security

TN True Negatives

TP True Positives

UER-ABE Unbounded and Efficient Revocable based ABE

ULWC Ultra–Lightweight Cryptography

Chapter 1

Introduction

1.1 Overview

Light Weight Cryptography (LWC) is a classification of cryptographic methods and algorithms intended for use on resource-constrained devices such as smart cards, sensors, and embedded systems [1]. Because of their limited processing power, memory, and energy, these devices need cryptographic solutions that are both efficient and have low overhead to offer a sufficient degree of security [2]. The need for LWC emerged in the 1990s when smart cards and other small, embedded devices became increasingly popular. Many applications, such as payment systems, authentication, and access control, use these devices at various times. However, classic cryptographic methods such as Elliptic Curve Cryptography (ECC), Advanced Encryption Standard (AES) [3] and Rivest-Shamir-Adleman (RSA) encryption [4] were not suitable for these devices because of their high computational and memory requirements. As a result, researchers started developing new cryptographic approaches and algorithms tailored specifically for devices with lower power consumption. In the early days of LWC, researchers focused on developing new lightweight encryption algorithms such as Rivest Cipher5 (RC5) [5], Skipjack [6], and so on. These algorithms were designed to be computationally efficient and to have low memory requirements. However, they lacked some of the security features of traditional cryptographic algorithms. In the 2000s, several standardization bodies began to develop standards for LWC. The ISO/IEC 18033-3 standard [7] for lightweight block ciphers and the ISO/IEC 29192 standard [8] for LWC for financial transactions are the two standards

considered to be the most significant among these standards. The International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) were accountable for developing both standards. In recent years, researchers have focused much of their emphasis on the development of innovative LWC techniques and algorithms that are specialized for specific applications such as the Internet of Things (IoT), Internet of Medical Things (IoMT) [9], wearable devices [10], and cyber-physical systems [11].

1.2 Research Motivation

LWC primary focus revolves around the research, design, and advancement of cryptographic algorithms specifically engineered to operate effectively in settings with notable limitations and constraints on available resources, such as resource-constrained and limited size. There is a growing need for safe communication and data security in low-power devices like smart cards, Radio Frequency Identification Device (RFID) tags, and wireless sensor networks because of the development of low-power devices like these. However, traditional cryptographic algorithms are often too resource-intensive for these devices, making them unsuitable for implementation. To overcome this obstacle, LWC is working to build algorithms that are effective enough to be applied to such devices. With the advent of quantum computers, traditional cryptographic algorithms currently considered secure income are vulnerable to attacks. LWC algorithms are designed to resist quantum attacks, making them an attractive option for post-quantum cryptography. The IoMT has emerged as a critical area of research in recent years, potentially revolutionizing healthcare delivery and patient outcomes. IoMT devices are connected to the internet, allowing healthcare professionals to monitor patients remotely, collect real-time data, and improve treatment outcomes. However, using IoMT devices presents several challenges, including security, privacy, and energy consumption. One way to address these challenges is by using LWC. It is a type of cryptography specifically designed for use in resource-constrained environments, such as IoMT devices. The compact code size, low computational complexity, and low memory need characteristics of LWC algorithms to make them well-suited for devices with limited processing power and memory. The use of LWC in IoMT devices has several benefits. First, it can enhance the security of these devices, protecting sensitive

patient data from malicious attacks. Second, it can reduce the energy consumption of IoMT devices, extending their battery life and reducing the need for frequent battery replacements. Finally, it can improve the privacy of IoMT devices by reducing the amount of data that needs to be transmitted over the internet. Given the potential benefits of LWC for IoMT applications, there is a growing interest in this area of research. Researchers are exploring different LWC algorithms, their performance characteristics, and their suitability for different types of IoMT devices. Additionally, there is a need for standards and guidelines for using LWC in IoMT devices to ensure they are secure and reliable. So, the motivation for using LWC in IoMT applications is driven by the need to address the unique challenges posed by these devices, including security, privacy, and energy consumption. Using LWC, researchers and healthcare professionals can ensure that IoMT devices are secure, efficient, and effective, improving patient outcomes and healthcare delivery.

1.3 Problem Statement

The problem statement for LWC is to design cryptographic algorithms that are efficient, compact, and appropriate for implementation in resource-constrained situations while preserving a sufficient degree of security. The cryptographic algorithms used in LWC offer an appropriate degree of security against attacks is the primary obstacle that must be overcome in this protocol. It is necessary to assess the robustness of the security provided by LWC algorithms against various attack types, including side-channel attacks, fault attacks, and cryptanalysis. Finding a perfect medium between safety and productivity is the obstacle that must be overcome. The cryptographic algorithms employed in LWC must be very efficient in terms of the computations they make and the amount of memory and power they require. The problem is building algorithms that offer sufficient protection while reducing the processing resources needed for execution. To ensure interoperability and widespread adoption, LWC algorithms need to be standardized. The challenge is to develop a consensus on the best LWC algorithms and ensure that they are implemented consistently across various devices and platforms. LWC algorithms must be implemented on various platforms, including low-power, IoMT, and embedded systems.

The difficulty is in the creation of algorithms that are not only capable of being effectively implemented on a variety of platforms but also resistant to implementation-based attacks such as side-channel attacks. IoMT devices often have limited resources, including processing power, memory, and battery capacity, among other things. The challenge is to develop LWC algorithms that are lightweight and efficient enough to be implemented on these devices without compromising security. Confidentiality collects and transmits sensitive medical data, making data privacy and confidentiality a critical concern. The difficulty is developing LWC algorithms that can offer robust encryption and authentication procedures to prevent illegal access and protect data from being compromised. IoMT devices are susceptible to cyber-attacks, including hacking, malicious software distribution, and denial-of-service attacks. The challenge is developing LWC algorithms that can provide robust security against these threats, including side-channel and fault attacks. The IoMT devices need to be interoperable with existing medical systems and standards. The problem is to build LWC algorithms that can interact with current cryptographic protocols and federal laws and regulations, including the Health Insurance Portability and Accountability Act (HIPAA) and Federal Information Processing Standards, without sacrificing the level of security or the efficiency they provide. The IoMT devices are subject to various regulatory requirements, including privacy and security regulations such as General Data Protection Regulation and HIPAA. The challenge is to develop LWC algorithms that comply with these regulations while maintaining the necessary level of security and efficiency.

1.4 Research Objectives

The following research objectives are defined to solve the problems presented in the existing systems.

- Design of Ultra–Lightweight Cryptography (ULWC) for heterogenous IoT environments using **DLCNN!** (**DLCNN!**). The DLCNN plays a crucial role in predicting both normal and attacked data based on the input requested data, thereby fortifying the overall security measures. Because it is resistant to attacks involving key reset and device acquisition, the logistic map is used here to generate symmetric keys.

- Development of LWC-based Attribute-Based Encryption (LWC-ABE) utilizing ChaCha and Playfair as the encryption algorithms. The LWC-ABE operates within a multiple trusted authority environment, which serves as the trusted authority. This unique environment challenges IoT servers and IoT devices as they seek to modify their access policies.

- Development of Lightweight-Medical Image Cryptography (LW-MIC) system using Ensemble LWC (ELWC) protocols. Initially, the medical image data from users is converted into digital data. Then, ELWC operation is applied to vector data, which implements play-fair and Cha-Cha-based encryption algorithms. So, secured image data of users are transmitted over the IoMT environment. Finally, the ELWC decryption algorithms restore the original image data at the receiver (doctor) side.

1.5 Organization of the Thesis

This thesis is organized as follows

Chapter 1 provides an in-depth analysis of various cryptographic methods, establishing the foundation for the research. It defines research problems and motivations by exploring conventional cryptography methods. Furthermore, this chapter sets research objectives aimed at addressing these research problems.

Chapter 2 focuses on encryption methods and their hybrid combinations. The LWC method is introduced, and its advantages over other encryption methods are discussed. Various performance metrics are presented to highlight the superiority of the research work.

Chapter 3 details the implementation process of the DLCNN based ULWC protocol, specifically designed for resource constrained applications.

Chapter 4 discusses the LWC-ABE method, utilizing Cha-Cha and Playfair encryptions. Additionally, this chapter includes a comprehensive performance comparison of the proposed method.

Chapter 5 is dedicated to the LW-MIC system, using ELWC protocols, particularly

for IoMT applications.

Chapter 6 serves as the conclusion of the research, summarizing the key findings and contributions made throughout the thesis. It also discusses potential avenues for future research.

Chapter 2

Review of Literature

2.1 Introduction

Encryption [12] and cryptography [13] are two closely similar ideas that are sometimes confused, even though they are not the same thing. Cryptography, a field encompassing the study and practical application of secure communication, employs various methods, including encryption. Encryption serves as a fundamental technique within cryptography, enabling the transformation of plaintext into ciphertext to ensure confidentiality and privacy. It involves utilizing cryptographic algorithms and keys to encode information so that it becomes unintelligible to unauthorized individuals or third parties who may intercept the communication. Encryption offers a method for protecting sensitive data and enabling secure communication even in the face of possible adversaries or eavesdroppers. This was accomplished via the use of ciphers. The study and practice of keeping one's communications private when others are present is referred to as cryptography. The following is an in-depth comparison of two closely related concepts: cryptography and encryption. The study of methods for secure communication, such as encrypting and decrypting messages and authenticating senders and recipients, is known as cryptography. The process of transforming plain text into ciphertext by using an encryption method is what we refer to as encryption. Cryptography aims to provide secure communication between two or more parties, even in the presence of third parties who may try to intercept, modify, or block the communication [14]. Encryption is one of the primary methods used in the field of cryptography.

It is intended to protect the secrecy and privacy of the information conveyed. Cryptography uses various techniques such as encryption, decryption, hashing, digital signatures, and key exchange algorithms. Encryption and decryption are one of the most used techniques in cryptography. Cryptography has various applications, including securing online transactions, sensitive information [15], communication between two or more parties, networks, and data at rest.

2.2 Types of Cryptography

Cryptography is safeguarding communication to prevent unauthorized access by other parties. It includes transforming plaintext (unencrypted data) into ciphertext (encrypted data) to transmit over the internet or other communication channels securely. Figure 2.1 [16] illustrates the different kinds of cryptographic methods. There are many types of cryptography, including Public Key Cryptography (PKC), Symmetric Key Cryptography (SKC) and Hash Function-based Cryptography.

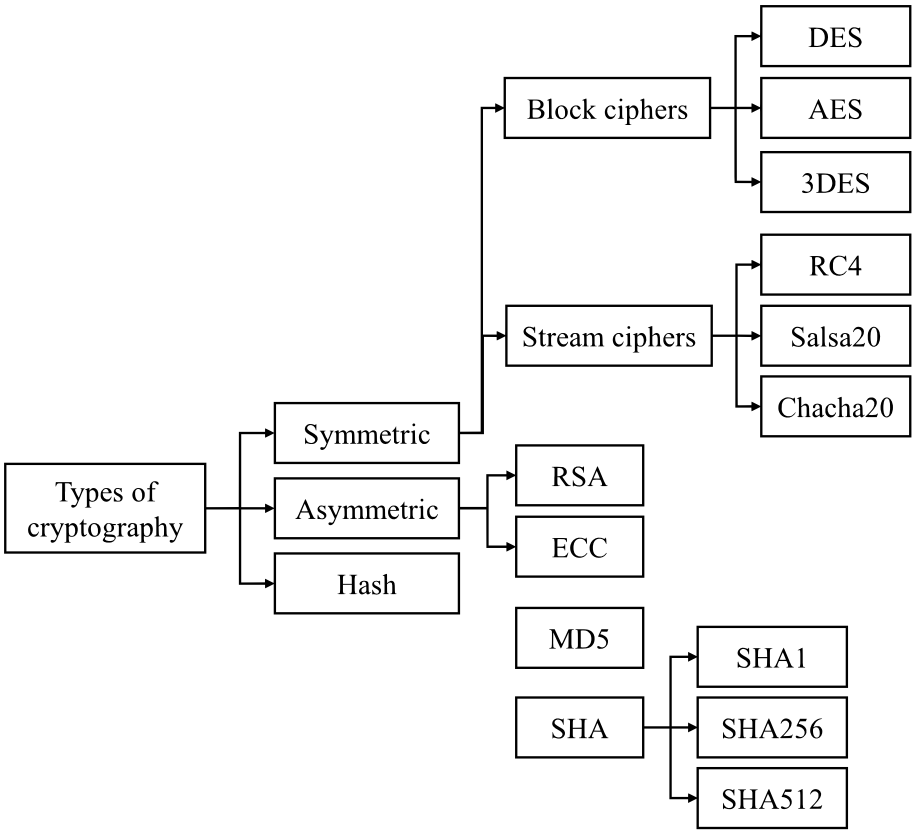


Figure 2.1: Types of cryptography

2.2.1 Public Key Cryptography

PKC is also known as asymmetric cryptography, which secures sensitive information by using two unique keys, a public key and a private key, instead of only a single one alone. The public key is widely distributed and accessible to anyone, while the private key remains securely held by the intended recipient or owner. This technique is extensively used in many forms of digital communication, including but not limited to email, online banking, and online shopping, to ensure that information sent over the internet is both safe and secret [16]. When a sender wishes to communicate with a receiver, they must first encrypt the message using the recipient's public key before sending it. After that, the recipient decrypts the communication using their private key. This restricts access to the message so that it can only be viewed by the designated recipient. The PKC structural components are shown in Figure 2.2. Since a recent development, inferring K from knowledge of K-1 is no longer possible or vice versa. This enables the primary use of technology based on public keys, in which one key is disclosed to the public while the other is kept a secret.

This offers a far higher level of functionality, expanding the use of cryptography to enable authentication and integrity and the secrecy of the encrypted data. An item of text is taken, and then authentication is supplied by encrypting it with the private key, which serves as a mechanism for authenticating the text. However, because encryption is time-consuming, another mathematical function is used instead. This function takes a text input and generates a seemingly random number of a specified size that can only have originated from the text used as input. This is what referred to as a "hash function". The whole cleartext is input into the hash function, creating a message digest that is 128 bytes long. TThe message digest undergoes encryption using the public key, with a digital signature serving as an alternative term for this process. After receiving the message, the receiver will recalculate the message digest by applying the hash function to the data and running it through its entirety. They decode using the public key, and if the digests match, then they know that the alleged sender truly sent the message and that the message was not altered, with the integrity of the message has been safeguarded. In other words, they know that the message was not altered. Below are a few variations of PKC:

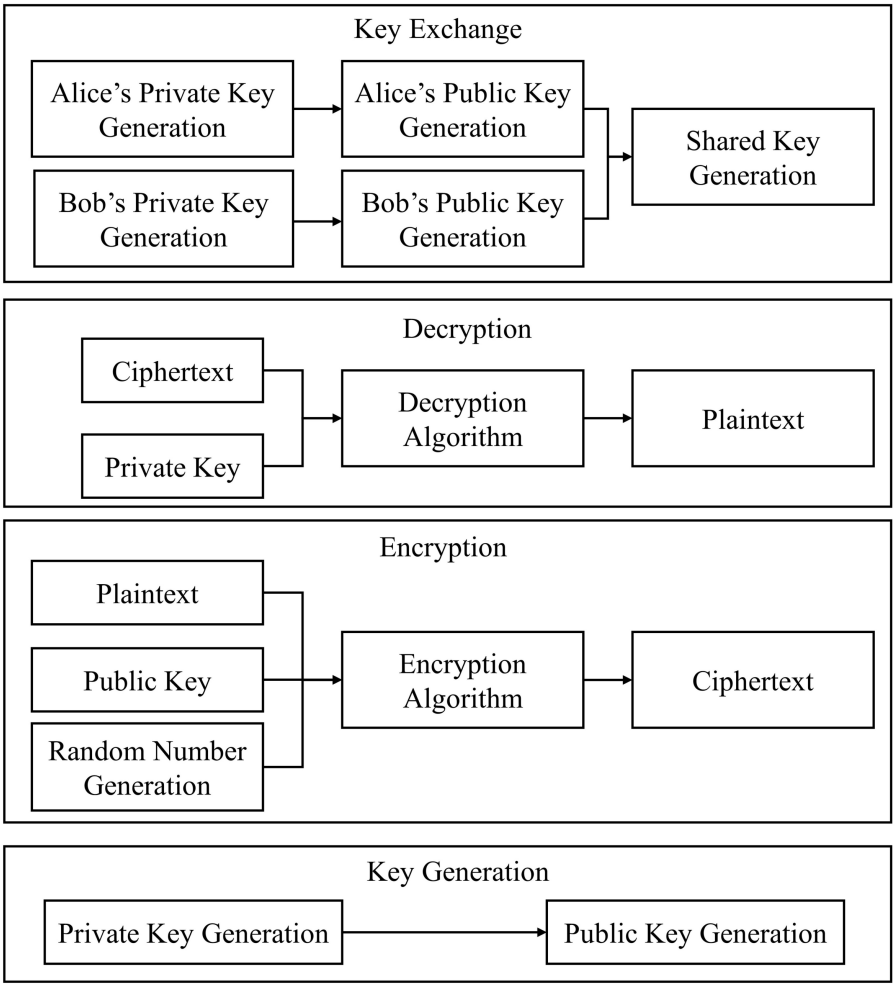


Figure 2.2: PKC block diagram

RSA is one of the PKC methods that finds the most extensive usage. It was developed in the 1970s. It requires the use of modular exponentiation as well as the challenging task of factoring huge composite numbers into their respective prime components. Encryption, digital signatures, and the exchange of keys are some of the most popular applications for RSA.

Using the Diffie-Hellman Algorithm to Conduct a Key Exchange Diffie-Hellman is a protocol for exchanging keys that allows two parties to generate a shared secret key via the use of a channel that is not secure. This key may then be used to communicate with each other. The complexity of the discrete logarithm issue is the foundation upon which it is built. In symmetric encryption systems, the Diffie-Hellman algorithm may be used for safe key exchange, and it can also be utilized as a building block for other cryptographic protocols.

ECC is a special PKC that may be used to solve problems involving elliptic curves over finite fields. In comparison to previous asymmetric encryption methods, it provides superior safety while using key lengths that are much lower. In various applications, particularly those with limited resource access, ECC encrypts data, creates digital signatures, and exchanges keys.

ElGamal Encryption is an asymmetric encryption technique that is based on the difficulty of the discrete logarithm problem. This problem requires the user to solve a complex mathematical equation. It offers a method for encrypting communications by making use of the recipient's public key and necessitates the use of the recipient's private key to decode the ciphertext. A wide variety of applications and protocols make use of the ElGamal encryption algorithm.

PKC also includes digital signature algorithms, which make it possible to generate and verify digital signatures. These algorithms enable digital signatures to be generated and verified. Signatures based on the RSA Public-key Cryptography Standard (RSA-PSS), Elliptic Curve Digital Signature Algorithm (ECDSA), and Digital Signature Algorithm (DSA) are all examples of digital signature algorithms. Data integrity, authenticity, and non-repudiation were achieved using digital signatures.

Lattice-based cryptography is a family of PKC schemes based on the difficulty of issues connected to lattice theory. This kind of cryptography is used in the field of cryptography. Using lattices in encryption provides a post-quantum level of protection, which means it can withstand attacks from quantum computers. It incorporates strategies such as Learning with Errors (LWE) algorithm, Ring Learning with Errors (RLWE) algorithm, and the NTRU algorithm.

Avantages of PKC:

- Security: Traditional SKC offers greater security than PKC since it requires two parties to discuss a secret key. Because of this, it is now more difficult for an adversary to read the encrypted communication if they manage to intercept it.
- Key circulation: With PKC, there is no need to exchange keys between two parties, as each has its own public and private keys. This makes key distribution much

easier, especially in large networks.

- Digital signatures: PKC makes it possible to generate digital signatures, which may subsequently be used to authorize the veracity of a message or article and ensure that it has not been tampered with. This is particularly useful for online transactions and other digital communication.
- Non-repudiation: Once a message has been signed with the sender's private key, they cannot refute that the message was sent by them, which is what is meant by the term non-repudiation in the context of PKC.

Disavantages of PKC:

- Complexity: PKC is more complex than traditional SKC, making it more difficult to implement and use.
- Performance: PKC is slower and more computationally costly than symmetric key encryption, so it was less suited for certain applications, such as real-time communication.
- Key management: Keeping track of public and private keys was difficult, especially in big networks with many different users. It cannot be easy to ensure that keys are kept secure and up to date.
- Vulnerabilities: While PKC is generally considered to be secure, it is nevertheless susceptible to a variety of attacks, including brute-force attacks and man-in-the-middle spells, among others. These vulnerabilities can be mitigated with proper key management and other security measures, but they are still risky.

2.2.1.1 RSA

The security provided by RSA is predicated on the fact that it is difficult to factor huge integers down to their prime factors. To break the RSA encryption, it needs to factor the *modulus* N, which is a computationally difficult task for prime numbers that are sufficiently big. The confidentiality of the RSA private key is essential to the algorithm's security. If an adversary manages to get their hands on the private key, they

can decode communications and even pose as the key's rightful owner. The RSA encryption technique is a deterministic one, which means that passing the same plaintext block and the same key to it will always result in the same ciphertext block being produced. This determinism can be a vulnerability in certain scenarios, such as encrypting the same message multiple times. To improve security, padding schemes are commonly used in RSA encryption. The padding adds randomization and structure to the plaintext before encryption, preventing certain attacks and improving security. Due to its computational complexity, RSA encryption is typically used for encrypting small amounts of data, such as symmetric encryption keys or digital signatures. For encrypting large files or messages, hybrid encryption schemes combining RSA with symmetric encryption algorithms are commonly employed. Figure 2.2 shows the RSA encryption block diagram. Initially, a key generation operation is performed. Choose the huge prime numbers p and q as starting points. These primes need to be kept a secret from everyone. To get the modulus, N, multiply p and q as shown in the following equation: $N = p * q$. The *modulus* N is included in both the public and private keys as an integral component.

Calculate the value of Euler's totient function, which is denoted by the symbol (N) and refers to the number of positive integers that are smaller than N and are coprime (meaning that they share no common factors) with $N.\phi(N) = (p-1) * (q-1)$. Pick an encryption exponent, e, that is less than (N) and is relatively prime to (N). The public key is constructed using the encryption exponent. Calculate the decryption exponent, d, in a way that makes $(d * e)(N)$ equal to 1. Using the extended Euclidean method will allow us to accomplish this task. The private key is constructed using the decryption exponent. (N, e) represents the public key, while (N, d) represents the private key.

Convert the plaintext message into a numeric representation for the purpose of encryption. This process is known as encoding. It can involve mapping characters or bit sequences to integers. Break the message into smaller blocks if necessary. The block size typically depends on the key size and the desired security level. For each block, calculate the ciphertext c using the encryption formula: $(c = (m^e)N)$, where m is the plaintext block and e is the encryption exponent.

The resulting ciphertext blocks from the encrypted message. Take the ciphertext blocks obtained from the encryption step for the decryption process. For each block,

calculate the plaintext message using the decryption formula: $m = (c^d)N$, where c is the ciphertext block and d is the decryption exponent. If necessary, combine the decrypted blocks to obtain the original plaintext message.

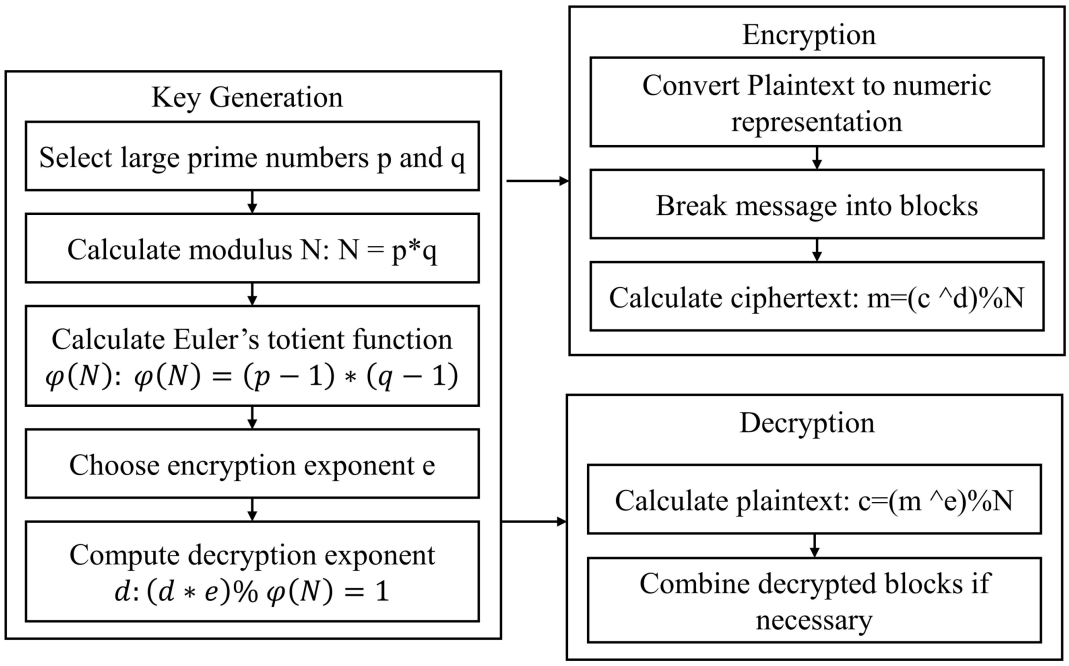


Figure 2.3: RSA encryption block diagram

2.2.1.2 Diffie Hellman Algorithm

The computational complexity of calculating discrete logarithms is a major component in the Diffie-Hellman key exchange, which provides the basis for the protocol's inherent security. Without having access to the private keys, it is thought to be computationally impossible to derive the private keys A and B or the shared secret key K from the corresponding public values A and B. This is the case even if the public values A and B are known. The Diffie-Hellman key exchange does not ensure either authentication or integrity of data. It does nothing more than establish a secret key known to both parties. As a result, further procedures, such as digital signatures or message authentication codes, are required to guarantee the genuineness of the communication and maintain its integrity.

Diffie-Hellman key exchange is often used in combination with other cryptographic protocols, such as Transport Layer Security (TLS), which ensures that communications

sent over a network are kept private to increase network safety and protection. It is possible for an opponent to perform a man-in-the-middle attack on Diffie-Hellman, in which they steal the public keys and replace them with their own keys after intercepting them. To mitigate this, mechanisms like certificate authorities or Public Key Infrastructure (PKI) can be used to verify the authenticity of the public keys. The security of Diffie-Hellman can be strengthened by using larger prime numbers and periodically updating the keys to mitigate potential attacks.

Figure 2.4 shows the Diffie Hellman encryption algorithm. Initially, the setup phase is performed using agreement on the parameters. The communicating parties agree on the values of a large prime number, p, and a primitive root modulo p, g. These values are publicly known and shared. Then, the key Exchange is performed, where Party-A generates its private key. Generate a random secret value, a, which is kept private. Party-B generates its private key. Generate a random secret value, b, which is kept private. Party-A calculates its public key, such as compute $A = g^a mod p$, where g is the primitive root and p is the prime number.

2.2.1.3 ECC

ECC is a kind of public-key cryptography that allows for secure communication as well as the exchange of keys and the creation of digital signatures. The mathematics of elliptic curves over finite fields is the foundation of this theory. When compared to other asymmetric encryption algorithms like RSA, ECC provides superior protection with key lengths that are far less than those required by RSA. The essential mathematical procedures in ECC revolve on elliptic curves and their associated points. An equation of $y^2 = x^3 + ax + b$ describes an elliptic curve. The variables a and b are constants. The fact that the curve is defined over a finite field indicates that the coordinates of the points on the curve have values that are also finite. The difficulty of finding a solution to the discrete logarithm issue posed by an elliptic curve contributes to security. To solve this issue, perform scalar multiplication to get the private key, which is a scalar value, from the public key, which is a point on the curve. The computational complexity of solving this problem is significantly higher than the analogous problem in traditional approaches.

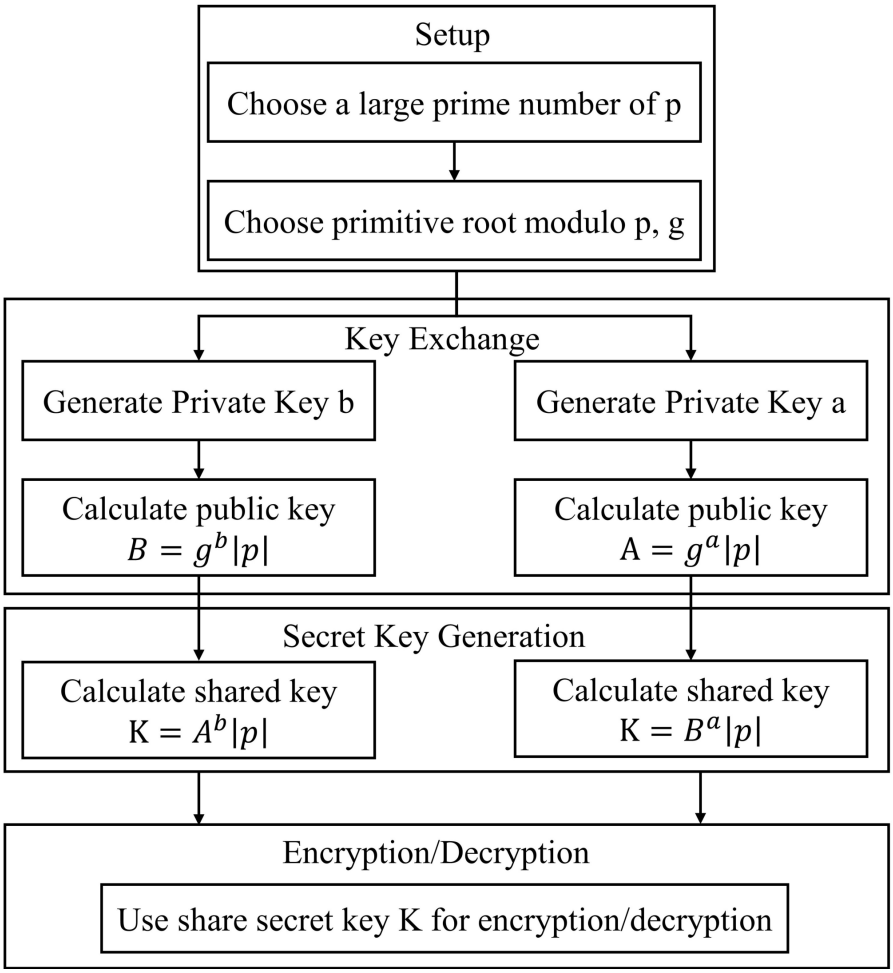


Figure 2.4: Diffie Hellman encryption

ECC offers a high level of security even with shorter key lengths, making it suitable for resource-constrained environments. ECC requires fewer computational resources and less bandwidth than other PKC algorithms, making it ideal for applications with limited resources. ECC enables secure key exchange between parties and can be used for secure communication by encrypting messages using shared secret keys. ECC allows the generation and verification of digital signatures, ensuring data integrity and authenticity. Figure 2.5 shows the ECC block diagram. Initially perform key generation. Choose an elliptic curve; select a specific elliptic-curve defined over a finite field. The curve parameters, such as the curve equation and base point, are publicly known and agreed upon. Select a private key, and generate a random secret number, which serves as the private key for ECC. Scalar multiplication is the name given to this operation. The x-coordinate and the y-coordinate on the elliptic curve make up the public key generated because of this process.

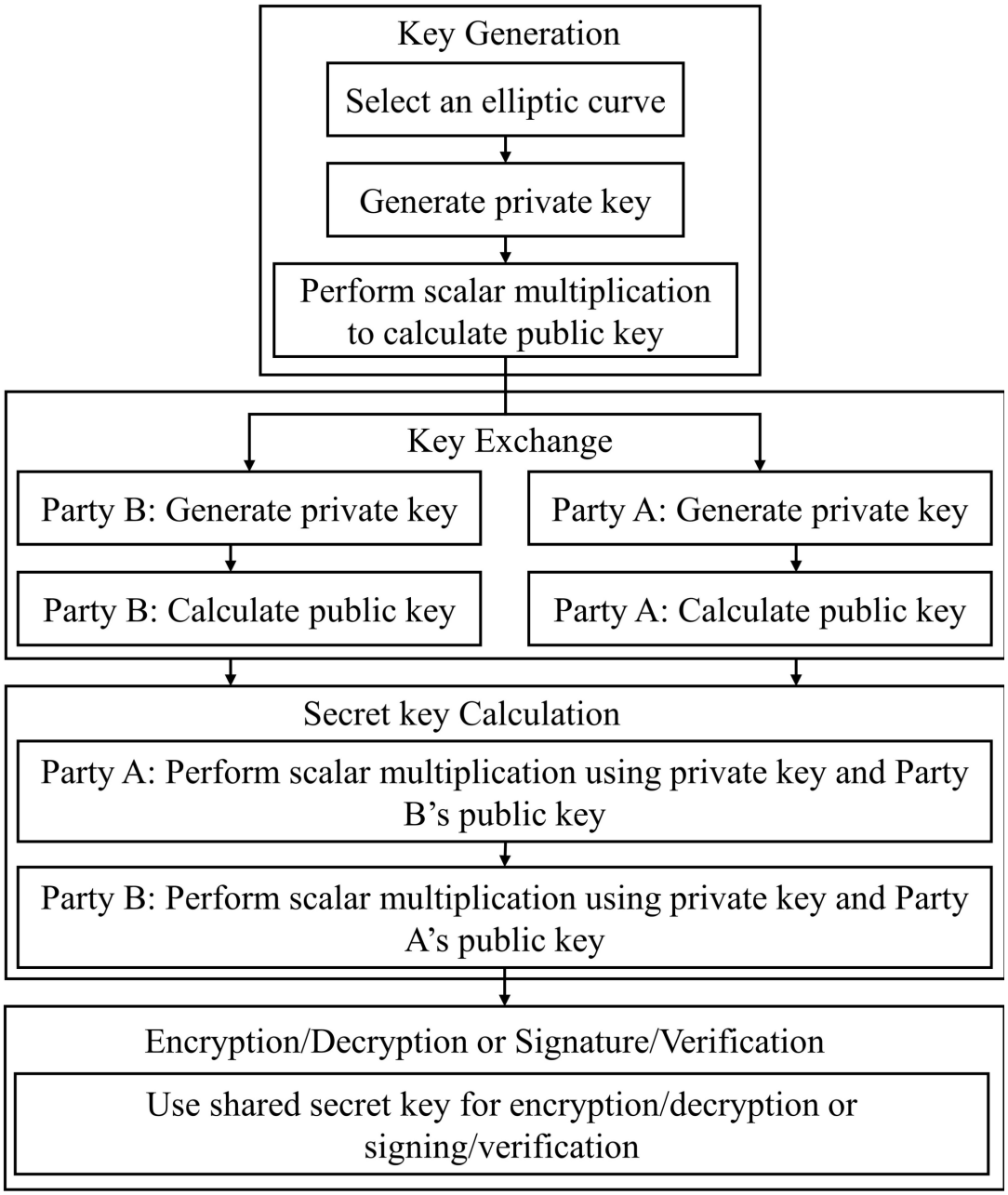


Figure 2.5: ECC block diagram

Key Exchange requires both party A and party B to use the identical set of elliptic curve parameters when calculating their respective private and public keys to complete the transaction. Party A provides Party B with access to its public key, while Party B provides Party A with access to its public key. Calculation of the Shared Secret Key: To compute the shared secret key, Party A must first conduct scalar multiplication using both its Private Key and Party B's Public Key. To determine the same shared secret key, Party B must first conduct scalar multiplication using both its own private key and

Party A's public key. A coordinate on the elliptic curve is what we get as a result for our shared secret key. Message Signing/Verification, Encryption/Decryption, or Digital Signature methods ECC may be used for either encryption/decryption or digital signature methods. Asymmetric encryption is commonly used to encrypt and decode data, along with a secret key known to both parties. This helps to keep the connection safe. When it comes to digital signatures, the shared secret key may be put to use to sign messages by using an appropriate signature algorithm, and the public key that corresponds to the shared secret key can be put to use to verify the signatures.

2.2.2 Secret Key Cryptography

SKC is characterized by a solitary secret key to encrypt and decrypt data. In this method, the sender and the receiver perform cryptographic operations using the same key. This ensures that the messages sent and received are secure [17]. This specific subject of cryptography has a broad use in computer networks, encrypted communication, and data storage to confirm confidentiality. The SKC encryption technique employs the same key for encryption and decryption processes [18]. The key, concealed in a safe place, is only known to the person who sends and receives the message. The transmission is encrypted with the secret key by the sender, and the message can only be deciphered by the recipient with the same key. The communication is encrypted when it is sent, and the message is decrypted when it is received. This restricts access to the message so that it can only be viewed by the designated recipient. Figure 2.6 depicts the SKC component architecture, essential for encrypting plaintext and decrypting ciphertext. This design is essential for attaining both goals. This method of securing information, encrypting, and decrypting data uses the same cryptographic key.

To transition between encryption and decryption procedures, the key must either be the same or be able to be quickly converted [19]. At least two of the people engaged need to know the secret that symbolizes the key for them to be able to retain the secrecy of their conversation. However, compared to PKC, often referred to as asymmetric-key encryption, SKC has a few drawbacks that must be considered. The fact that both people involved need to have contact with the secret key is the fundamental constraint in contrast to PKC [20], which requires two distinct keys for encryption and decryption, SKC customs

a single secret key known to both parties.

Despite these drawbacks, SKC systems provide several benefits when it comes to the protection of massive amounts of data. They are well known for their efficiency capacity and higher processing speeds. Except for the one-time pad encryption technique, the methods for encryption using symmetric keys often have lower key sizes. Because of this, less need for room to store data and quicker data transfer rates are necessitated. Combining symmetric-key and asymmetric-key encryption is common practice to circumvent the key-sharing constraint of SKC. This combination uses the benefits offered by both encryption methods to guarantee that any communication is safe and efficient.

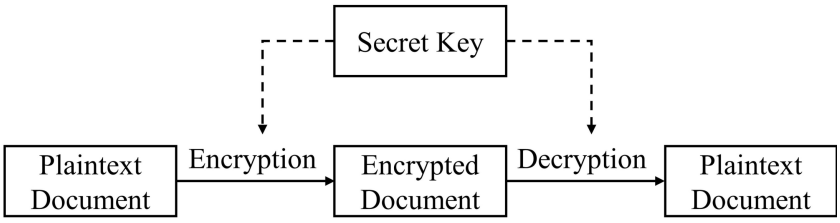


Figure 2.6: SKC block diagram

Advantages of SKC:

- Ease of Use: The simple key exchange is easy to construct and use since an individual single key is required and designed for encryption and decryption.
- Performance: SKC is superior to PKC in terms of speed and computational effectiveness; as an effect, it is better suited for real-time communication applications.
- Key management: It is simpler to handle a single key than to manage numerous keys simultaneously, which is the situation with PKC.
- Security: SKC can offer high security for data transfer and storage when utilized appropriately, which is one of its primary selling points.

Disadvantages of SKC:

- Key distribution: The biggest challenge with SKC is key distribution. If the key is stolen or otherwise made vulnerable, the security of any encrypted communications using that key was breached.

- Key compromise: If the key is compromised, it can be used to decrypt all encrypted messages, compromising the system's security.
- Key updates: If the key needs to be changed, updating all systems that use it can be challenging, particularly in large networks.
- Lack of non-repudiation: SKC does not provide non-repudiation, which means that a sender can deny that they sent a message once it has been decrypted.

2.2.2.1 Stream Cipher Cryptography

Stream Cipher Cryptography (SCC) encrypts data sequentially, either bit-by-bit or byte-byte. This means the plaintext is encrypted incrementally, with each bit or byte being processed individually. The encryption process involves the utilization of a key stream that is generated from a secret key. Stream cipher cryptography, also known as SCC [21], is used extensively in telecommunications, wireless communication, and other applications that need to encrypt huge volumes of data quickly and effectively. For the SCC to perform its functions.

The keystream must be produced by Pseudorandom Number Generator (PRNG) that inputs the secret key. To generate the ciphertext from the plaintext, a lengthy series of random bits or bytes is generated by the PRNG and then mixed with the plaintext [22]. An XOR operation performed in a bitwise fashion is used to merge the plaintext with the key stream. Separate encryption is performed on each bit and byte of the plaintext, with each bit and byte of the key stream being used appropriately. This procedure is repeated several times for each bit or byte that makes up the plaintext. The decryption of the ciphertext is performed independently for each bit and byte of the ciphertext, using the appropriate bit and byte of the key stream.

The structural component analysis of SCC is seen in Figure 2.7. Self-synchronous stream ciphers generate each character in the keystream by considering a predetermined number (n) of characters from the previous ciphertext [23]. This type of encryption can be traced back to the period of Vigenère, dating back to the 16th century. Additive self-synchronous stream ciphers encompass variations such as autokey ciphers and cipher feedback schemes. Because it uses a feedback technique, this kind of stream cipher is

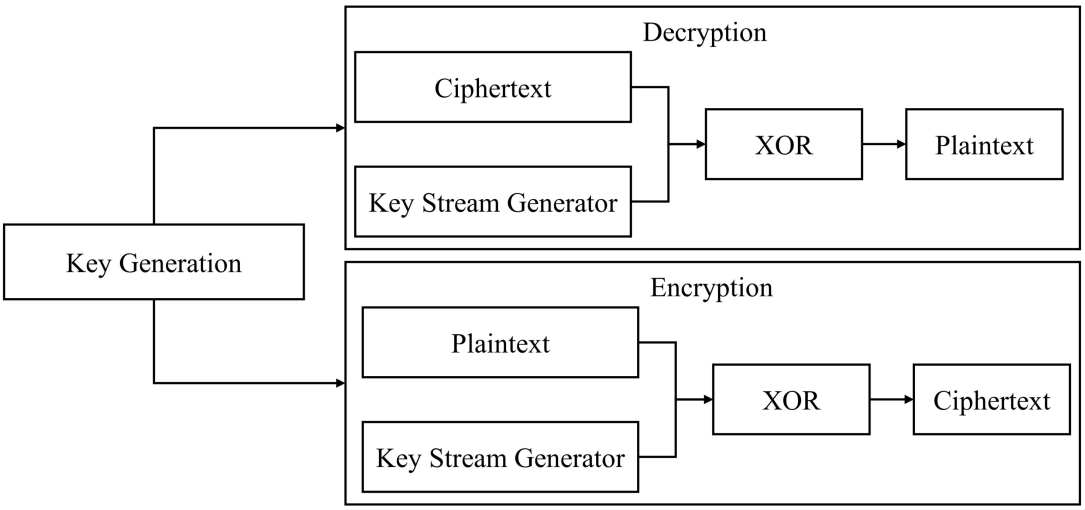


Figure 2.7: SCC block diagram

one of the most challenging to develop and analyze.There are several subtypes of stream ciphers commonly used in cryptography. Here are few examples:

Synchronous Stream Ciphers: Synchronous stream ciphers generate a keystream independently of the plaintext and ciphertext. They combine a secret key with an **IV!** (**IV!**) to generate a keystream. The keystream is then XORed with the plaintext to produce the ciphertext, and XORed again with the ciphertext to retrieve the plaintext during decryption. Examples of synchronous stream ciphers include RC4 and A5/1.

Self-Synchronizing Stream Ciphers: Self-synchronizing stream ciphers generate the keystream based on previous ciphertext bits. They use a feedback mechanism incorporating previously generated ciphertext bits into the keystream generation process. Self-synchronizing stream ciphers are designed to automatically synchronize the keystream generator with the incoming ciphertext stream, even in the presence of errors or missing bits. One example of a self-synchronizing stream cipher is the Self-Synchronizing Stream Cipher.

Grain-Based Stream Ciphers: Grain-based stream ciphers are based on the Grain family of stream ciphers. They utilize shift registers and non-linear feedback functions to generate the keystream. Grain-128 and Grain-256 are examples of grain-based stream ciphers that offer high security with relatively low resource requirements.

Multiplicative Stream Ciphers: Multiplicative stream ciphers are based on the multiplication of elements in a finite field. They employ mathematical operations such as

modular multiplication and exponentiation to generate the keystream. Multiplicative stream ciphers, like the Trivium cipher, are known for their simplicity and efficiency.

Filter Generators: Filter generators are stream ciphers that combine multiple Linear Feedback Shift Registers (LFSR) and nonlinear filter functions to generate the keystream. The keystream is produced by passing the output of the LFSRs through the filter function. Filter generators, such as Alternating Step Generator (ASG), offer increased security compared to basic LFSR-based stream ciphers.

Advantages of SCC:

- Speed: SSCs offer a fast and efficient encryption approach, allowing for data encryption on a bit-by-bit or byte-byte basis. This attribute makes them particularly well-suited for applications that demand swift and efficient encryption of substantial volumes of data, such as telecommunications and wireless communication.
- Key management SCC requires only a small key size, making key management easier than with other encryption methods.
- Low latency: SCC allows minimal latency, essential for real-time audio and video communication applications.
- Resistance to errors: SCC is more resistant to errors than block ciphers, as an error in one bit or byte of the ciphertext affects only that bit or byte rather than the entire block.

Disadvantages of SCC:

- Keystream reuse: If the same key stream is used for several messages, SCC is susceptible to attacks because an adversary may use statistical analysis to figure out what the key is and then recover it.
- Vulnerability to plaintext attacks: SCC is vulnerable to attacks if an attacker knows part of the plaintext, as they can use this information to improve the key.
- Limited security: SCC provides limited security, as the key stream is generated from a fixed-length key, which means the key can be brute-forced.

RC4 Encryption: Ron Rivest developed the idea for the symmetric stream cipher technique known as RC4 in 1987. As a result of its ease of use and effectiveness in software installation, it quickly achieved broad use. RC4 is well-known for its speed, both in terms of encrypting and decrypting data, which makes it an ideal choice for applications that have restricted access to computing resources. RC4 is a symmetric encryption algorithm that produces ciphertext by generating a stream of pseudo-random bytes, which are then XORed with the plaintext to form the ciphertext. The key length for RC4 may range from 1 to 256 bytes. A symmetric encryption algorithm is what RC4 is since it uses the same key for both the encryption and the decryption processes. It is essential to remember that RC4 is a stream cipher, which indicates that it encrypts and decrypts data on a byte-by-byte basis. This fact is critical to keep in mind.

Performing an XOR operation between the plaintext/ciphertext and the pseudo-random stream is one of the operational processes. Other parts of the process include initialization, key scheduling, the generation of a pseudo-random stream using the PRGA, and the generation of the ciphertext.

Figure 2.8 shows the RC4 encryption block diagram. Perform key scheduling. Key Setup, RC4 requires a secret key as input. Convert the key into an array of bytes, represented as $K[i]$, where i ranges from 0 to (key length - 1). Initialization of State Array: Create an internal state array, usually called S," with values from 0 to 255 in ascending order. Key Mixing, perform initial mixing of the state array S by iterating through all elements and swapping them with elements determined by the key. This is achieved by using a loop that processes each element of the state array and performs a swap operation based on the key.

Pseudo-Random Generation Algorithm (PRGA), Generate a pseudo-random stream of bytes, which is used to encrypt the plaintext. This is accomplished using a loop that iterates through the state array and generates a byte of the pseudo-random stream at each iteration. The state array is continuously modified during the generation process. Swap elements of the state array S to further enhance the randomness and unpredictability of the pseudo-random stream. The swapping is performed by iterating through the state array and swapping elements according to a specific algorithm. In the encryption process carry out an XOR operation on each plaintext byte in conjunction with the byte that

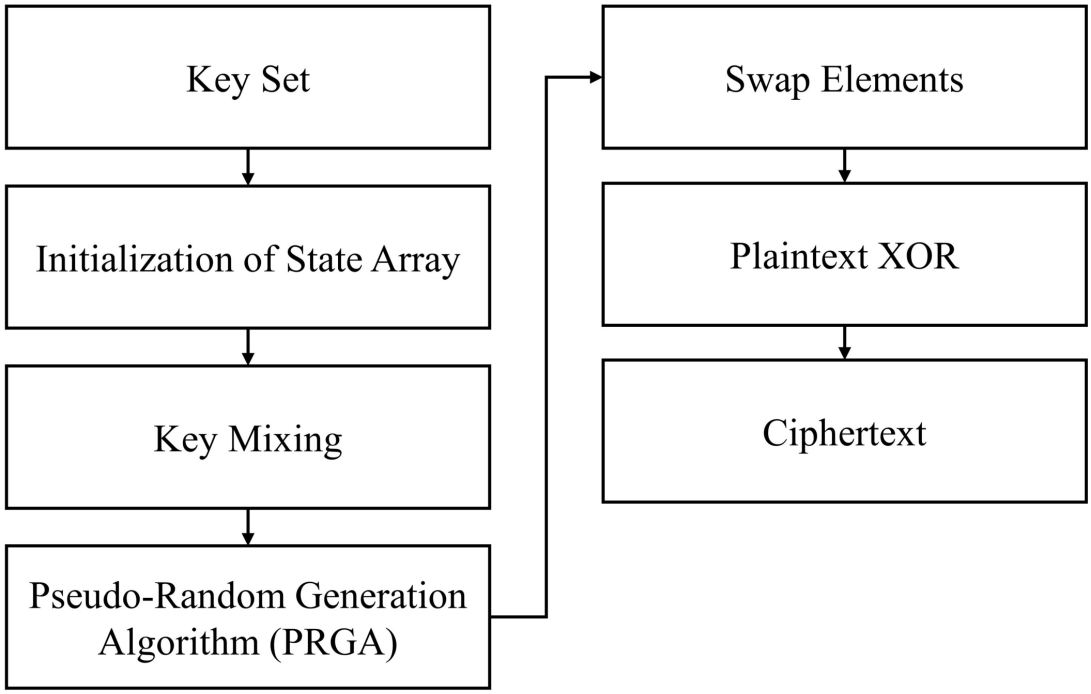


Figure 2.8: RC4 encryption block diagram

corresponds to it in the generated pseudo-random stream. The ciphertext is created using this bitwise operation by combining the bits of the plaintext with the bits of the pseudo-random stream. The technique of encrypting data using RC4 is identical to decrypting data using RC4. To decipher the original plaintext, use the same key and construct the pseudo-random stream, then execute an XOR operation between each ciphertext and the matching byte from the generated stream.

Salsa20 Encryption: Daniel J. Bernstein is the brains behind the Salsa20 algorithm, a symmetric stream cipher. Because of its high level of performance, ease of use, and security, it is used by many people. The key lengths used with Salsa20 are 128, 192, or 256 bits, and the algorithm runs on 64-byte blocks. The cipher known as Salsa20 is renowned for having superior diffusion qualities and a high level of resilience to known cryptanalytic attacks. It finds widespread usage in various applications, including secure communications and network protocols, as well as the encryption of disks.

Figure 2.9 shows the Salsa20 encryption block diagram. Generate a secret key, which is a sequence of bytes. The key should be a random value and should have sufficient entropy. Ensure that the key length is appropriate for the desired level of security. Salsa20 supports key lengths of 128, 192, or 256 bits. Store the key securely and keep it confiden-

tial. Generate a nonce, a unique value used only once for a specific encryption operation. The nonce can be randomly generated or derived from a counter that is incremented for each encryption operation.

The nonce should have sufficient length to prevent repetition. It is typically 64 or 96 bits long. Ensure the nonce differs for each encryption operation, even if the same key is used. Initialize the Salsa20 algorithm's internal state. The internal state comprises a 16-byte matrix called the "Salsa20 state". Set the initial state with specific constants and values derived from the key and nonce.

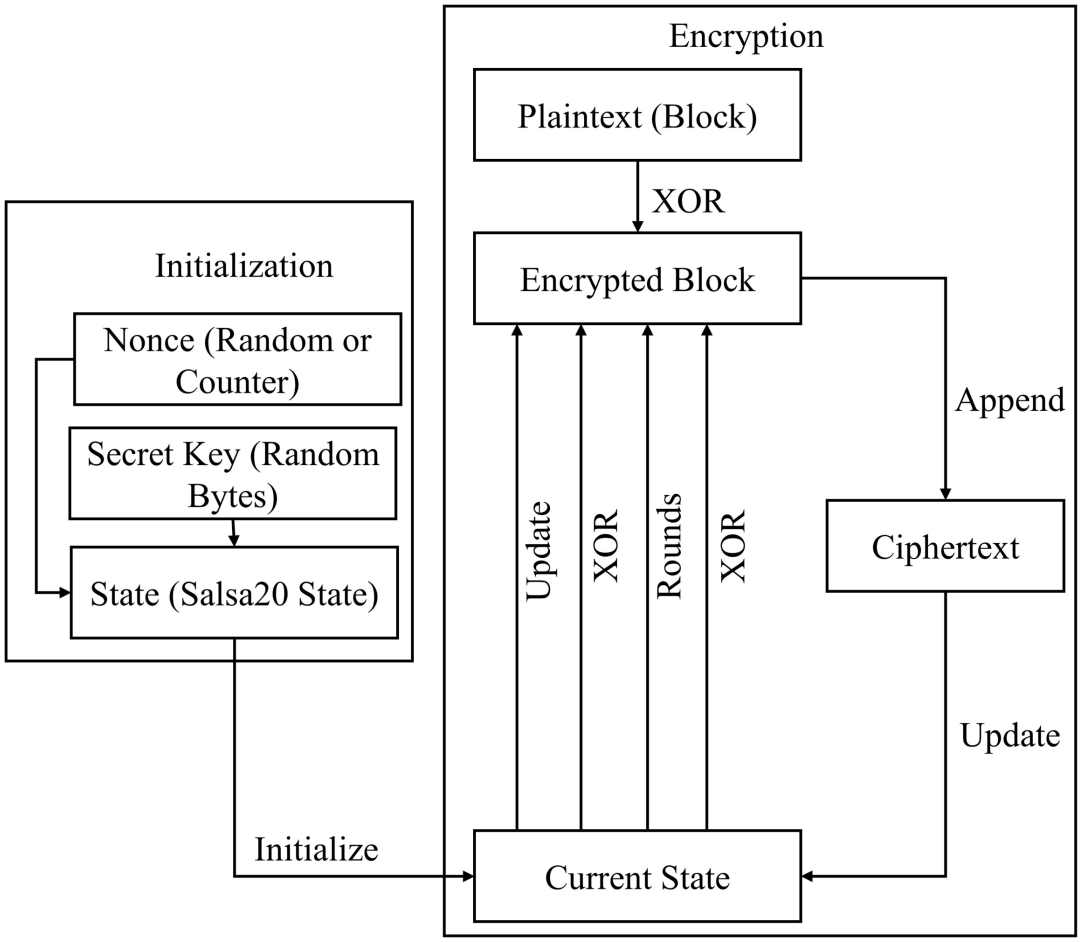


Figure 2.9: Salsa20 encryption block diagram

Divide the plaintext data into fixed-size blocks (usually 64 or 128 bits). To process each block using the Salsa20 algorithm, XOR the block with the current state. Perform a series of rounds (typically 20) of column and row operations on the block. XOR the block with the updated state. Repeat this process for each block of the plaintext data. If necessary, append padding to the last block to ensure it is of the required size. The

padding scheme used depends on the specific application or protocol. Apply any additional cryptographic operations or integrity checks, if required. The resulting encrypted blocks are the ciphertext. Transmit or store the ciphertext securely.

Grain-128 Encryption: Grain-128 is a stream cipher designed for efficient and secure encryption. It operates on a keystream generator based on a combination of LFSR and non-LFSRs.

Figure 2.10 shows the Grain-128 encryption block diagram. To perform key setup, obtain a secret key of 128 bits (or 80 bits for reduced version Grain-128a) Load the key into the key registers K (128 bits) and $LFSR$ (64 bits). Initialize the feedback tap positions for the $LFSR$. Load an initialization vector IV of 96 bits. Load the IV into the $LFSR$ state register (S). Perform 256 clocking cycles, discarding the output to initialize the cipher state.

To generate the key, perform clocking cycles to generate the keystream. During each cycle, the $LFSR$ is clocked once, and the output bit is obtained. The feedback taps are applied to the $LFSR$ based on the contents of the key registers. The keystream bit is obtained by XORing the output bit with the output of a non-linear feedback function. To encrypt a plaintext (P), XOR it with the generated keystream bit to obtain the ciphertext (C). To decrypt a ciphertext, XOR it with the generated keystream bit to obtain the plaintext. For each bit of the plaintext/ciphertext, repeat the keystream generation process described in step 3.

To update the Key, after generating each keystream bit, update the key registers for the next bit generation. Shift the key registers K and $LFSR$ by one bit to the left. If the leftmost bit of K was shifted out, shift it into the rightmost bit of the $LFSR$. Calculate the new value for the rightmost bit of K based on the old bit values of K and $LFSR$. Repeat this process until all keystream bits are generated. Once all plaintext/ciphertext bits have been processed, the cipher operation is complete.

2.2.2.2 Block Cipher Cryptography

Block Cipher Cryptography (BCC) refers to a specific cryptographic method that encrypts information using blocks of a predetermined size. In this approach, the plaintext

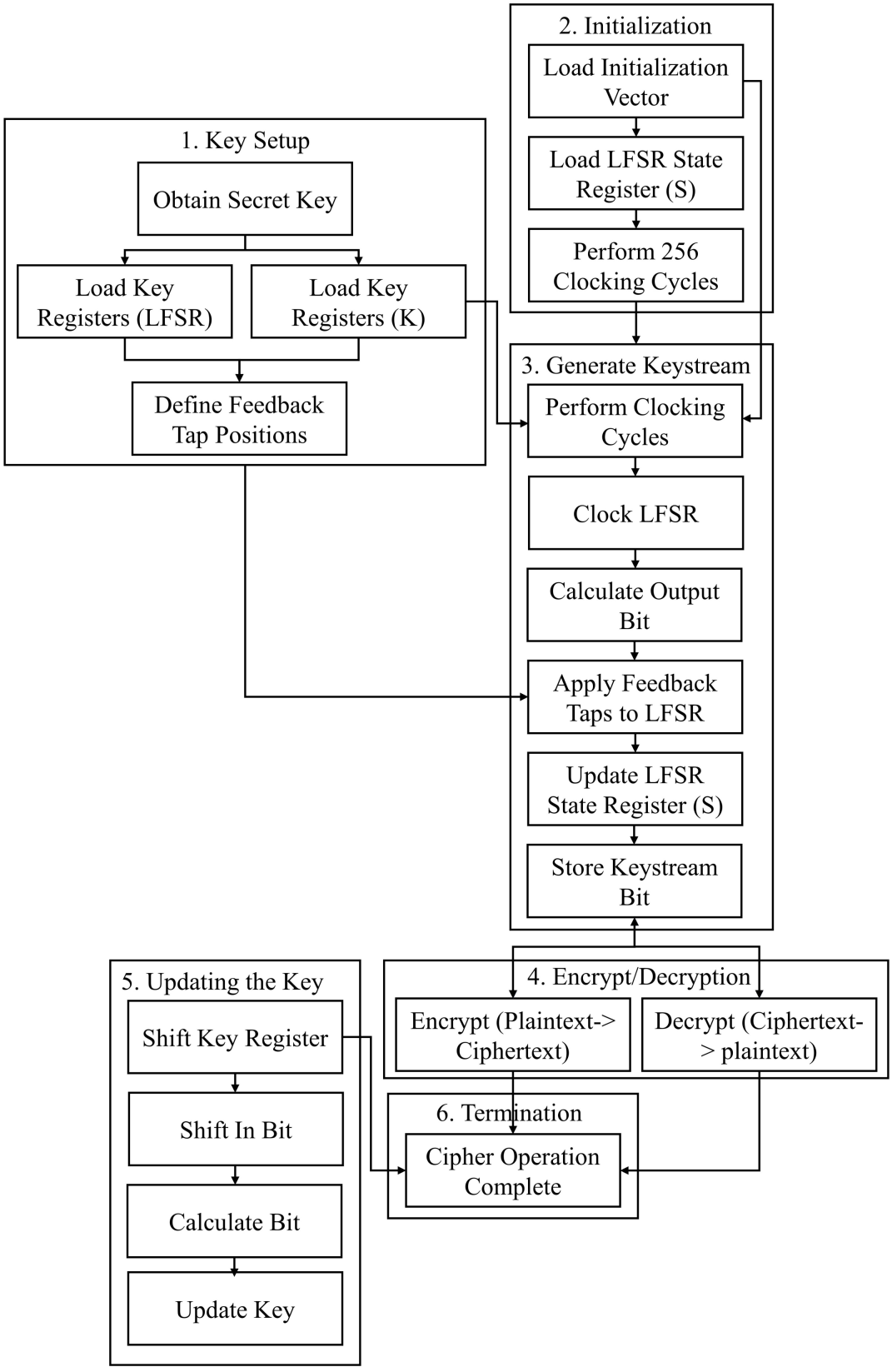


Figure 2.10: Grain-128 encryption block diagram

is segmented into blocks of a certain size, and the encryption of each block is carried out independently using a private key. File encryption, database encryption, and network security are just some of the many areas where BCC [24] finds widespread use. The implementation of the BCC is shown as a block diagram in Figure 2.11. The first step in the BCC process is called key generation, and it consists of a key generator producing a string of random bits or bytes to be used to create the private key. Following this step, the plaintext is segmented into blocks of a certain size, and the private key encrypts each block.

To carry out several iterations of both permutation and substitution, substitution boxes and permutation boxes are used throughout the encryption process [25]. These two types of boxes are referred to respectively as S-boxes and P-boxes. This ensures that the data is thoroughly scrambled and encrypted. During decryption, the steps are performed in the reverse order of encryption. Each block of the ciphertext is decrypted independently using the same secret key. The decryption process involves applying the same rounds of substitution and permutation but in the reverse order. This allows the original plaintext to be reconstructed from the ciphertext [26]. Using S-boxes and P-boxes in the encryption and decryption processes adds a high grade of confusion and diffusion, making it problematic for an adversary to analyze the encrypted data and retrieve the original information without knowing the secret key. So, the BCC block cipher employs key generation, block encryption, and block decryption to provide secure encryption and decryption of data. Using S-boxes and P-boxes combined with multiple rounds of substitution and permutation ensures strong encryption and data confidentiality.

A block cipher is a type of encryption technology that operates on data blocks of a predetermined size, often 64 or 128 bits in length. This kind of cipher works on data blocks rather than individual data bits. An operation of a block cipher consists of performing a sequence of mathematical operations on the block that is being input, which is then turned into a new block that is the output of the operation [27].

The Substitution-Permutation Network (SPN), a particular kind of block cipher architecture, is an example of a method often used in block ciphers. An SPN network begins by sending the input block via a substitution layer, often called an S-box. This layer performs a non-linear modification on the bits sent in and is the initial step in the

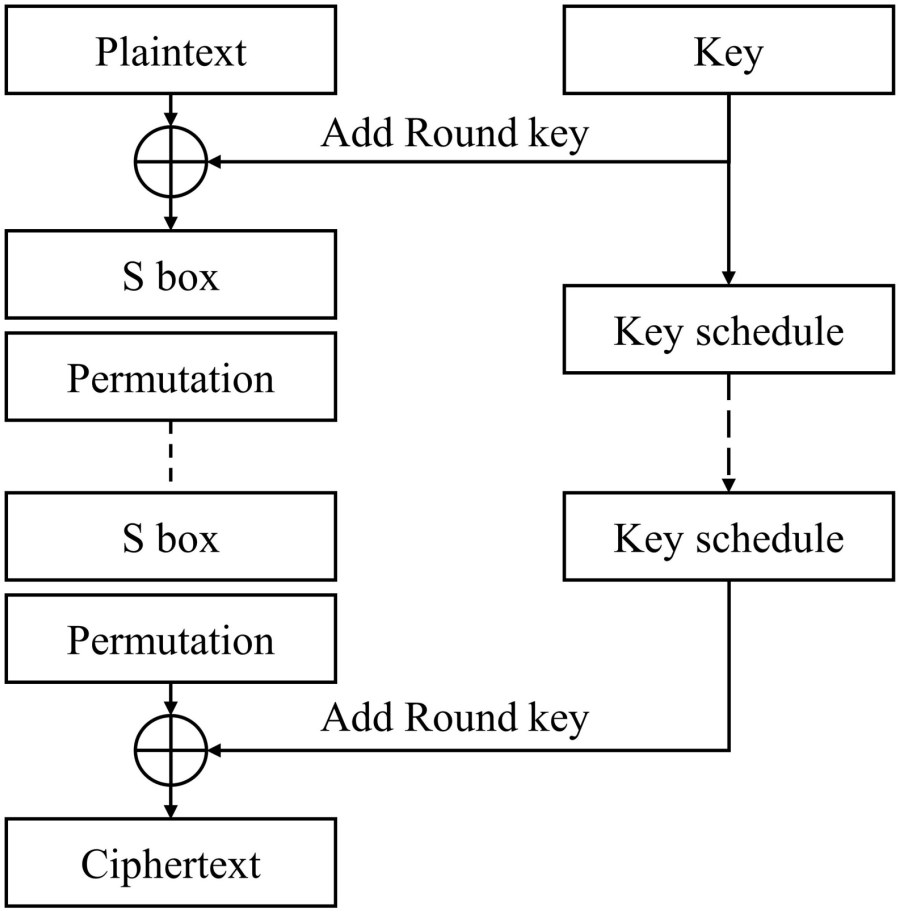


Figure 2.11: BCC block diagram

process. After the output of the S-box has been encrypted, it is sent to a permutation layer. This layer further scrambles the data by shuffling the bits in random order. The S-box is an essential part of the SPN network, and the design of this component is very important to the cipher's overall safety. In most cases, the S-box is implemented as a lookup table that connects each potential input value with a value corresponding to it on the output side. Changing the encryption key may drastically alter how the S-box behaves. This is because the precise mapping utilized by the S-box is often decided by the encryption key. Combining algebraic qualities with cryptographic features is a method that is often used in the process of designing S-boxes.

For instance, the S-box was developed to have strong differential and linear characteristics, which makes it resistant to differential and linear cryptanalysis attacks. Additionally, the S-box was built to have high differential and linear properties. Therefore, using S-boxes as block ciphers is a useful strategy for enhancing the level of protection

provided by encryption techniques. S-boxes may assist in guaranteeing that the output of the cipher is extremely unexpected by performing a non-linear modification to the data being entered into the cipher. This makes it more difficult for potential data thieves to decode the information. Here are some types of block cipher cryptography:

The **DES!** (**DES!**) is a symmetric block cipher that employs a 56-bit key and works on 64-bit data blocks at a time to encrypt and decrypt information. To encrypt data, it goes through 16 different rounds using a Feistel network topology. DES was extensively employed in the past; but, since it had a tiny key size, it was susceptible to brute-force attacks; as a result, it was eventually replaced with more secure algorithms.

The AES is a symmetric block cipher that can work on data blocks that are 128 bits long and can handle key sizes of 128, 192, or 256 bits. It operates using an SPN structure and executes a different number of rounds (ten, twelve, or fourteen) depending on the size of the key being used. Because of its high level of security and its high level of efficiency, AES has become the de facto standard for symmetric encryption, and it is extensively employed in a variety of applications.

The Triple Data Encryption Standard, often known as 3DES, is a symmetric encryption system that employs two or three distinct keys to perform a cascaded application of the DES algorithm three times. It processes data in blocks of 64 bits and can handle key sizes of 128, 192, or 256 bits (with the effective key size being 112, 168, or 168 bits, respectively). The key size is 168 bits. Due to the greater key length required, 3DES provides more security than DES; nevertheless, its performance is much inferior.

Blowfish is a symmetric block cipher that can work on variable-length blocks ranging from 32 bits all the way up to 448 bits in length. It also supports key sizes ranging from 32 bits all the way up to 448 bits in length. It employs a Feistel network topology and executes a configurable number of rounds dependent upon the size of the key being used. Blowfish is renowned for its ease of use and adaptability, and it quickly rose to prominence because to the rapidity with which it could encrypt and decode data.

Twofish is an improved symmetric block cipher developed as a successor to Blowfish. Twofish was given the name "Twofish." It can work on data blocks that are 128 bits in size and supports key sizes that are 128, 192, or 256 bits. Twofish makes use of a Feistel

network that has a maximum of 16 rounds and includes key-dependent S-boxes. This combination results in high levels of security and excellent speed.

The Serpent cipher is a symmetric block cipher that works on data blocks of 128 bits and supports key sizes of 128, 192, or 256 bits. Its name comes from the fact that it resembles a serpent. A SPN-based framework is used, and 32 rounds of encryption are carried out. The cryptographic community has conducted a great deal of research and evaluation on Serpent, which has led to the widespread recognition of its ease of use and high level of safety.

Camellia is a symmetric block cipher developed jointly by NTT and Mitsubishi Electric. Camellia was given the name Camellia. It can work on data blocks that are 128 bits in size and supports key sizes that are 128, 192, or 256 bits. Camellia incorporates aspects of both AES and Serpent, making it a flexible encryption algorithm with excellent levels of performance and security.

CAST-128 is a symmetric block cipher that works on data blocks that are 64 bits in size and supports key sizes that range from 40 bits all the way up to 128 bits in length. To encrypt data, it goes through 12 cycles using a Feistel network topology. CAST-128 strikes a healthy compromise between performance and safety, and as a result, it has found widespread use in various applications.

IDEA, or the International Data Encryption Algorithm, is a symmetric block cipher that works on 64-bit data blocks and has a constant key size of 128 bits. Its full name is the International Data Encryption Algorithm. It uses a modified Feistel network structure and performs 8.5 rounds of encryption. IDEA employs a combination of modular arithmetic and bitwise operations to achieve its encryption and decryption operations. It was widely used and gained popularity for its strong security and efficiency. However, due to the advancement of newer encryption algorithms, IDEA is less commonly used today.

Advantages of BCC:

- Security: BCC provides a high level of security, as it uses a complex encryption algorithm resistant to brute-force attacks.
- Flexibility: BCC can be used with different block sizes, allowing it to be customized

for different applications.

- Key management: BCC requires only a small key size, making key management easier than with other encryption methods.
- Resistance to statistical attacks: BCC is resistant to statistical attacks, as it uses a complex encryption algorithm designed to produce ciphertext that is statistically indistinguishable from random data.

Disadvantages of BCC:

- Slower than Stream Cipher: BCC is slower than SKC, as it encrypts data in fixed-size blocks rather than on a bit-by-bit or byte-byte basis.
- Vulnerability to known plaintext attacks: BCC is susceptible to attacks using known plaintext because an adversary possessing a portion of the plaintext may use this information to retrieve the key.
- Vulnerability to side-channel attacks: attacks on the side channel, such as timing attacks and power analysis attacks, may successfully target BCC because of its insecure design.

AES: It is a popular symmetric encryption technique. In 2001, the NIST in the United States chose it to replace the deprecated DES. AES has now established itself as the de facto standard for protecting sensitive information. It is being used in a variety of applications, ranging from the protection of stored data to the safeguarding of communication channels. The ability of AES to deliver encryption that is both safe and effective is one of its greatest strengths. It processes data in blocks of a fixed size, usually 128 bits, and supports key sizes of 128, 192, and 256 bits. The AES encryption algorithm uses a structure known as SPN, which offers excellent protection against cryptanalysis attacks. The AES algorithm comprises numerous rounds of operations, some of which include substitution, permutation, and mixing. These operations are carried out on the data read using a set of round keys for each round. The number of rounds required varies depending on the key size: 10 rounds are required for 128-bit keys, 12 rounds are required for 192-bit keys, and 14 rounds are required for 256-bit keys. Each round modifies the data, producing

diffusion and confusion to make the encrypted output highly randomized and resistant to attacks. This is done to guarantee that the encrypted output is secure.

The plaintext is transformed into ciphertext during the AES encryption process, rendering the data inaccessible without the correct decryption key. The ciphertext may be converted back into its original plaintext form by the use of AES decryption, which is the reversal of the encryption process. The only thing that changes while encrypting or decrypting with AES is the sequence in which the round keys are used; otherwise, the method is the same. It can operate effectively on a diverse collection of hardware, ranging from low-powered embedded systems to powerful servers. The fact that AES has been subjected to a great deal of scrutiny and scrutiny from members of the cryptographic community instills trust in the system's security and dependability. The block diagram for the AES may be seen in Figure 2.12. During the key expansion process, the primary encryption key is stretched out to produce a collection of round keys. A key schedule technique is used to generate the key for each subsequent round from the key for the preceding round. first Round: The plaintext input block is used to start the first round of the AES algorithm. An XOR operation performed bitwise on the plaintext block and the first-round key merges the two.

Rounds: AES consists of multiple rounds, which vary based on the key size. Each round applies a set of transformations to the data, including substitution, permutation, and mixing operations. The number of rounds is determined by the key size: 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. The operations performed in each round include SubBytes. Nonlinear substitution of each byte using the AES S-box. ShiftRows: Shifting the rows of the block to provide diffusion. MixColumns: Mixing the columns of the block to provide diffusion. AddRoundKey, combining the round key with the current state using bitwise XOR. Final Round: The final round does not include the MixColumns operation. It performs the SubBytes, ShiftRows, and AddRoundKey operations on the current state. Output: After all the rounds are completed, the resulting state represents the encrypted ciphertext.

AES Decryption: The original encryption key is expanded in the same way as in the encryption process. Initial Round: The first round of AES decryption begins with the input ciphertext block. The ciphertext block is combined with the last round key

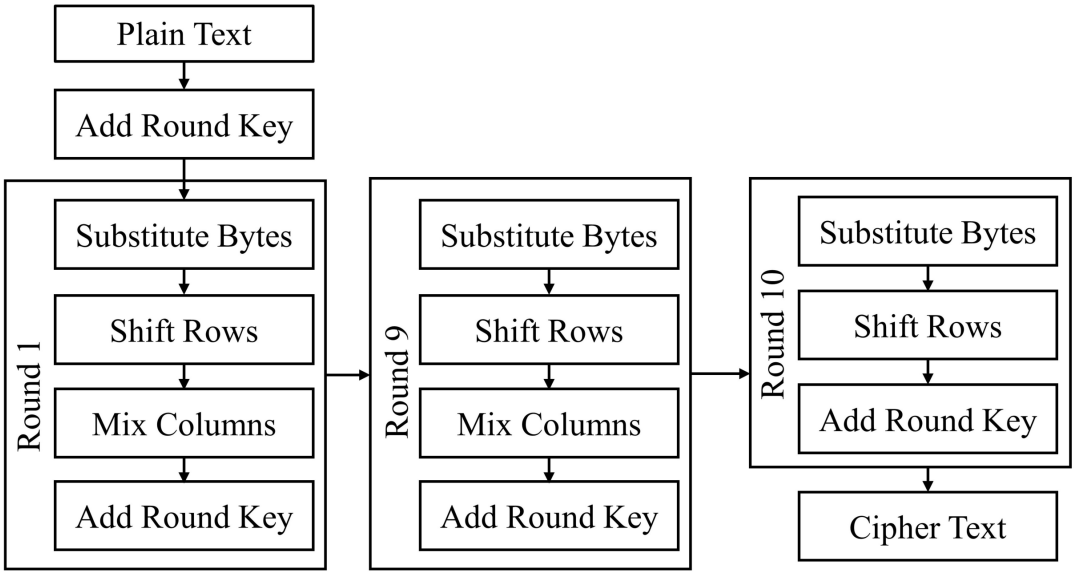


Figure 2.12: AES block diagram

using a bitwise XOR operation. Rounds: The decryption process reverses the order of the encryption rounds. Each decryption round applies inverse operations to those used in encryption: InvSubBytes: Inverse substitution of each byte using the inverse S-box. InvShiftRows: Inverse shifting of the rows. InvMixColumns: Inverse mixing of the columns. AddRoundKey: Combining the round key with the current state using bitwise XOR. Final Round: The final round in decryption does not include the InvMixColumns operation. It performs the InvSubBytes, InvShiftRows, and AddRoundKey operations on the current state. After all the rounds are completed, the resulting state represents the decrypted plaintext.

3DES: It is a symmetric encryption algorithm that applies the DES cipher three times to each data block. It was designed to provide enhanced security compared to the original DES algorithm. DES was a widely used symmetric encryption algorithm introduced in the 1970s. However, over time, advances in computing power made DES vulnerable to brute-force attacks. To address the security concerns associated with DES, 3DES was developed. It applies the DES algorithm three times in a row, using two or three different keys. The three stages of encryption are typically referred to as Encrypt-Decrypt-Encrypt (EDE) or Encrypt-Encrypt-Encrypt (EEE).

3DES supports different key lengths: 2-key and 3-key. In 2-key 3DES, two 56-bit keys are concatenated to form a 112-bit key. In 3-key 3DES, three different 56-bit keys

are concatenated to form a 168-bit key. The larger key size of 3-key 3DES offers higher security. 3DES, while considered more secure than DES, has become less popular in recent years due to the availability of more advanced encryption algorithms like AES. AES offers better performance and security compared to 3DES. However, 3DES is still used in legacy systems, as well as in certain industries and applications where compatibility or regulatory requirements mandate its use.

Figure 2.13 shows the 3DES block diagram. Key Generation, for 2-key 3DES, two 56-bit keys, Key 1 and Key 2, are generated. For 3-key 3DES, three 56-bit keys, Key 1, Key 2, and Key 3, are generated. Encryption Process, divide the plaintext into blocks, typically 64 bits each. Apply the following steps to each plaintext block. Encrypt the plaintext block using Key 1 with the DES algorithm. Decrypt the result of Round 1 using Key 2 with the DES algorithm. Encrypt the result of Round 2 using Key 3 with the DES algorithm. The final encrypted block becomes the ciphertext.

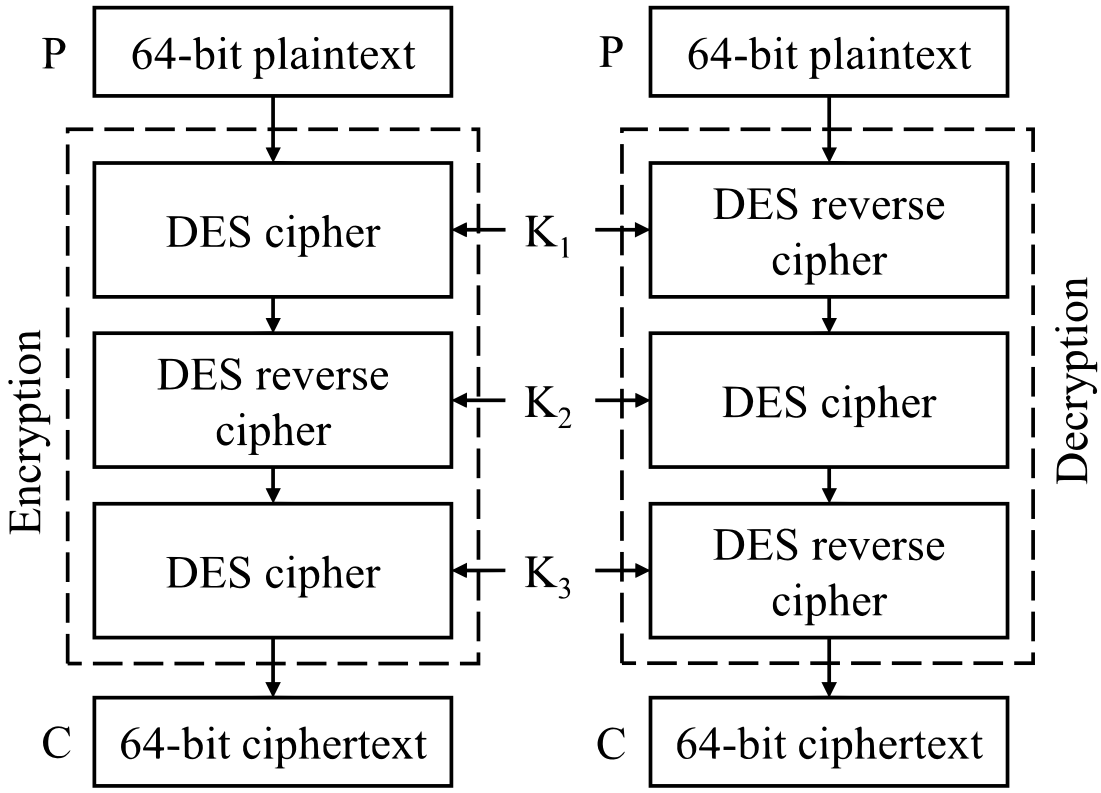


Figure 2.13: 3DES block diagram

The first step in the decryption process is to segment the ciphertext into blocks, which normally consist of 64 bits each. It is necessary to apply the following processes on each block of ciphertext. Decrypt the ciphertext block using Key 3 in conjunction with

the DES algorithm. Using the DES method, encrypt the output of Round 1 using Key 2. Using the DES method, decrypt the output of Round 2 using Key 1. The plaintext is obtained from the last encrypted block that is decoded. In the event that the length of the plaintext does not constitute a multiple of the block size, which is 64 bits, padding will be added to the last block before encryption takes place. PKCS 5 and 7 are examples of common padding schemes. In these schemes, the value of the padding bytes is equivalent to the number of padding bytes that have been added. The security provided by 3DES relies heavily on using sound key management methods. It is important that keys be produced and maintained safely and that only authorized parties can access them. It is advised that while trying to improve security, rotate keys and delete old keys in a safe manner.

Blowfish: Bruce Schneier developed the Blowfish symmetric key block cipher method in 1993. Blowfish is a block cipher. It is well-known for its ease of use and its adaptability, and it provides a satisfactory compromise between safety and performance. Blowfish allows various key sizes ranging from 32 bits all the way up to 448 bits. However, it works on blocks of a fixed size (usually 64 bits). Blowfish employs a key setup phase to expand a variable-length key into a series of subkeys. During this phase, the algorithm performs several iterations, known as rounds, using a combination of the key data and a fixed set of S-boxes (substitution boxes). The generated subkeys are used in subsequent encryption and decryption operations. Blowfish was designed to be secure and resistant against known cryptographic attacks. It gained popularity for its speed and simplicity and has been widely used in various applications and protocols. However, due to the advancement of newer encryption algorithms like AES, Blowfish is now considered less secure and is not recommended for new systems. Nevertheless, it remains used in some legacy systems where compatibility or specific requirements mandate its usage.

Figure 2.14 shows the Blowfish block diagram. The user provides a variable-length key (between 32 and 448 bits). The key establishes the Blowfish state, which contains the initial P-array (an array of 18 32-bit subkeys) and four S-boxes (each holding 256 32-bit entries). Both components are part of the Blowfish state. Initial values for the P-array and S-boxes are calculated from the hexadecimal digits of pi, which are used to populate the arrays. To produce the first subkeys, the key is XORed with the elements of the

P-array in an iterative way. During the encryption process, the plaintext message is first broken into blocks, each generally consisting of 64 bits. A left half consisting of 32 bits and a right half consisting of 32 bits are each separated from the beginning block. The block proceeds through several rounds, each consisting of the following phases (normally 16 rounds, but there may be up to 24 rounds total).

In this step, the round subkey is XORed with the left half of the current block. The output is then put through the Blowfish encryption mechanism, which consists of operations known as S-box substitution and P-box permutation. The result of the encryption function is XORed with the portion of the current block to the right of the current position. In the next iteration of the game, the left and right halves of the block will switch places. After the last round, the block left behind creates the ciphertext. Decryption Process, the ciphertext block undergoes the same rounds as encryption, but the subkeys are applied in reverse order. The decryption process follows the same steps as encryption, but the round subkeys are used in reverse order. After the final round, the resulting block becomes the plaintext.

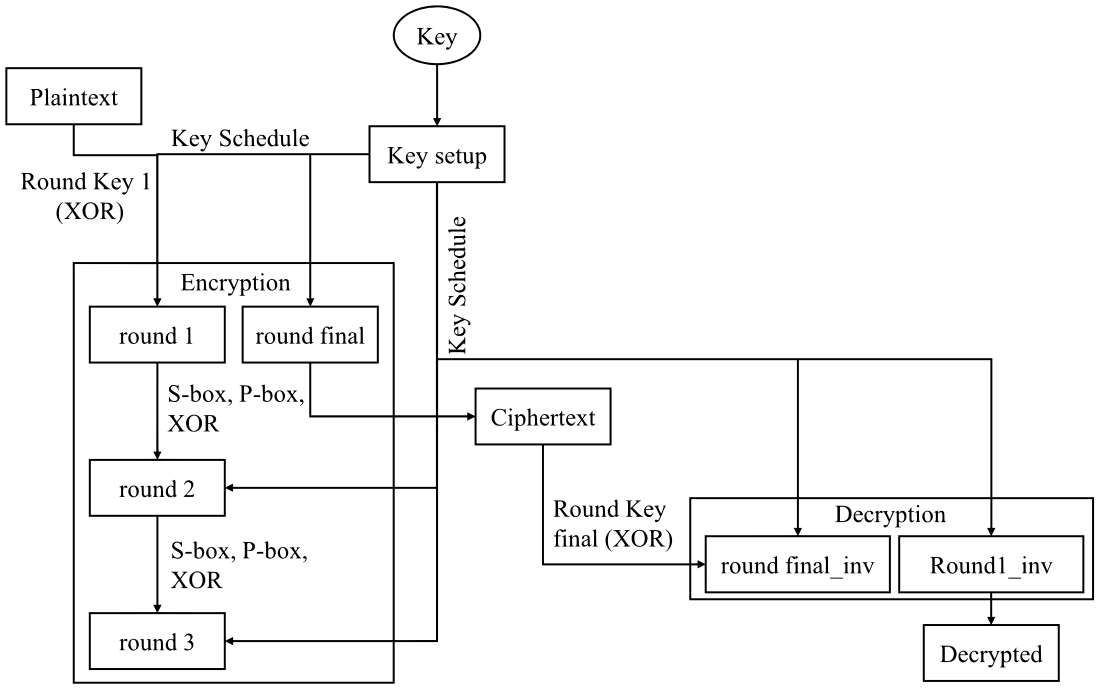


Figure 2.14: Blowfish block diagram

CAST-128: In 1996, Carlisle Adams and Stafford Tavares came up with the idea for the symmetric encryption technique known as CAST-128, which is also known as CAST5. It is a block cipher that uses fixed-size blocks (64 bits) to carry out its operations and

supports key sizes ranging from 40 to 128 bits in length. The CAST-128 protocol is well-known for its ease of use, security, and effectiveness.

The structure of the Feistel network serves as the foundation for CAST-128. The ciphertext is generated using a Feistel network by dividing the input block into two halves and then applying a sequence of rounds in an iterative manner. Each cycle includes the merging of the two parts of the data by using a combination of the operations of substitution and permutation. After a comprehensive analysis, it has been shown that CAST-128 is safe against all known forms of cryptographic attack. It has found widespread use in a broad variety of protocols and programs. On the other hand, as with other more antiquated encryption algorithms, it is typically suggested that more current algorithms like AES be used for brand-new systems and applications. The block diagram for the CAST-128 may be seen in Figure 2.15. The user supplies a key with a length that may range anywhere from 40 bits to 128 bits. The key goes through a procedure called "key setup," which involves mixing the key bits with constant values and executing key-dependent S-box replacement. This process is performed on the key. The round subkeys that are used in the encryption and decryption procedures are generated during the process of setting up the key.

The Process of Encryption: The message's plaintext is broken up into blocks, which generally consist of 64 bits apiece. The left half that has 32 bits and the right half that also contains 32 bits make up the first block. The block proceeds through a series of rounds, each consisting of the following stages. The number of rounds commonly ranges from 12 to 32, depending on the key size. A round subkey performs an XOR operation on the right half. The result is then sent to the CAST-128 round function, which performs operations such as S-box substitution and permutation. The result of the XOR operation performed on the output of the round function and the left half of the current block is shown. In the next iteration of the game, the left and right halves of the block will switch places. After the last round, the block left behind creates the ciphertext. The Process of Decryption, The same number of encryption rounds, are performed on the ciphertext block as on the plaintext block, but the subkeys are applied in the opposite order. The decryption process follows the same stages as the process of encryption, with the exception that the round subkeys are utilized in the opposite order. The resultant block is what is

known as the plaintext after all rounds have been completed.

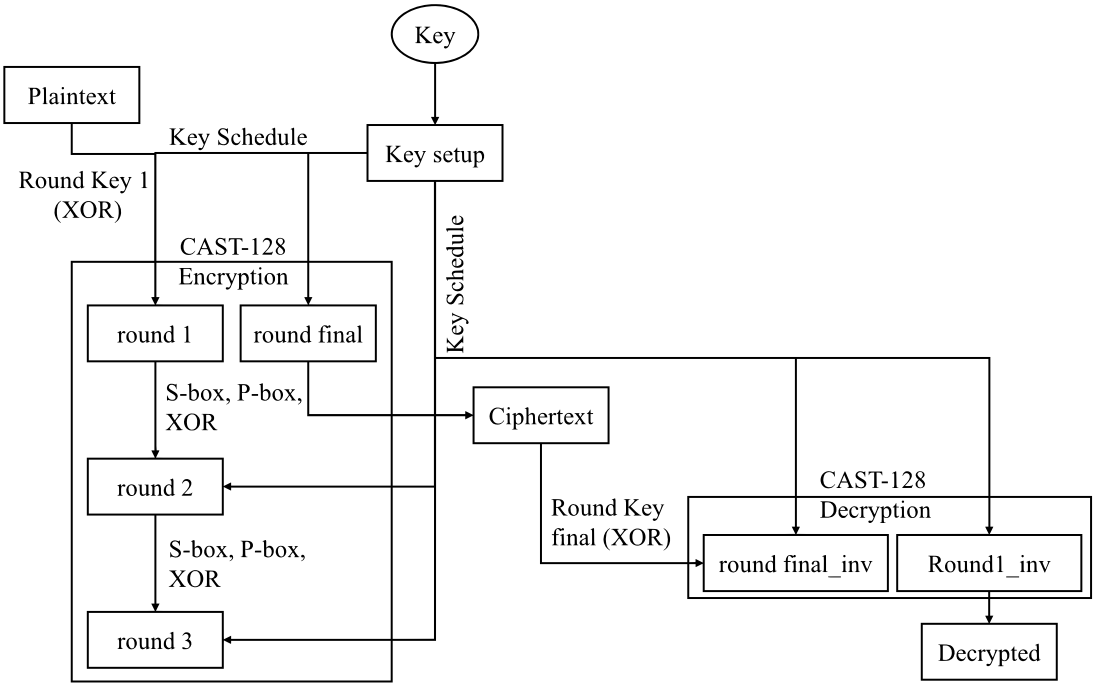


Figure 2.15: CAST-128 block diagram

2.2.3 Hash Function Cryptography

Hash Function-Based Cryptography (HFC) is a kind of cryptographic technology that secures the genuineness and integrity of data using hash functions [28]. This style of cryptography is frequently referred to as HFC. The result of running a hash function, sometimes called a message digest or a hash value, is a mathematical procedure set in length and generated from the input data by a hash function.

Because the hash value is specific to the input data [29], any variation in the data will result in a new hash value. This is true even if a single character is altered in the data. Figure 2.16 visually represents the block diagram for the hash function. A string (numbers, alphabets, or media files) of any length was sent into a hash function, changing it into a string of a predetermined length. The hash function now being used may determine the fixed bit length, ranging from 32 bits to 64 bits or even 128 bits or 256 bits. The output of a hash function is always of a defined length [30]. Moreover, this hash is a cryptographic byproduct produced by a hash algorithm. The below Diagram should help us comprehend it better.

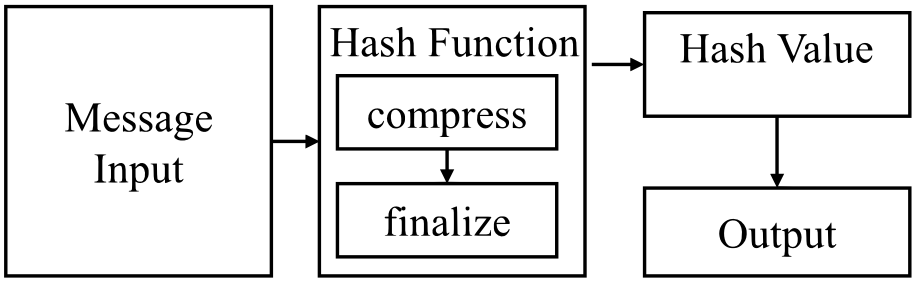


Figure 2.16: HFC block diagram

The first step in HFC is message digest generation. Hash functions are applied to the data that are being supplied to construct the message digest. The result is a value called a hash with a fixed length and reflects the supplied data. The message digest is used to perform a validity check on the data that has been delivered. In a system utilizing hash functions, the hash value is computed and stored alongside the input data. When the data needs to be retrieved, the hash function is applied again to generate a new hash value. By comparing the new hash value with the original hash value, it can be determined whether the data has been tampered with or remains unaltered. In addition, HFC is used to guarantee the accuracy of the data. The hash value is computed before the data is sent over the network and computed once again by the receiver after the data has been successfully received. If the two hash values are identical, the data has not been tampered with in transit and can be trusted.

Hash functions are a kind of cryptographic technique that, when given an input (a message or some data), generates an output of a defined size that is referred to as the hash value or the digest. They find widespread usage in a broad variety of cryptographic applications, including the verification of data integrity, the storing of passwords, digital signatures, and other similar uses. There are several subtypes or variants of hash functions, and each intended to suit different criteria or address distinct concerns about security. The following are some examples of common subtypes of hash functions:

Hash functions for use in cryptography: Hash functions used in cryptography are developed to provide additional layers of security. These layers include collision resistance, preimage resistance, and second preimage resistance. Because of these qualities, it is computationally impossible to discover two different inputs that create the same hash value, find an input that matches a given hash value, or locate a different input with the

same hash value. Additionally, it is computationally impossible to find two distinct inputs that produce the same hash value. Hash functions used in cryptography include SHA-256 (Secure Hash Algorithm 256-bit), SHA-3, and BLAKE2, as well as other variants.

Message Digest Algorithms: Message Digest Algorithms are a class of hash algorithms. Regardless of the input data size, they generate a hash result with a set amount of bytes. Typically, these algorithms are used in the process of validating the authenticity of data or communications. Message Digest 5 (MD5) and Secure Hash Function (SHA) are two examples for this type of algorithms. On the other hand, MD5 and SHA-1 are currently regarded to be weak and are susceptible to collision attacks; as a result, their use in new cryptographic applications is not advised.

Password Hashing Functions: Password hashing functions are designed to securely store passwords. They incorporate additional security features, such as key stretching and salting, to make it more difficult for attackers to recover the original passwords from the hash values. Examples of password hashing functions include bcrypt, Password-Based Key Derivation Function 2 (PBKDF2), and Argon2.

Keyed Hash Functions, sometimes referred to as Hash-Based Message Authentication Codes (HMAC), are hash functions that make use of a secret key in addition to the message that is being entered. They make it possible to use symmetric key cryptography to check the legitimacy of a communication and ensure that it has not been tampered with. Keyed hash functions include things like HMAC-SHA256 and HMAC-MD5, for instance.

Hash Functions with Variable Output Size: While most hash functions produce fixed-size hash values, there are hash functions that allow for variable output sizes. These functions are designed to provide flexibility regarding the desired hash value length. Examples include SHA-3 and BLAKE2, which can generate hash values of different lengths.

Advantages of HFC:

- Verification of Data Integrity: Hash functions are often used in the process of verifying the data's integrity. When a hash value is generated for a set of data, such as a message or a file, any future changes to the data will result in a new hash value

being generated for the set of data. It is possible to identify whether or not the data has been changed or tampered with by comparing the calculated hash value to the hash value that was originally used.

- Efficiency: Hash functions are very efficient when it comes to calculation, which enables the fast and reliable creation of hash values. Hash functions always create hash values of the same fixed size, regardless of the amount of the input data, which makes them efficient to both process and store.

- Hash functions are supposed to be one-way functions, which means obtaining the original input from its hash value is computationally impossible. This is because hash functions are one-way functions by design. This characteristic is essential for the storing of passwords since it guarantees that the original password cannot be readily discovered even in the event that the hash value is stolen or otherwise compromised.

- Message Authentication: Hash functions can provide message authentication, especially with symmetric key cryptography. By using a secret key to generate a keyed hash value, HMAC, both the authenticity and integrity of a message can be verified.

- Fixed Output Size: Most hash functions produce fixed-size hash values, regardless of the input size. This property ensures the hash values have a consistent length, making them easier to handle and store in various cryptographic protocols and systems.

- Non-Reversible: Hash functions are non-reversible, meaning the original input cannot be derived from the hash value alone. This property provides an additional layer of security in password storage, as even if an attacker gains access to the hash values, it is challenging to retrieve the original passwords without significant computational effort.

- Versatility: Hash functions are versatile and find applications in various cryptographic protocols and systems. They are used for data integrity checks, password storage, digital signatures, key derivation, and more. Their wide range of applications makes them a fundamental building block of modern cryptography.

Disadvantages of HFC:

- Collision Attacks: The HFC algorithm is susceptible to collision attacks, occurring when two input data sets return the identical hash result. This can be mitigated by using stronger hash functions and longer hash values.
- Limited Encryption: HFC does not provide encryption, which implies that the data is not secured from being seen or accessed by unauthorized parties.
- Limited Key Management: HFC does not require a secret key, meaning key management is limited.

2.2.4 Other Cryptographic Methods

2.2.4.1 Quantum cryptography

The concepts of quantum physics are used in a specific kind of cryptographic algorithm known as quantum cryptography. This sort of algorithm ensures that two parties may communicate safely. The basic concepts of quantum physics, including the uncertainty and superposition principles, serve as the foundation for developing quantum cryptography. Key distribution is the first step in quantum cryptography [31]. The sender and the receiver use a quantum channel to exchange quantum states, such as photons, to generate a shared secret key. The quantum states used for key distribution are often entangled, meaning they are connected so that any change to one state will be reflected in the other [32]. The receiver measures the quantum states sent by the sender to obtain the key bits. Any attempt to intercept the quantum states will alter the states and introduce errors that the receiver can detect. The receiver and the sender compare a small portion of the key bits to verify that the key is secure.

Figure 2.17 is an example of a schematic form used to describe quantum cryptography. The following example illustrates that one-way quantum cryptography was used to ensure the secure distribution of keys [33]. In this specific use of quantum cryptography, Alice and Bob can take advantage of the characteristics of quantum particles like photons to set up secure communication channels. One such protocol is known as Quantum Key

Distribution (QKD), which enables them to trade a secret key without running the danger of Eve listening in on their conversation. In this hypothetical situation, Alice wishes to send Bob a message that can only be read by Bob while Eve is listening in on their conversation. Alice concludes that the best way to send a stream of photons with arbitrary polarizations is to utilize a photon cannon [34]. Four distinct angles correlate to these polarizations: 0 degrees, 45 degrees, 90 degrees, and 135 degrees. Bob will choose a filter at random for each photon.

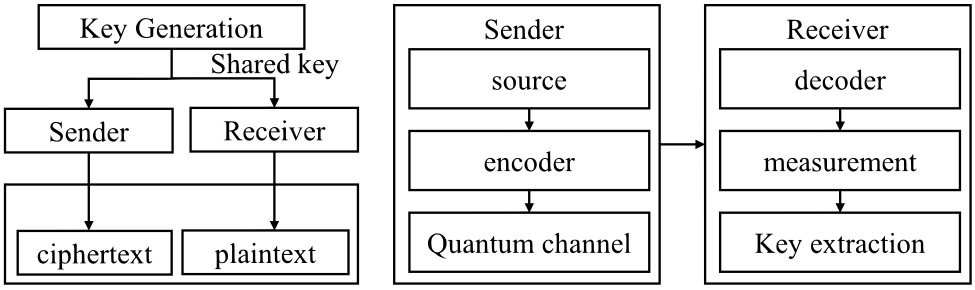


Figure 2.17: Quantum cryptography block diagram

After that, it will use a photon receiver to count the polarization, which will either be rectilinear or circular and measure it (with an angle of 0 or 90 degrees) or diagonal (with an angle of 45 or 135 degrees). After that, it will determine whether the measurements were correct regarding the polarizations that Alice selected and maintain a record of the outcomes based on that information. Even though a portion of the stream of photons will be lost due to the length of the connection, to generate a key sequence for a one-time pad [35], only a certain number of photons from the original stream are needed. After then, Bob will use an out-of-band communication method to inform Alice about the kind of measurement that was carried out and which measurements were appropriate. Despite this, Bob is not going to divulge the factual findings that were obtained from the measurements. The polarization of the photons whose measurements were found to be correct will have their bits converted, while the polarizations of the photons whose measurements were found to be wrong will be discarded. To ensure the security of the information being sent, a one-time pad is built with the assistance of these photons, which also act as the structure's basis. It is crucial to emphasize that neither Alice nor Bob can pre-determine the key in advance, as the key is generated based on random choices made by both. As a result, quantum cryptography makes it feasible to reliably exchange one-time keys and disseminate them.

Advantages of Quantum Cryptography:

- Security: Quantum cryptography provides a high level of security, as any attempt to intercept the quantum states will introduce errors that the receiver can detect.
- Authentication: Quantum cryptography can be used for authentication, as the shared key can be used to verify the identity of the sender and the receiver.
- Unbreakable Encryption: Quantum cryptography provides unbreakable encryption, as any attempt to intercept the quantum states will be detected, and the interceptor will not use the shared key.
- Future Proof: It is believed that quantum cryptography will be unbreakable since it is not susceptible to attacks from quantum computers, which are now in the research and development stage.

Disadvantages of Quantum Cryptography:

- Complexity: Quantum cryptography is complex and expensive, requiring specialized equipment and infrastructure.
- Range Limitations: Quantum cryptography has limitations, as the quantum states can only travel a certain distance before losing their integrity.
- Practical Limitations: Quantum cryptography is not practical for large-scale applications, such as Internet communication, due to the limitations of the technology.

2.2.4.2 Homomorphic encryption

In 1978, Rivest, Adleman, and Dertouzos were the ones who first presented the idea of homomorphic encryption [36]. However, actual implementations have just been available in the most recent few years. The secrecy of the data is maintained via homomorphic encryption algorithms, which also let calculations be conducted on the encrypted data without compromising the system's security. This has significant implications for

privacy and security in scenarios where sensitive data needs to be processed while minimizing exposure. IBM developed this type of encryption. This ensures the data is handled and examined without compromising its secrecy since it stays encrypted. To begin the process of homomorphic encryption, the data must first be encrypted using an encryption method. The encrypted data, also known as ciphertext, can only be decoded by the data owner using a special key that is kept in their possession. It is possible to execute calculations on the ciphertext using homomorphic encryption techniques without first needing to decode it [37]. This indicates that various mathematical operations were carried out on the ciphertext, including addition, multiplication, and comparison. Following completion of the calculation, the result will be sent back in an encrypted format. After that, the data owner may use the private key to decode the result to access the plaintext version of the result.

The schematic representation of homomorphic encryption was seen in Figure 2.18. It is a data encryption method that allows calculations to be carried out on encrypted information without first decrypting the data. This indicates that data was handled safely in an encrypted manner, even when it is being processed in a cloud environment or when it is being shared between different parties. Utilizing a partitioning strategy is a typical method when applying homomorphic encryption in a cloud context [38]. In this method, the data is first partitioned into many sections, then homomorphic encryption is applied to each section. After that, the encrypted partitions are dispersed among several cloud servers so that they are processed.

The cloud servers utilize homomorphic operations, which are calculations compatible with the encryption technique used to encrypt the data. This allows the cloud servers to execute computations on encrypted data. These procedures make it possible to execute calculations on encrypted data without first needing the data to be decrypted so that it can be accessed. The results of the calculations are encrypted using the same homomorphic encryption algorithm after completion [39]. The encrypted results are then transmitted back to the client, who may then decode the results to receive the final output. Because partitioning enables the computation to be divided over numerous servers, it is an effective method for implementing homomorphic encryption in a cloud setting. As a result, this method may minimize the time required for processing and increase overall performance.

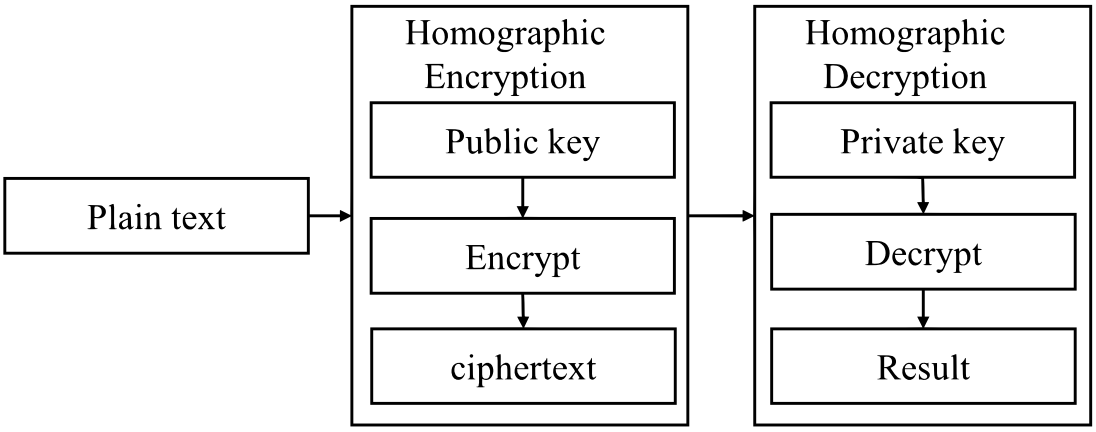


Figure 2.18: Homomorphic encryption block diagram

In addition, partitioning may improve system security by lessening the likelihood of a single point of failure or compromise inside the system.

In general, employing a cloud service and partitioning in conjunction with homomorphic encryption is a very effective method for ensuring the safety of data processing done in the cloud. Homomorphic encryption may help safeguard sensitive data while allowing for secure and efficient processing since it lets calculations be conducted on encrypted data. This makes the encryption method both secure and efficient.

Advantages of Homomorphic Encryption:

- Confidentiality: Because the data is kept encrypted throughout the computing process, homomorphic encryption offers an exceptionally high degree of secrecy.
- Privacy: Calculations were achieved on encrypted data using homomorphic encryption, but the data is not disclosed to other parties. This protects the data owner's right to personal privacy and allows homomorphic encryption.
- Security: Because the data is never decrypted during the calculation process, homomorphic encryption offers high security. This lowers the possibility of data breaches and unauthorized access.
- Flexibility: Homomorphic encryption techniques have a wide variety of potential uses, some of which include cloud computing, machine learning, and data analysis.

Disadvantages of Homomorphic Encryption:

- Computational Complexity: Homomorphic encryption algorithms are computationally complex, requiring significant processing power to perform computations on encrypted data.
- Limited Functionality: Homomorphic encryption algorithms have limited functionality compared to traditional encryption algorithms, as they only support a limited set of operations.
- Performance: Homomorphic encryption algorithms can be slow, especially when performing complex computations on large amounts of data.
- Key Management: Homomorphic encryption algorithms require the management of a secret key, which can be challenging and costly.

2.2.4.3 Obfuscation

Obfuscation-based cryptography [40] is a type of cryptography that aims to hide the functionality of a software program by obfuscating its code. The purpose of obfuscation is to make it difficult for potential attackers to comprehend how the program operates. This assists in defending the software against other forms of attack, such as reverse engineering. The structure of obfuscation-based cryptography is seen in Figure 2.19. The objective of employing obfuscation tactics is to increase the complexity of a design or system, thereby thwarting attacks. However, it is essential to ensure that despite the added complexity, the design or system retains the same level of functionality as its original version. Obfuscating the software application's source code is the first stage in implementing obfuscation-based encryption. This is done by modifying the code to make it difficult for attackers to understand the program's functionality. Obfuscation-based cryptography also involves encrypting the code of the software program [41].

This is done to protect the code from being read by attackers who may have access to the program. The encrypted code is decrypted at runtime by the software program [42]. This allows the program to run as usual but also helps protect it from being reverse-engineered by attackers. The technology has the capacity to offer security at every stage of the process of designing and manufacturing hardware. It has the potential to be used against piracy and tampering, and it has the potential to secure hardware that is protected

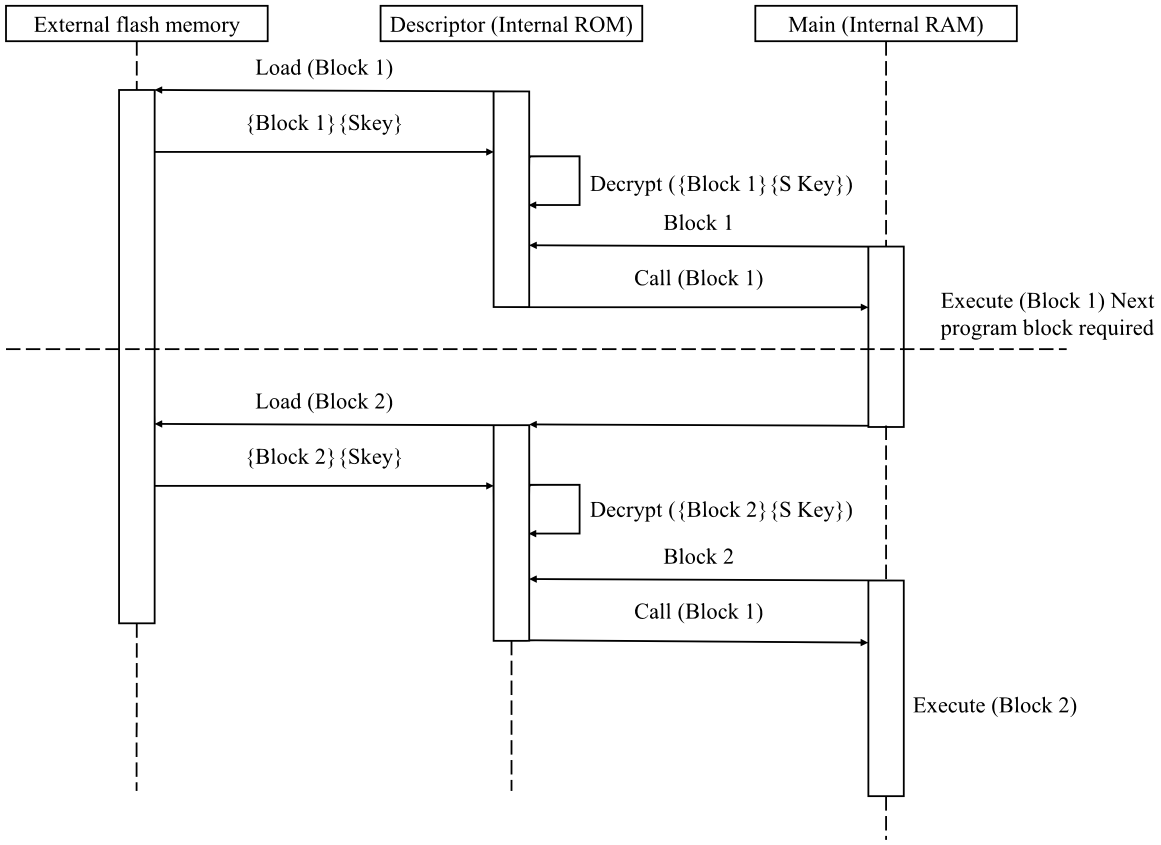


Figure 2.19: Obfuscation-based Cryptography block diagram

by the obfuscation of its netlist. Employing a technique that includes the methodical alteration of the gate-level IP core's internal logic structure and state-transition function is required to obfuscate the functioning of the gate-level IP core. This is done via the use of the method.

The circuit's transition from the obfuscated mode to the normal mode is triggered only by specific input vectors, commonly referred to as the "key" for the circuit [43]. This transition takes place when the circuit is given the key. When the circuit is turned on, only then will this take place. Furthermore, an interlocking obfuscation strategy was implemented in the design of the Register Transfer Level (RTL). This strategy allows the RTL to be unlocked and traverse a specific dynamic route when necessary. This was done to protect the design from being reverse-engineered. The circuit was powered up and operated in functional or entry modes (concealed). After forming a particular interlocked code word, the functional mode is activated so that it may perform its functions correctly. The information sent into the circuit is used as the basis for the encoding of the code word, followed by its implementation in entry mode so that functional mode is

reached. To protect the secret word from being revealed by reverse engineering, increased interaction with the state machine and interdependency on state-switching functions has been introduced. This means that the secret word is closely intertwined with the operations and transitions of the state machine, making it harder for an adversary to extract the secret word without understanding the underlying functionality of the circuit. Another advantage of interlocking obfuscation is that any modification or adjustment made by an opponent to the circuit will have a more significant impact due to the interdependencies created by the obfuscation technique. This increases the overall security of the circuit and makes it more resistant to reverse engineering attacks.

However, it is important to note that the level of protection provided by interlocking obfuscation must be balanced with the associated overhead. The technique may introduce additional complexity and computational overhead, which must be carefully managed to ensure that the benefits of increased protection outweigh the potential drawbacks [44]. Finally, interlocking obfuscation strengthens the protection of the secret word by increasing its interaction with the state machine and creating interdependencies with state-switching functions. It provides an additional level of protection for the circuit, making it more challenging for adversaries to extract the secret word through reverse engineering.

Advantages of Obfuscation-based Cryptography:

- Protection: Obfuscation-based cryptography protects against reverse engineering and other attacks that aim to understand the functionality of a software program.
- Customizability: Obfuscation-based cryptography can be customized to meet the specific needs of a software program, making it a flexible solution for protecting software.
- Compatibility: Obfuscation-based cryptography can be used with various programming languages and platforms, making it a versatile solution for protecting software.

Disadvantages of Obfuscation-based Cryptography:

- Limited Security: Obfuscation-based cryptography is not a completely secure solution for protecting software, as it is still vulnerable to side-channel and brute-force

attacks.

- Performance: Obfuscation-based cryptography can reduce the performance of a software program, as it adds extra overhead to the program.
- Debugging: Obfuscation-based cryptography can make it difficult to debug a software program, as the obfuscated code can be hard to understand.

2.3 LWC

The LWC [45] refers to designing cryptographic algorithms with minimal resource consumption. Because of this, they are suited for usage in contexts with limited resources, such as low-power devices such as smart cards, RFID tags, and IoT devices. The design goals of LWC include low-power consumption, small code size, and low memory usage [46]. Mathematical and functional analysis of LWC involves analyzing the design and properties of the cryptographic algorithms used in LWC, which generally use lightweight primitives such as block ciphers, hash functions, and authentication mechanisms.

Lightweight block ciphers are designed to be efficient regarding code size, memory usage, and power consumption. Examples of lightweight block ciphers include Simon, Speck, and Prince. These block ciphers exhibit variations in block sizes, key sizes, and the number of rounds they employ [47]. Each cipher is optimized to meet specific performance metrics based on the intended use case.

Hash functions are used in LWC to generate fixed-length output from an input of arbitrary size. They are used for message authentication, integrity verification, and digital signatures. Lightweight hash functions include PHOTON, SPONGENT, and GimliHash [48]. These hash functions are a solid compromise between security and performance since they are meant to be efficient regarding code size, memory use, and power consumption. Additionally, they create a decent balance between the two. In the LWC, authentication procedures are used to validate the identity of a message sender and to guarantee that the message's integrity is maintained. Examples of lightweight authentication mechanisms include HMAC-Based One-Time Password (HOTP) algorithm, **PACE!** (**PACE!**) protocol, and Privacy-Enhanced Robust Authentication Protocol (PERAP).

In addition to analyzing the design and properties of LWC algorithms, functional analysis involves analyzing the performance and security of these algorithms in real-world scenarios. This involves examining the resilience of LWC algorithms [49] against attacks such as differential and linear cryptanalysis, side-channel attacks, and fault attacks, among other types of attacks. The performance of LWC algorithms in terms of speed, power consumption, and memory utilization is also evaluated as part of functional analysis. Therefore, LWC offers an excellent compromise between security and speed in settings with limited resources. LWC algorithms, despite being less secure than their heavier counterparts, provide a workable alternative for securing low-power devices without negatively impacting their performance.

The design trade-offs for the LWC are shown in Figure 2.202.20. Currently with the IoT, it is very necessary to have different gadgets that are more compact, less expensive, and use less power. For instance, a cardiac implant surgically placed inside a patient's body must be compact and able to function for an extended period without requiring the battery to be recharged or replaced. These are often referred to as devices with limited resources, and ensuring their security is crucial for most of these devices. The fear is that the restricted resources on these devices may create performance difficulties when regular cryptographic algorithms are performed on them. This is because typical cryptographic methods need a significant amount of processing power. As a result, during the last several years, academics have been focusing on creating LWC and various other effective cryptographic systems [50].

Security, cost-effectiveness, and excellent performance are constraints imposed by its needs. The LWC project aims to explore these trade-offs and find solutions that balance performance, security, and cost-effectiveness for lightweight devices. Researchers [51] analyzed the requirements, constraints, and threat models associated with lightweight devices to propose cryptographic algorithms and protocols that meet the desired objectives. It is important to note that any adjustments made to key size, number of rounds, or system architecture should be carefully evaluated to ensure that the resulting level of security remains adequate for the intended application and threat environment. Striking the right balance requires a thorough analysis and consideration of the specific use cases and constraints involved [52].

The LWC project was initiated in 2013 by the National Institute of Standards and Technology (NIST) to achieve a better trade-off between performance and security for devices with limited resources. It involves evaluating existing cryptographic standards and developing new ones [53]. In 2018, NIST announced the entry rules and judging criteria for the LWC competition, which received algorithm submissions. One algorithm was disqualified for not meeting the criteria.

The subsequent phase in 2021 marked the final stage of the competition, where NIST revealed the selected competitors. Comprehensive specifications and reference implementations are essential to the established standards, including algorithm specifics, mathematical operations, design choices, and justifications. Each proposal must include an **AEAD!** (**AEAD!**) method and declare the security level and information on known cryptographic attacks. The LWC competition allows for meticulous evaluation and selection of efficient cryptographic standards suitable for lightweight devices.

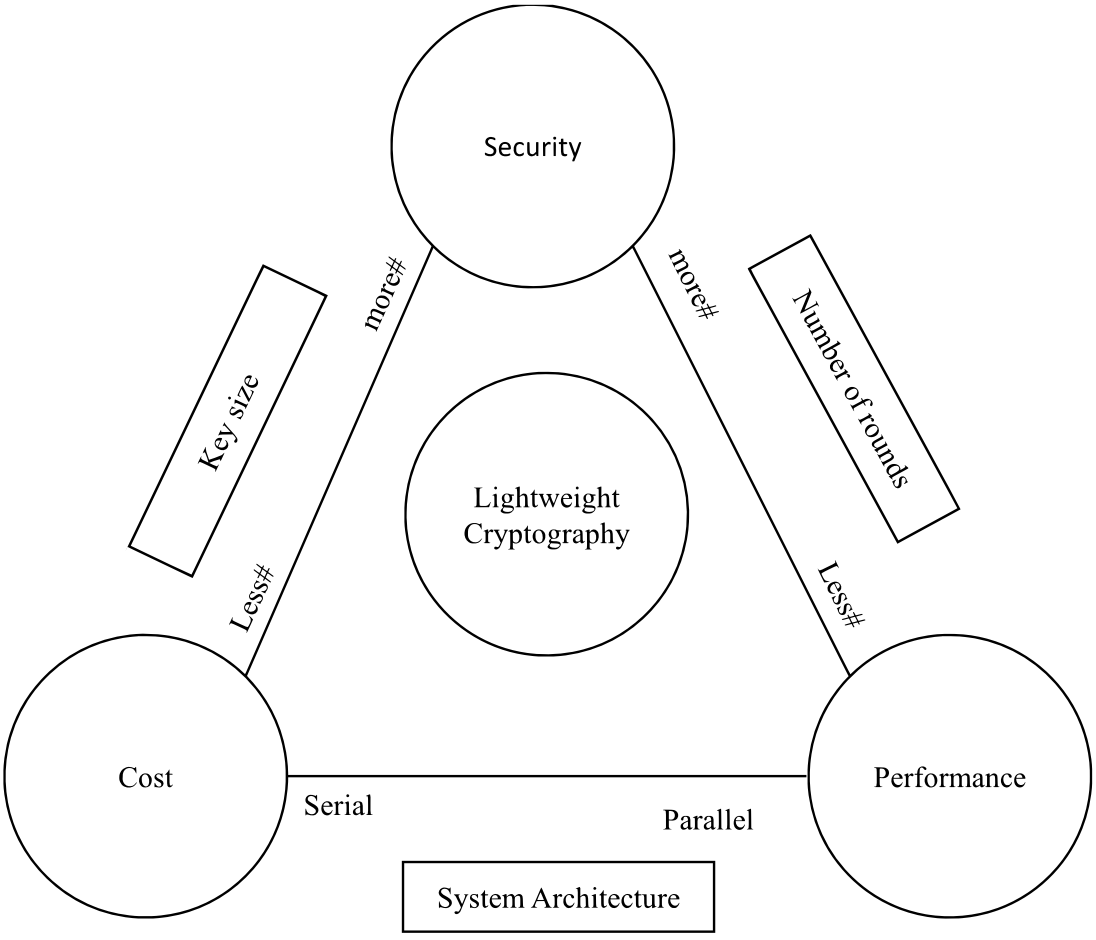


Figure 2.20: Design trade-offs for LWC

Advantages of LWC: LWC has several advantages over other cryptographic methods, especially in resource-constrained environments. Here are some of the main advantages:

- Low Resource Consumption: LWC algorithms are developed to be very efficient in terms of the amount of memory they use, the amount of code they need, and the amount of power they use. Because of this, they are well suited for use in low-power devices, such as RFID tags, smart cards, and IoT devices, which have limited resources [54].

- Faster Processing Speed: LWC algorithms are optimized for low-resource environments to perform encryption and decryption operations faster than traditional cryptographic methods. This is especially important in applications that require fast processing, such as real-time authentication and secure communications.

- Enhanced Security: LWC algorithms are designed to provide high security in resource-constrained environments. They use lightweight primitives resistant to common cryptographic attacks, such as differential and linear cryptanalysis, side-channel, and fault attacks. This makes them a more secure option than traditional cryptographic methods for low-power devices [55].

- Cost-Effective: LWC algorithms are often more cost-effective than traditional cryptographic methods. They require fewer hardware and software resources, making them less expensive to implement and maintain. They are also easier to integrate into existing systems, which can reduce costs further.

- Standards-Based: Both the ISO and the NIST have come together to create worldwide standards for safety and compatibility with different systems, which are complied with by the LWC algorithm design [56]. This makes them a reliable and widely accepted option for securing low-power devices.

- So, LWC provides a practical solution for securing low-power devices without compromising their performance or security. The advantages of LWC make it a compelling option for applications that require secure communication and authentication in resource-constrained environments.

2.4 Applications of LWC

LWC has several real-time applications in a wide range of industries. Here are some examples: IoMT Devices: LWC is ideal for securing IoMT devices with limited resources, such as sensors and actuators. These devices require lightweight security solutions that consume minimal power and memory, making LWC ideal, which can be used to secure IoT devices and networks, ensuring data privacy and integrity [57].

Smart Cards: Smart cards are widely used for secure authentication, identification, and payment transactions. LWC algorithms can be used to secure smart cards and enable fast and secure processing of transactions in real-time. This makes LWC an ideal choice for applications such as contactless payments, public transportation, and secure access control [58].

Automotive Industry: The automotive industry increasingly uses embedded systems to enable real-time communication between vehicles and infrastructure. LWC can be used to secure these systems, providing secure and reliable communication between vehicles and infrastructure. LWC can also secure in-car entertainment systems, ensuring data privacy and integrity [59].

Military and Defense: Military and defence applications require real-time secure and reliable communication. LWC can be used to secure communication between soldiers, vehicles, and aircraft, ensuring data privacy and integrity. LWC can also secure critical infrastructure, such as power grids and water treatment plants, ensuring national security [60].

Healthcare Industry: The area of healthcare is a huge and complicated sector that involves a diverse array of services, ranging from preventative care and diagnostics to treatment and rehabilitation. The healthcare industry is broad and complex. A diverse group of participants, such as healthcare providers, insurers, pharmaceutical firms, medical device manufacturers, and regulatory organizations, are also involved in the business. In this context, LWC can be a valuable tool for developing applications that help healthcare professionals and organizations to streamline their operations, improve patient outcomes, and comply with regulatory requirements [61].

Electronic Health Records (EHR): LWCs can be used to develop custom user interfaces for EHRs, digital versions of a patient's medical history. The LWCs can be designed to display data user-friendly and intuitively, making it easier for healthcare professionals to access and analyze patient data [62].

Medical Imaging: X-rays, MRI scans, and CT scans are just examples of medical pictures that medical experts may see and analyze with the help of apps that can be developed using LWCs. The LWCs can provide a user-friendly interface for viewing images, with features such as zooming, panning, and measurements [63].

Remote Patient Monitoring: LWCs can be used to develop applications allowing healthcare professionals to monitor patients' health and well-being remotely. The LWCs can integrate with various sensors and equipment, allowing for data collection such as a person's heart rate, blood pressure, and blood glucose levels, which can then be examined in real-time [64].

Healthcare Analytics: LWCs can be used to develop applications that provide healthcare organizations with insights into their operations and patient outcomes. The LWCs can be designed to display data interactively and visually, making it easier for healthcare professionals to identify trends, patterns, and opportunities for improvement [65].

Patient Engagement: LWCs can be used to develop applications that engage patients in their healthcare journey, providing information about their condition, treatment options, and progress. The LWCs can also allow patients to schedule appointments, communicate with healthcare professionals, and access their medical records [66].

2.5 Performance Metrics

After the goals mentioned above have been accomplished, the system's performance is assessed using the quantitative measures listed below.

Attack detection time (ADT): The ADT [67] in this system can be calculated using the following mathematical formula:

$$T = \frac{N - L + 1}{F}$$

Here, T represents ADT, N represents the number of input samples, L represents the length of the filter (i.e., the number of features extracted from each input sample), and F represents the number of filters used in the convolutional layer. In this formula, the input samples are first passed through the convolutional layer of the DLCNN, which extracts a set of features from each sample. The number of features extracted from each sample equals $(N - L + 1)$, where N is the number of input samples, and T is the length of the filter used in the convolutional layer. The features extracted from each sample are then passed through a fully connected layer, which predicts whether the sample represents an attack. The time required for this prediction is proportional to the number of filters used in the convolutional layer, denoted by F. Therefore, the overall ADT for DLCNN-based LWC can be calculated as the time required to pass the input samples through the convolutional layer plus the time required for the fully connected layer to generate a prediction.

Encryption time: The time required to complete the encryption operation on a message or data set using a certain encryption technique or scheme is called the encryption time [68]. The amount of time it takes to encrypt a message may vary depending on several variables like the length of the message, the level of difficulty of the encryption method, the processing power of the computer or other device-Being used for encryption, etc. The encryption time for LWC algorithms can be calculated using the following mathematical formula:

$$Encryption\ Time = (Number\ of\ Rounds) \times (Time\ per\ Round)$$

The "$NumberofEncryptionRounds$" represents the number of times the encryption algorithm iterates over the message to produce the encrypted output, and the "$TimeperRound$" represents the time the algorithm takes to perform one encryption round. The exact values of these parameters may vary depending on the specific LWC algorithm used, the size of the message, and the hardware or software platform used for encryption. However, LWC algorithms are generally designed to be fast and efficient, with encryption times typically measured in microseconds or milliseconds rather than seconds or minutes.

Decryption time: When discussing LWC, the term "decryption time" [69] refers to the amount of time required to carry out the decryption operation on a certain ciphertext by making use of a particular cryptographic method. The amount of time required to

decode a message was affected by various variables, including the size of the key, the size of the blocks, the number of rounds, and the computing performance of the decryption method. The formula for calculating the decryption time for a given ciphertext using an LWC algorithm can be expressed as follows:

$$Decryption\ Time = KST + BCDT + PPT$$

The Key Setup Time (KST) is the time required to set up the decryption key, which includes any key expansion or initialization steps. This time is typically a one-time cost and is not included in the overall decryption time for subsequent ciphertexts. The Block Cipher Decryption Time (BCDT) is the time required to decrypt each ciphertext block using the LWC block cipher.

This time is dependent on the block size and the number of rounds in the cipher. The Post-Processing Time (PPT) is the time required to perform any additional operations after the decryption operation, such as padding removal or verification of the decrypted data. So, the decryption time for an LWC algorithm depends on several factors, including the algorithm's complexity, the key size, and the length of the ciphertext. Choosing an LWC algorithm with an appropriate balance between security and computational efficiency is important to minimize decryption time while maintaining strong security.

Precision: It is the proportion of true positives among all positive predictions (i.e., detected attacks) [70]. A low accuracy score means the LWC algorithm is prone to false alarms or missed attacks. In contrast, a high precision score indicates that the system is excellent at identifying attacks with minimal false alarms. Therefore, evaluating the precision of an LWC algorithm in attack detection is an important step in assessing its overall security and reliability in real-world applications.

$$Precision = \frac{TP}{TP + FP}$$

A confusion matrix-based attack detection precision is a statistic used in the context of LWC to assess the efficacy of an LWC algorithm in identifying attacks such as fault attacks and side-channel attacks. In the context of attack detection, an LWC algorithm may provide one of four possible results: True Positives (TP), False Positives (FP), True Negatives (TN), False Negatives (FN). These four possibilities are included in the confusion matrix. The following definitions apply to these results:

- TP is the number of attacks that have been accurately identified.
- FP often known as false alarms, refer to the number of improperly detected attacks.
- TN is the number of non-attacks that have been appropriately recognized.
- FN is refer to the total amount of missed attacks.

Recall: It is the proportion of true positives among all actual positive cases (i.e., the total number of attacks) [71].

$$Recall \ = \frac{TP}{TP + FN}$$

F1-score: The $F1 - score$ is a metric used to evaluate the accuracy of a classification model, considering both precision and recall. It is particularly useful when dealing with imbalanced datasets, where there is a disparity between the number of positive and negative instances. By considering precision and recall, the $F1 - score$ provides a comprehensive measure of the model's performance. This is because the $F1 - score$ [72] incorporates both factors. A low $F1 - score$ suggests that a model is either skewed toward one of the two metrics or doing badly overall, whereas a high $F1 - score$ shows that a model has strong precision and recall. A high $F1 - score$ indicates that a model has good accuracy and recall.

$$F1 - score = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)}$$

Accuracy: This measure computes the percentage of accurate predictions produced by the model compared to the total number of predictions produced [73]. It may vary from 0 to 1, with 0 indicating that there is no accuracy and one suggesting that there is complete precision [74].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Peak Signal-to-Noise Ratio (PSNR): The $PSNR$ is a commonly used metric [75] for assessing the quality of an image by comparing the original image with its encrypted version. $PSNR$ represents the ratio between a signal's maximum strength and

noise's power that affects its quality. $PSNR$ provides a quantitative measure of the distortion introduced during encryption [76], helping to evaluate the fidelity of the encrypted image compared to the original. The formula for calculating $PSNR$ is as follows:

$$PSNR = 10 \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right)$$

Here, MAX refers to the highest potential pixel value of image [77], which is commonly 255 for 8-bit images, and MSE refers to the mean squared error between the encrypted version and the original. The MSE [78] is determined by taking the meaning of the squared differences in the pixel dimensions of the two compared images. The $PSNR$ is often represented in decibels (dB), a logarithmic number used to quantify the ratio of two values. The $PSNR$ value indicates the quality of the decrypted picture; a greater $PSNR$ number implies higher quality [79], while a lower $PSNR$ value suggests a poorer quality. The formula for MSE is as follows:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (x_i - y_i)^2$$

where N is the total number of values inside the pixels [80], and x_i and y_i are the values of the original and decrypted pixels that correspond to them, respectively. The MSE measures the average difference between the estimated values and the actual pictures. If the MSE number is lower, the original picture and the decrypted image match up more closely [81], whereas if the MSE value is higher, there is a greater disparity between the two.

SSIM: Structural Similarity Index (SSIM) [82] is a widely used metric for measuring the similarity between two images, particularly in image processing and computer vision. It aims to measure the difference in pixel values between the two images and the structural similarity in terms of luminance, contrast, and structure [83]. The formula for $SSIM$ is as follows:

$$SSIM = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{((\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2))}$$

Where μ_x and μ_y represent the means of the two pictures, σ_x^2 and σ_y^2 represent the variances of the images, and the variables σ_{xy} represents the covariance of the images, which is a measure of the relationship between the pixel values of two images [84]. Within

that formula, $C1$ and $C2$ are constants that stabilize the division operation when the denominator approaches zero. These constants prevent potential mathematical issues and ensure a stable computation of $PSNR$. The $SSIM$ measure generates a result between -1 and 1, with 1 indicating an exact structural match between the two pictures and -1 suggesting no structural match [85]. In actuality, the $SSIM$ values, on the other hand, are almost always in the range of 0 to 1. The $SSIM$ is a complex metric that considers the luminance, contrast, and structure of the compared images.

2.6 Summary

This research focuses on designing efficient cryptographic algorithms for resource-constrained devices such as smart cards, RFID tags, and wireless sensor nodes. This literature survey provides a detailed summary of the key developments in LWC. Most LWC research has concentrated on developing symmetric-key algorithms such as block ciphers, stream ciphers, and hash functions. The researchers devised several ideas for simple block ciphers, some of which are LBlock, PRESENT, and LED. These ciphers were optimized for hardware implementation and had low area and power consumption. Then, researchers proposed lightweight stream ciphers such as Grain-128 and Trivium, which were optimized for software implementation and had a low code size. Researchers have recently proposed lightweight hash functions such as SipHash and Gimli Hash, designed to have low computational and memory requirements.

Although PKC-LWC is typically more computationally intensive than symmetric-key cryptography, there has been some research in developing lightweight public-key algorithms. The researchers proposed the NTRU cryptosystem, designed to be efficient on resource-constrained devices. However, NTRU has some security weaknesses that limit its practical use. Researchers have recently proposed lightweight ECC versions, such as Ed25519 and Curve25519. These algorithms have smaller key sizes and lower computational requirements than traditional ECC algorithms.

LWC algorithms are often utilized in resource-constrained settings susceptible to side-channel attacks. These environments were broken into several different ways. Side-channel attacks use information released via physical channels such as timing variations,

power usage, and electromagnetic radiation. To address this issue, researchers have proposed various countermeasures, such as masking, which involves randomizing intermediate values during computation to prevent leakage. Other countermeasures include using constant-time algorithms and reducing the number of memory accesses.

The need for standardized LWC algorithms has led to various standardization efforts. The ISO/IEC established a working group to develop standards for LWC. The ISO/IEC suggested several different algorithms for the LWC, such as block ciphers, hash functions, and authenticated encryption techniques. Similarly, the NIST has initiated a contest to standardize LWC algorithms. As a result of this increased desire for safe communication in settings with limited resources, LWC has become an important topic of study. Developing efficient and secure LWC algorithms is crucial for the security of the IoT, IoMT and other embedded systems.

Chapter 3

ULWC Protocol with DLCNN for Heterogenous IoT Environment

3.1 Introduction

The ULWC is a cryptographic technique optimized for resource-constrained devices like those in the IoT environment. However, traditional security mechanisms such as signature-based, anomaly, and rule-based detection have limitations in detecting new and unknown attacks. Hence, there is a need to explore new methods for detecting attacks on IoT devices that utilize ULWC. In [86] authors alluded to previously presenting a communication protocol in the same article that utilizes the symmetric key-based scheme to offer very lightweight encryptions capable of properly safeguarding data transfers. This technique was proposed to deliver extremely lightweight encryptions capable of correctly securing data transfers. This is accomplished using the symmetric key-based technique, which results in relatively lightweight encryptions. This is being done to cut down on the overall amount of resources that are necessary to be used. The symmetric keys that are generated by using this protocol are delegated via a chaotic system that is known as the Logistic Map. This protects the symmetric keys against efforts to reset the key or seize the device. After that, they investigate the features of safety based on the semantic models of the relevant procedures. In addition to this, the consumption of resources is investigated to guarantee that the runtime will be successful.

In [87], the researchers aimed to design a unique chaotic encryption method that would incorporate exceptional dynamic features into an S-box. To achieve this, they devised a strategy using a six-dimensional known as Fractional Lorenz-Duffing Chaotic System (FLDSOP), along with an O-shaped route scrambling algorithm. The first requirement was to employ a six-dimensional FLDSOP to construct a prototype S-box. This system was chosen for its remarkable dynamic properties and significance in encryption.

The next step involved introducing a route scrambling scheme with an O shape. This strategy messed with the normal arrangement of components inside the S-box, which both increased the encryption method's level of complexity and strengthened its level of safety. In the end, the emphasis of the research was on developing a strategy for chaotic encryption by combining a six-dimensional FLDSOP with an O-shaped route scrambling algorithm. These techniques aimed to create an S-box with extraordinary dynamic properties, effectively disrupting the usual order of its components. Because of this, the components end up being reorganized in an unpredictably chaotic fashion. The research on the chaotic S-box has progressed owing to the contributions made by these chaotic systems. Each chaotic system has played an important role in studying the chaotic S-box. On the other hand, it is not straightforward to construct integral chaotic S-box generating systems that are acceptable for use in engineering projects that are being carried out. This is because integral chaotic S-box generating systems need much computational power.

In [88] authors presented a secure and fine-grained data access control system for limited IoT devices and cloud computing based on Ciphertext-Policy based ABE (CP-ABE). This technique was developed to regulate access to data in a more granular manner. This technique simplifies the process of key administration by incorporating a hierarchical structure within the attribute authority.

The mentioned study proposed the adoption of an outsourced encryption and decryption architecture to alleviate the computational burden on local computer resources. This approach aimed to minimize the strain imposed by the research. The system design involves transferring time-consuming activities to a gateway and a cloud server. This design choice offers several advantages, including efficient policy updating. With this

approach, the transmitting device can modify access restrictions without the need to recover and re-encrypt the data. This efficient policy updating capability is facilitated by the methodology employed in the system design. Because this technique allows both efficient policy updating and efficient policy administration, this is made feasible. As a result of the fact that this method enables effective policy updating, it is now possible to carry out the activity mentioned above.

In [89] authors presented their work for electronic health records employing Modified CP-ABE (MCP-ABE), which is used to keep information on the servers of third parties and exchange data with other parties. In addition, MCP-ABE is used for retrieving data from other parties. When EHR are outsourced to third-party servers such as the cloud, it might result in a few complications. The Internet and the "storage cloud" are examples of such servers. Protecting the privacy of a patient's medical records and other personally identifiable information is one of the challenges that must be faced.

The authors of [90] proposed using an LWC-based multi-factor AES, more often referred to as MF-AES, to authenticate the cloud-enabled IoT ecosystem. In this situation, critical information is split in two and encrypted using two different approaches. The RC6 algorithm is used to encrypt the first component of the data, and the Fiestel algorithm is used to encrypt the second half of the data. In addition, the AES encoding method was used to encode data that is not considered especially sensitive. In addition to this, AriaNN is a neural network training and inference solution for sensitive data. This solution safeguards users' privacy while only needing a small amount of user participation.

In [91] authors advocated hybrid LWC model using deep learning. The cryptographic mechanism of function secret sharing is used in two-party computing protocols. A lightweight and novel kind of encryption, this process is widely regarded as being one of its kind. Because of this, we can complete the step of the process that takes place online a lot more quickly than we would have been able to do under any other set of circumstances. Primitives like ReLU, MaxPool, and BatchNorm are central to optimising our attempts. These primitives play a crucial role in the building of neural networks.

In [92] authors explored deep learning for preimage attacks against different implementations of the Xoodyak hash mode. This lightweight hashing approach was presented to the NIST-LWC standards team during a meeting. The initial version of the Xoodyak

hash algorithm employed three distinct attack models, each with its own unique set of internal permutations and declining rounds. Training Deep Neural Networks (DNN) to target models with one round of underlying permutations makes it possible to predict the associated messages reliably. This training process, commonly known as "attacking models with one round of underlying permutations", involves instructing the networks to perform a "deep attack" on the models. However, it should be noted that DNNs exhibit lower accuracy and higher loss rates when exposed to various attack scenarios. This is because DNNs have more moving parts. This is since DNNs have a greater number of hidden nodes.

In [93], active S-box prediction is considered a regression problem. CNNs are trained to leverage various properties such as input-output changes, number of rounds, and permutation patterns to address this. These considerations are crucial for the successful completion of the task. Initially, the focus was on a reduced Generalized Feistel Structure (GFS) cipher consisting of four branches. However, it is worth noting that the Impact of each feature and its representation significantly influences the degree of prediction error, making it a more crucial factor than initially perceived.

The DLCNN-based attack detection in ULWC protocol for the IoT environment is a research topic that addresses security concerns in the IoT domain. IoT is an emerging technology that enables medical devices to collect and transmit health-related data wirelessly, which provides patients with remote health monitoring and physicians with real-time data analysis. However, IoT devices are vulnerable to cyber-attacks, and the sensitive medical data they collect and transmit must be protected from unauthorized access. So, this chapter presented the DLCNN-based approach that uses a deep learning algorithm to detect attacks on IoT devices using ULWC. The algorithm is trained on a dataset of network traffic captured from IoT devices and attacks generated in a lab environment. The DLCNN model learns to identify patterns and features in network traffic that distinguish between normal traffic and traffic generated by attacks. The DLCNN model consists of multiple layers of CNNs that extract features from the input traffic data. The output of the CNNs is then fed into a fully connected neural network that classifies the traffic as normal or attack traffic. The model is trained using supervised learning techniques and is optimized using backpropagation and stochastic gradient descent. The

proposed DLCNN-based approach has several advantages. It is lightweight and can be implemented on resource-constrained IoT devices. It is also effective in detecting known and unknown attacks, as the model is trained on a dataset that includes various attacks. Furthermore, it can adapt to changes in the traffic patterns of IoT devices, making it suitable for use in dynamic environments.

3.2 DLCNN-based ULWC for IoT Environment

The monitoring of industrial equipment, the military, agricultural regions, and many other fields are only a few examples of IoT applications which are used in practically every business. This IoT is vulnerable to being hacked and controlled by intruders, which would result in distorted data being sent. To avoid this problem, we must encrypt and decode the data used in Communication. Even though this technique requires a significant amount of processing and the trading of both public and private keys, it is necessary to safeguard the data used in Communication. IoT devices are often small, and they run on battery power. Consequently, they cannot do sophisticated calculations, and if they exchange keys, those keys may also be vulnerable to being stolen by cybercriminals. Therefore, this strategy for maintaining data security and privacy in a heterogeneous (different) IoT environment is successful since it adopts lightweight communication technology to facilitate the necessary connection.

Figure 3.1 illustrates the system model for the proposed symmetric key based ULWC protocol with DLCNN. An IoT network is first built with a sizable number of connected devices, a command-and-control node, a source device-A, and a destination device B selected randomly from the pool of accessible devices and many different devices. The "initialization" stage gets A and B to work together on the authentication procedure between them. In this scenario, every device has a unique Identification Number (ID).

The database has a complete copy of all the IDs associated with the devices. The Message Authentication Code, often known as MAC, is a phrase that is frequently used to refer to the ID of a device that is connected to the IoT. In the process known as initialization, the control center will generate the MAC for device-A by using both chaotic and logistic map-based characteristics. This will take place throughout the course of the

initialization method. It is common practice to use the chaotic method to satisfy the formal mathematical requirements of ULWC. The chaotic functions are to blame for the emergence of chaotic characteristics. They accomplish this goal using topological transitivity, ergodicity, starting conditions, and sensitive reliance. Over the last several years, cryptography has seen several effective applications of chaos theory. The freshly formed MAC is then validated against the ID of the local database as the next step in the process of starting up the machine. If a match is found, the initialization process will run via the control center to generate a successful authentication request, abbreviated ASR. After that, a request like this is issued to DLCNN to locate attack incidents.

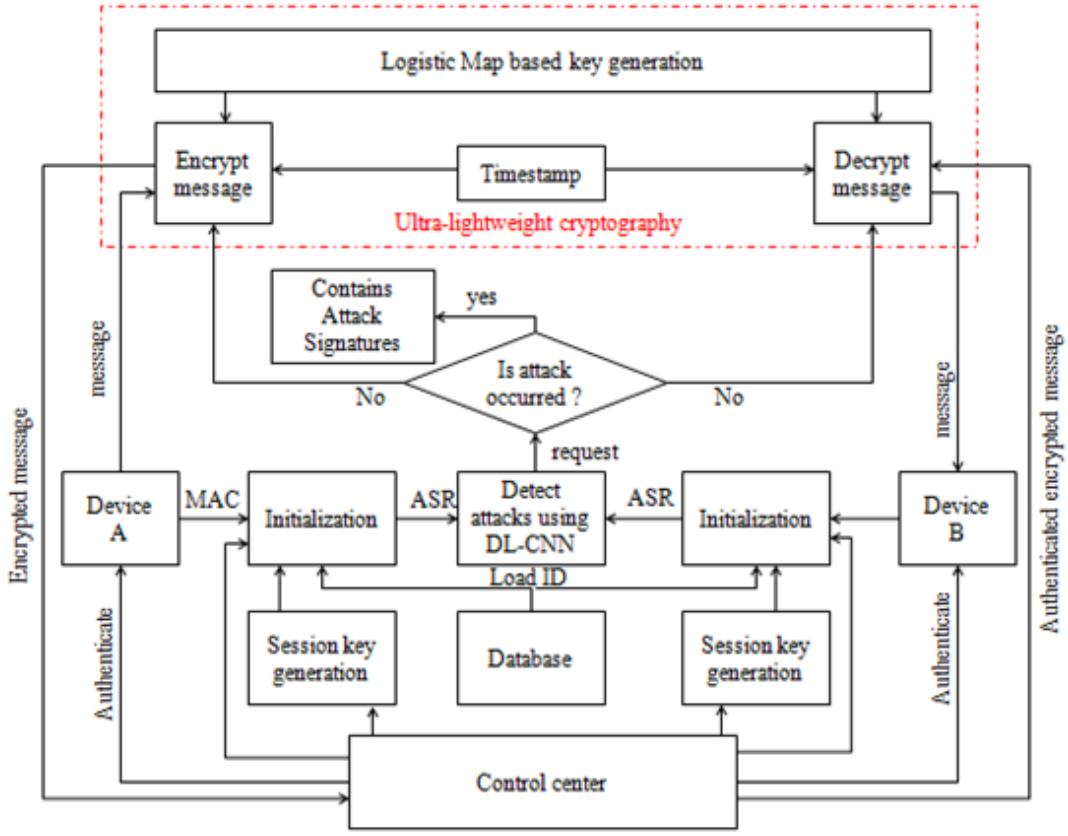


Figure 3.1: Proposed IoT network model

When comparing a test request from device-A with a pre-trained database, the DLCNN will classify the request as either a normal request or an attack signature based on its findings. Subsequently, device-A is validated by the control center and added to the DLCNN of the associated IoT device. A "Session Key generation" process establishes a session communication key between device-A, the Control Center, and device-B in the specified order. Once the Initialization of device-A is completed, the session key

is generated, enabling the IoT source to utilize this key for sending and receiving an unlimited number of messages within the established session. The session key will be created once the Initialization of Device-A has been properly done. It is also feasible to authenticate device-B with Control Center in the same manner, and creating a session key will result from this authentication process.

After the session key has been established, devices A and B can communicate and interact with one another. After that, the request from device-B is authorized by the control center and included in the DLCNN of the relevant IoT device. In the end, the DLCNN of device-B is used to verify the request made by device-B. Device-A will initiate symmetric key-based ULWC on message data using a timestamp and a logistic map-based key after the session has been successfully built up by authenticating the user. This will take place once the session has been properly set up. After the control center receives the encrypted message passed to the control center, it will verify both the device that was used to send the message, which is device-A, and the device that was used to receive the message, which is device-B. After the authentication procedure, the control center will send an encrypted message to the target device-B. The target device-B will then receive the message and decode the data it contains.

3.2.1 DLCNN Model

The DLCNN is a neural network architecture characterized by multiple layers of feature processing. It consists primarily of two layers: the nonlinear sub sampling layer and the convolutional layer. In the context of test images, the convolutional layers apply filters of various sizes, while the subsampling layers reduce the image size and feature complexity. The DLCNN also incorporates multinomial logistic regression layers and fully connected layers to enhance the logic of the network. These layers establish connections between the generated features from previous layers. At the kth stage, the bias b_k and filter W_k are utilized by the dense layers. The filters are applied to the input data $(X_{k-1}(i))$, generating the corresponding output feature $(X_k(j))$, which is determined by following equation:

$$X_k(j) = \sigma\left(\sum_{i\in\Omega(j)} X_{k-1}(i)\cdot W_k(i,j) + b_k(j)\right)$$

Here, convolution operation is indicated by $\cdot$, and nonlinear ReLU function is indicated by *Omega*. Further, the nonlinear sub-sampling layers operation was differentiated by

$$X_k(j) = X_{k-1}(j) \downarrow$$

Here, the region of the input data is denoted by $\omega(j)$, and its values are pooled using a sub-sampling function represented by $\sigma(.)$. Subsequently, the multinomial logistic regression layer is employed to compute the probability of the features X_L from the L^{th} layer corresponding to the i^{th} class. Finally, the SoftMax function is defined as follows:

$$y(i) = \frac{e^{X_L(i)}}{\sum_j e^{X_L(j)}}$$

The SoftMax classifier can limit the classification error created and approach the functioning of the desired function while keeping a low computational complexity. These two goals can be accomplished without significantly increasing the work that must be done. Figure 3.2 depicts the DLCNN potential organizational structure for data storage and retrieval. This design is used to detect the attacks carried out on the test data, and it is used to do so by analyzing the data. The various authentication requests generated by the source and destination IoT devices make up the test data in this scenario.

Following this, the requests are compared with the pre-trained COWRIE-IoT dataset to determine whether an attack occurred. The following is a comprehensive summary of the architecture suggested for the DLCNN, broken down by layer.

- The computation of neuron outputs is achieved by utilising the dense one layer, which is connected to the corresponding local region in the input image. A total of 512 filters are utilized to generate the weight coefficients. Each neuron performs a dot product operation between these weights and a small, interconnected test data feature. This allows the weight coefficients to be generated.

- The ReLU1 layer, indicated by red in Figure 3.2, serves as the activation function and performs an element-wise operation with a maximum threshold set to zero. As a result, the layer's dimensions will not change and will continue to be [512 x 512].

- The Dense-2 layer is 256 bytes and carries out the same operation as the Dense-1 layer. The number 256 indicates the size of the filter correspondingly.

- The ReLU2 layer executes the same procedure as the ReLU1 layer, and its dimensions have not changed; they are still [256 x 256].
- An attack happened status is achieved by using a fully connected layer, such as a SoftMax classifier, which is utilized to conduct the attack.

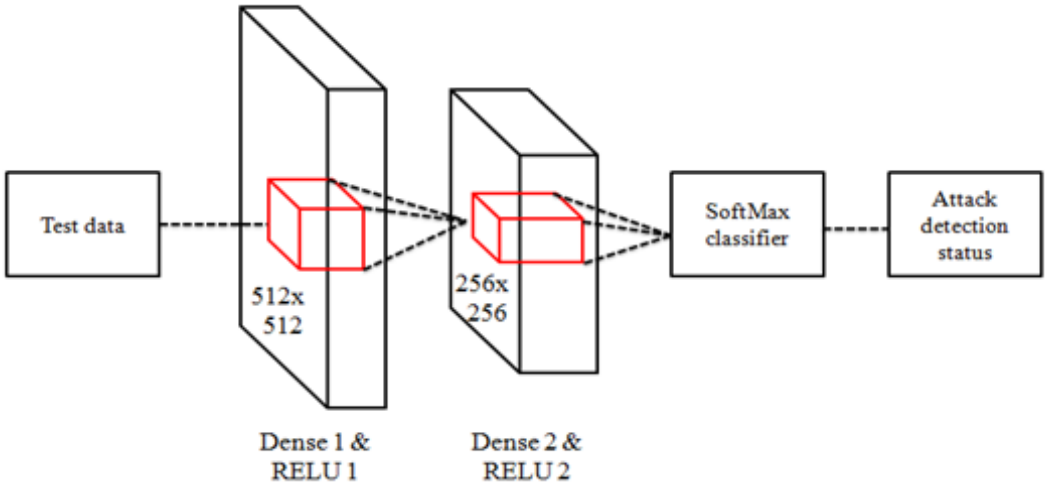


Figure 3.2: Proposed DLCNN architecture

3.3 Protocol Design

The suggested protocol included three entities, the Control Center S, Device-A, and Device-B respectively. In this case, the plaintext message is shown by the symbol M_*, and the devices unique identification is indicated by the symbol ID_*. For example, the identifier for device A is indicated by the symbol ID_A. The parameter chosen randomly is signified by the symbol μ_*, the symbol T_j^* represents the timestamp, and the initial key value that the Logistic Map determined is represented by the symbol x_0^*, respectively. This study assumed that the parameters are permanently encoded in the devices firmware since the manufacturer stores them somewhere where S can access them.

The notations $[-]_x$ and $h(-)$ indicate the encryption function, and the hash function. When this occurs, the Message Authentication Code is represented by MAC_*, and the symbol $< -, - >$ denotes the pair function. The proposed DLCNN-based ULWC system's protocol level operation is shown in Figure 3.3. This operation includes the phases of

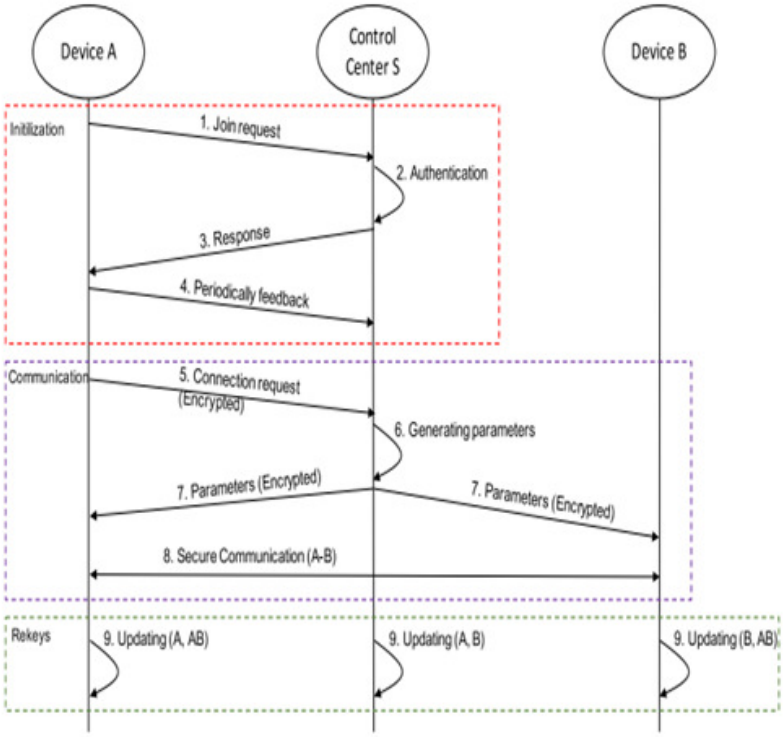


Figure 3.3: Proposed ULWC protocol

Initialization, Communication, and key updating (rekeys) respectively. During the startup period, Device-A tries, with the assistance of S to join the IoT network. A control center is used in this scenario to disseminate identities and keys to various IoT devices. If an IoT device intends to transfer data to another IoT device, both devices will provide their encrypted identities and MAC addresses to the control center. It will employ an advanced deep-learning technique to search for the IoT identity in the database.

Subsequently, the devices MAC code will be verified. Each IoT device is equipped with the DLCNN model implementation. When a new request is received, the DLCNN will test the request signature, predicting whether it is a normal request or contains attack values. Based on the trained models prediction, the device will decide whether to process the request. Device-A built-in DLCNN model typically initiates the process by sending a message to the control center, including device identifiers. If the control center successfully authenticates this message, it will respond with a confirmation indicating device-A is successful establishment of the network connection. In the following time, device-A will consistently establish a connection with the control center to either send or receive data. All the messages sent throughout the configuration process are encrypted

using the symmetric key generated using the built-in Logistic Map settings.

Once device-A successfully joins the network through deep learning MAC verification, if it wants to establish a connection with device-B, device-A needs to request a new set of initial values and parameters from the control center. This request is made when device-A attempts to connect with device-B. Subsequently, the control center utilizes the symmetric communication channel to transmit the parameters to both device-A and device-B. Calculation operations are performed on a temporary session key between devices A and B, updated using the newly obtained initial value and parameter pair.

Consequently, the proposed protocol ensures secure communication channels for Device-to-Device (D2D) and Device-to-Control-Center (D2C) scenarios. During the key updating phase, all entities involved in the protocol synchronize their operations to iterate their initialized values and keys generated by the Logistic Map. This synchronization occurs throughout the entire phase. Additionally, their timestamps are adjusted after devices A and B connect with the control center. This synchronization and adjustment of timestamps enhance the protocol's security and make it resilient against various attacks, including key reset attacks.

Key reset attacks are when an attacker tries to disrupt the key updating process by resetting or manipulating the keys in an asynchronous environment. However, the proposed protocol's synchronous operations and timestamp adjustments prevent such attacks from being successful. Accordingly at all entities are synchronized, and their timestamps are adjusted accordingly; the protocol maintains the integrity and security of the key updating phase. Overall, the proposed protocol's synchronous operations and timestamp adjustments provide robust protection against key reset attacks and other potential threats in an asynchronous environment that attackers could create.

3.3.1 Initialization

Device-A stores the three crucial pieces of information: the initial key x_A^0, the Logistic Map parameter M_A, and the unique identification ID_A in that order. Additionally, the current timestamp T_A^0 is required to authenticate requests when using DLCNN since it is the most efficient verification method for aliveness. The proposed step-by-step process

Step 1: The M_A initializes the Join message M_A, encrypts the message using key x_A^0 with T_A^0, and generates the cipher text $C_A = [M_A \| T_A^0]$.
Step 2: MAC_A is generated by A, i.e., $(MAC)_A = h(C_A \| \mu_A)$.
Step 3: Device-A transfers the message with entities C_A, MAC_A, and ID_A to S.
Step 4:S searches in the database to find μ_A and x_A^0 indexed by ID_A using DLCNN.
Step 5: The non-existent device error code is generated and sent to A if the ID_A is not found; otherwise, S verifies device A integrity by calculating the $h(C_A \| \mu_A)$.
Step 6:If $h(C_A \| \mu_A) == MAC_A$, then S decrypts C_A with key x_A^0 and generates the M_A and T_A. Else, S rejects the joining request of A by comparing the request with the database using DLCNN.
Step 7: If $\|T_S^i - T_A^i\| > time\_limit\_threshold(\gamma)$, then a list of device identifiers (L_{id}) of reachable devices are encrypted with T_S^0 using key x_A^0 and sent to A; else, S rejects the join request.

Table 3.1: Initialization algorithm

for the initialization algorithm is laid out in Table 3.1. This method is first activated by device A, and requests are sent to S.

Device-A connects to devise S to set up the encrypted communication environment or channel. The A is periodically linked to the S regularly to upload the data being watched (D_A). In this case, T_A and D_A are encrypted, and the new MAC is computed appropriately using A. At last, the ID_A, C_A, and MAC_A are sent to the S to complete the authentication process using the environment dedicated to deep learning.

However, due to the many timing constraints, A's timestamp (T_A) may, in most cases, be different from S's. In this scenario, the user can manually set and synchronize the timestamp in short-distance communication channels. For instance, users can specify the different environmental circumstances in Bluetooth and WLAN networks. Consequently, adversaries cannot mess with the timestamps during the startup step.

3.3.2 Establish Session Key

As shown in Table 3.2, every device that participates in the proposed protocol has a persistent key reserved for the exclusive purpose of D2C communication. This key is established during the phase known as Establish session key. When one device A attempts to connect to another, the first thing the device needs to do is inquire whether S can generate a session key for secure Communication. Both gadgets can store such a secure key for the long term and protect against attacks.

3.3.3 Communication Between Devices

Once both devices A and B have acknowledged x_0^{AB} and μ_{AB}, respectively, secure D2D connections may then be established between the two devices. In an unsecured environment, device B, the receiver, has the right to reject the connection request initiated by device A, while device A waits for device B to initiate the session. Another situation still involves a lack of secure Communication: "If device B's compute resources are depleted, then it will refuse device A's communication request."

The communication method that takes place between devices A and B is outlined in Table 3.3. In this algorithm, device-B generates a start message containing $C_{AB} = [M_B||T_B^i|], MAC_{AB} = h(C_{AB}||\mu_{AB})$, and ID_B and then transmits it to device A, respectively. After the first device, device A has received this message, the built-in DLCNN of device A is used to check the availability of the timestamp and the identity of device B is authenticated by verifying MAC. This results in the development of a secured communication channel. When an IoT device wishes to transfer data to another IoT device, both of those IoT devices will submit their encrypted identities and MAC addresses to the control centre, respectively.

In addition, the control centre will search for the identification of IoT devices in a database using a sophisticated deep-learning method. The MAC code of the device will then be classified (verified). Every IoT device will have an implementation of the DLCNN model. Upon receiving a new request, the DLCNN will test the request signature. This test predicts whether the new request signature is classified as normal or contains any indication of attack values. Based on the trained model's prediction, the device will decide

Step 1: Source device A reads the identifier of destination device B; thus, ID_B is read by A from L_{ID}.
Step 2: Device A encrypts ID_B with key x_A^i based on timestamp T_A^i and generates the ciphertext $C_A = [ID_B \| T_A^i]$.
Step 3: Device-A calculates $MAC_A = h(C_A \| \mu_A)$ for integrity verification.
Step 4: Device A initializes the communication with Device B by sending ID_A,C_A and MAC_A to S.
Step 5: S loads T_S^i, x_A^i, and μ_A and compares $h(C_A \| \mu_A)$ with MAC_A for integrity verification.
Step 6: Then, S authenticates the identity of A; here, DLCNN will predict whether the new authentication request is normal or contains attack values.
Step 7: If authentication succeeds, it certifies T_A and decrypts C_A.
Step 8: S generates the new initial value x_{AB}^0 and new parameter pair μ_{AB}, if decryption is done in the specified range of T_A.
Step 9: Then, S calculates $C_{AB} = [< x_A^0, \mu_{AB} > \| T_S^i]$ and $C_{BA} = [< x_B^0, \mu_{AB} > \| T_S^i]$.
Step 10: $MAC_{AB} \| C_{AB} \| ID_S$ and $MAC_{BA} \| C_{BA} \| ID_S$ are transferred to devices A and B to allocate the pair $< x_{AB}^0, \mu_{AB} >$.
Step 11: After receiving the above messages from devices A and B, then the authentication process is conducted by B, as presented in Table 3.1.
Step 12: If $(h(C_{BA} \| \mu_B) == MAC_{BA})$, then B verifies the integrity, decrypts C_{BA}, and extracts the T_S^i, μ_{AB}, and x_{AB}^0.
Step 13: Then verify the T_S^i, then B declares μ_{AB} and x_{AB}^0; they are again utilized for communicating with A.

Table 3.2: Session Key configuration Algorithm

whether to process the request. The index i used in symbols such as x_A^i related to the number of iterations performed on the Logistic Map. This indicates the number of times that x_*^0 was updated. Additionally, each repetition requires an updated key, and the key delegation is utilized to update and assign the keys in the Communication.

Step 1: Generate message M_B and initializes ID_B, T_B^i, μ_{AB}, and x_{AB}^0.
Step 2: Encrypt $C_{AB} = [M_B \| T_B^i]$, calculate $MAC_{AB} = h(C_{AB} \| \mu_{AB})$, and transfer $ID_B \| MAC_{AB} \| C_{AB}$ to device A.
Step 3: Device A loads T_A^i, x_{AB}^0, and μ_{AB} after receiving entities.
Step 4: Device A verifies the MAC, such as $h(C_{AB} \| \mu_{AB})$, by using DLCNN.
Step 5: If $(h(C_{AB} \| \mu_{AB}) == MAC_B)$, A verifies the integrity, decrypts C_{AB}, and extracts the T_B^i and M_B.
Step 6: Then verify the T_B^i, then A declares success and starts data transmission.

Table 3.3: Communication Protocol Algorithm

3.3.4 Key Delegation

It can be seen from Table 3.1, Table 3.2, and Table 3.3 that both the device that sends the message and the device that recovers the message are keeping exact timestamps to keep the communication channel safe. Furthermore, the IoT network faces challenges due to many unconnected devices. These devices typically lack sufficient resources, making it difficult for them to establish connections with the network. Therefore, even if these gadgets are not linked to each other, the eavesdropper may nonetheless impact them. In addition, the attackers can constantly listen in on conversations until the symmetric key used in the proposed protocol is disclosed. As a result, it is essential to monitor whether devices are connected to the internet.

Consequently, the keys must be updated consistently to maintain the confidentiality of the user data. As a result, this study uses a method known as a synchronous key update to maintain the uniqueness of the permanent symmetric keys. This could be accomplished by establishing a key phase. In addition, developing a synchronous system faces a significant challenge in inconsistent timestamps gathered on numerous devices. The key updating process relies on timestamps, so the proposed protocol includes the Key delegation process. This process generates keys iteratively, ensuring synchronization with timestamps. This is being done to tackle the difficulty mentioned above.

Here, the genuine timestamp is kept by the key delegation process at control center S, and Communication is not initiated until after the "feedback requests timestamps and

keys" have been matched. The date of the response to the request is the most important factor in protecting against attacks in this scenario. Consider t_f as the feedback interval, such as the timestamp for a request's feedback, and t_r as the rekey period, such as the time interval for key reimplementation. In addition, the temporal consistency was attained by keeping the condition t_r much greater than the t_f in a respective manner.

For instance, if the key is updated via key delegation every hour, the feedback interval was set to one minute, and the timestamp could be calibrated sixty times for each update. In this scenario, the devices will update the keys by calculating $x_{(i+1)} = \mu.x_i.(1 - x_i)$. If the rekeying time runs out, the devices will do nothing. In addition, the control center ensures the safety of the Communication by requiring that the session key x^i_{AB} be kept up to date until the expiry time, rekey time, and feedback interval are specified. In conclusion, the suggested protocol creates a safe ULWC by preserving these impeccable time stamps.

3.4 Results and Discussion

In this study, the performance of the proposed ULWC protocol is thoroughly investigated through various timing calculations. These calculations include encryption time, decryption time, key generation time, signature creation time, and signature verification time. The performance of the proposed protocol is compared with several standard techniques, such as Ariann, MF-AES, CP-ABE, MCP-ABE, ECC, RSA, and LWC-Privacy-Preserving, in terms of key size and message size. Furthermore, the performance of traditional techniques like Ariann, DNN, and CNN for attack detection is contrasted with the performance of the DLCNN based attack detection of the proposed method. To train the DLCNN model, the COWRIE IoT dataset is utilized. This dataset contains MAC-based telemetry data from various IoT services, as well as network traffic and operating system logs. This investigation aims to assess and demonstrate the effectiveness and efficiency of the proposed ULWC protocol and DLCNN-based attack detection compared to existing techniques and datasets.

3.4.1 Dataset

The COWRIE-IoT dataset collects network traffic captures made accessible to the public. These captures were made using a honeypot meant to simulate IoT devices. Researchers from New York University's Tandon School of Engineering's Center for Cybersecurity are responsible for compiling the dataset. The honeypot used in the COWRIE-IoT dataset was designed to mimic the behavior of IoT devices such as routers, cameras, and smart home devices. The honeypot was deployed in a cloud environment and exposed to the Internet to attract attackers. The attackers interacted with the honeypot, and their activities were captured as network traffic captures. The dataset contains network traffic captures from attacks attempted on the honeypot over several months.

The attacks included various types of malware infections, brute force attacks, and exploitation attempts. The dataset includes both raw network traffic captures and preprocessed data in the form of CSV files. The COWRIE-IoT dataset is useful for researchers and practitioners who are interested in studying the behavior of attackers targeting IoT devices. The dataset can be used to develop and test new detection and prevention techniques for IoT security. It can also be used to evaluate the effectiveness of existing security solutions. However, the following are some common columns that are presented in the dataset:

- "Time stamp": A timestamp indicating the time of the network event, typically in the format of "YYYY-MM-DD HH:MM:SS.ssssss."
- "Source ip": The IP address of the source of the network event.
- "Destination ip": The IP address of the destination of the network event.
- "Source port": The port number used by the source of the network event.
- "Destination port": The port number used by the destination of the network event.
- "Protocol": The network protocol used for the event, such as TCP or UDP.
- "Session id": A unique identifier for the network session.
- "Username": The username used in the event, if applicable.

- "Password": The password used in the event, if applicable.
- "Command": The command issued in the event, if applicable.

These are just a few examples of the columns in the COWRIE-IoT dataset. The exact columns and their meanings may vary depending on the specific file. It is recommended to refer to the dataset documentation or metadata for more detailed information on the columns present in each file.

3.4.2 Influence on the ADT

In this section, a thorough analysis of the performance of the proposed method is conducted, comparing it to various state-of-the-art methods across different timing levels. The comparison aims to highlight the proposed method's advantages over existing approaches. One comparison is with traditional public key approaches, which typically require significantly more resources. This is mainly due to the higher I/O cost of PKC compared to symmetric encryption approaches. The proposed protocol, on the other hand, addresses this issue by introducing a DLCNN-based caching technique at the control centre. This technique helps to reduce the computation resources required by resource-constrained IoT devices. By utilizing the DLCNN-based caching technique, the proposed method achieves improved efficiency and reduces the computational burden on IoT devices. This allows for more efficient resource utilization and better performance than traditional public-key approaches. This allows for a significant increase in efficiency. Therefore, the suggested protocol considerably reduces the total time cost, and it identifies the attacks that have taken place in the environment at a high rate of speed.

Table 3.4 demonstrates that the proposed method detected the attacks quickly and efficiently while also requiring less time, in comparison to the conventional Ariann [91], MF-AES [90] in the presence of various covert attacks, and Lightweight Privacy-Preserving [86] techniques. This is shown by the fact that the table displays the results. These comparisons were made using the time it took for those methods to identify the attacks. The ULWC with DLCNN that was presented offers increased protection against brute force, key reset, and device capture threats. It also offers layered authentication by preventing malevolent users from accessing the network.

Method	ADT (ms)
Ariann [91]	0.56
MF-AES [90]	0.47
LWC- Privacy- Preserving [86]	0.0015
Proposed ULWC Protocol	0.001

Table 3.4: ADT comparision of various LWC methods

3.4.3 Impact on Encryption and Decryption time

In the following section, we examine the encryption and decryption times for various message sizes and compare them to state-of-the-art techniques. The encryption time refers to the time required to transform a message into ciphertext. In contrast, the decryption time refers to the time required to transform ciphertext back into the original message. One inherent issue with ULWC is that larger message sizes often result in increased encryption and decryption times. To address this issue, the proposed method incorporates DLCNN, which effectively controls the protocol timestamps and ensures constant encryption and decryption timings regardless of message size. Table 3.5 demonstrates that the proposed ULWC outperforms CP-ABE [88], MCP-ABE [89], and MF-AES [90] in terms of Encryption time measured for various message sizes (ETVMS) and Decryption time measured for various message sizes (DTVMS). This indicates that the suggested ULWC approach is faster. Moreover, the recommended method generates the public and private keys simultaneously. Additionally, the key sizes are automatically adjusted by attribute authorities based on the message size. This leads to a reduction in encryption and decryption time, ultimately resulting in shorter overall transaction times.

Furthermore, Figure 3.4 illustrates the estimated encryption time for ten independent users in an IoT context. Figure 3.5 shows the estimated decryption time for messages sent by ten randomly selected users within the same IoT network. In scenarios involving multiple users and authorities, the ULWC approach showcased decreased encryption and decryption time. On the other hand, the traditional CP-ABE [39] technique encountered challenges with its access policy, leading to an unnatural increase in the time required to generate keys for individual users and resulting in longer cryptographic times.

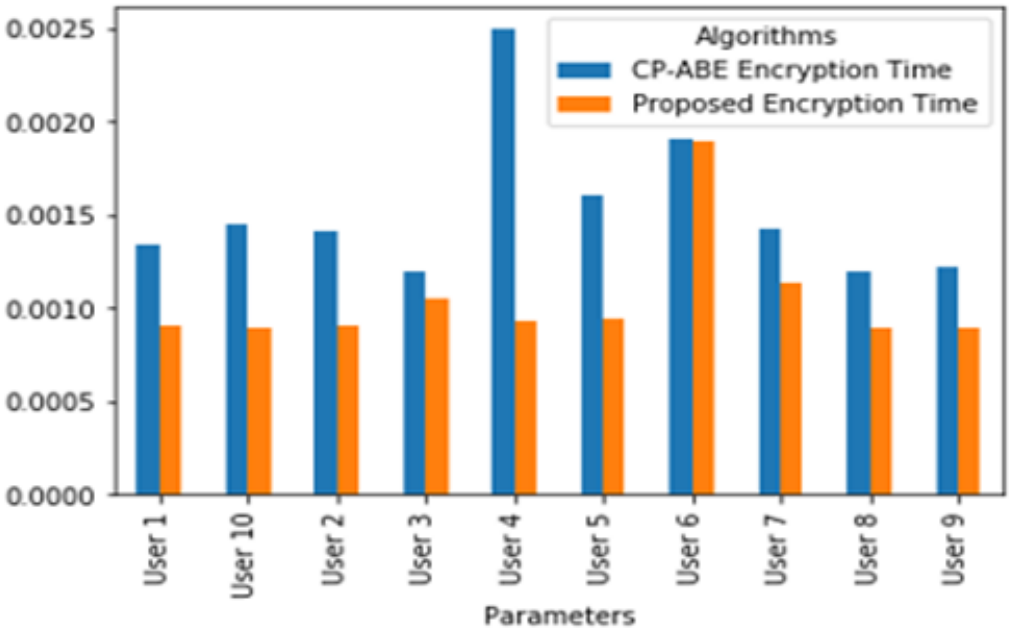


Figure 3.4: Analysis of the encryption time for ten users

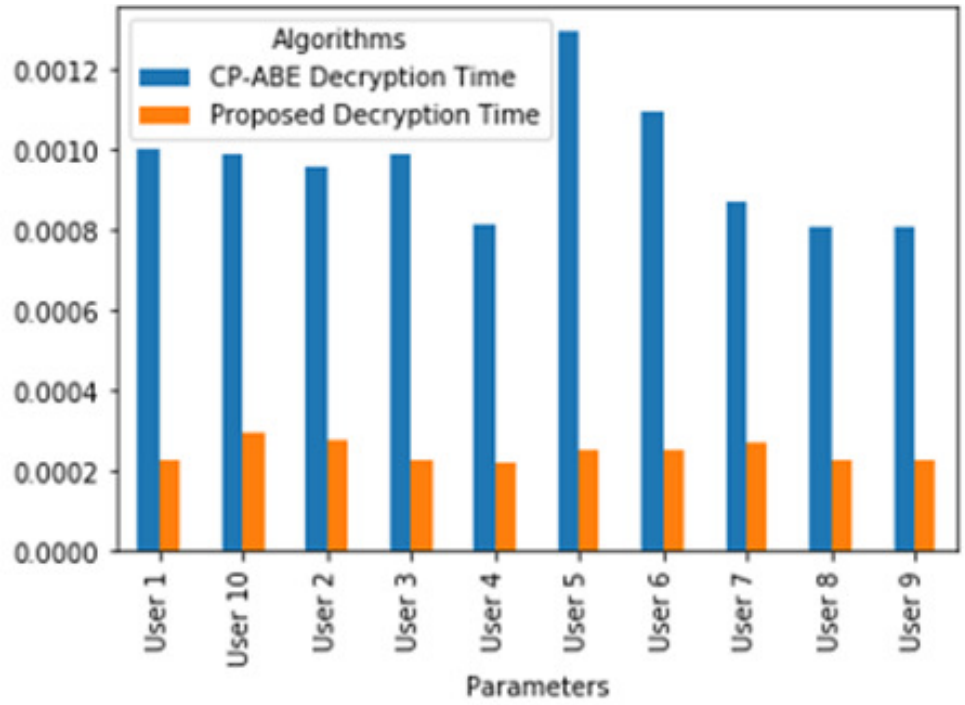


Figure 3.5: Analysis of the decryption times for ten users

3.4.4 Impact of attack detection performance

The proposed DLCNN-based ULWC for the IoT would have its level of security in the IoT environment measured based on the attack detection performance criteria. As a

Method	ETVMS=100	ETVMS=200	DTVMS=100	DTVMS=200
Proposed ULWC protocol	0.00098	0.00053	0.00075	0.00028
CP-ABE [88]	0.09259	0.06851	0.0766	0.09399
MCP-ABE [89]	0.07253	0.05963	0.0667	0.07916
MF-AES [90]	0.06855	0.04818	0.0487	0.06019

Table 3.5: Comparison of encryption and decryption times based on message sizes

result, the precision, recall, F-measure, and accuracy of attack detection were computed in this study and compared to the traditionally used methodologies. Figure 3.6 depicts a comparison of the relative performances of several contemporary LWC algorithms, including Arianna [91], DNN [92], and CNN [93]. These algorithms were used to obtain attack detection quality metrics. This comparison is made about the DLCNN-equipped ULWC method that has been presented. The conventional Ariann [91] approach uses various characteristics to authenticate authorized users and provide security for training data. Bilinear pairing, which encrypts such enormous quantities of data, involves multiple complicated multiplication operations, which results in a long computation time and poor attack detection performance because of the complexity of these operations.

Similarly, DNN [92] and CNN [93] employ multi-attribute data and an inefficient approach to conducting encryption and decryption operations, which lowers the threat detection performance of these networks. The outcome of this was that the proposed approach could differentiate between the various attacks with a high degree of accuracy and efficiency compared to the techniques now regarded as state-of-the-art.

3.5 Summary

A ULWC designed specifically for usage in an IoT environment, complete with an integrated DLCNN technique and a symmetric key-based communication protocol, has been created in this chapter. Within the scope of this task, the logistic map procedure was used to put into effect the primary delegation strategy. This approach uses random generation to develop an initial value for a parameter. Consequently, the control center is responsible for the default setup of each IoT device. This randomly generated key is

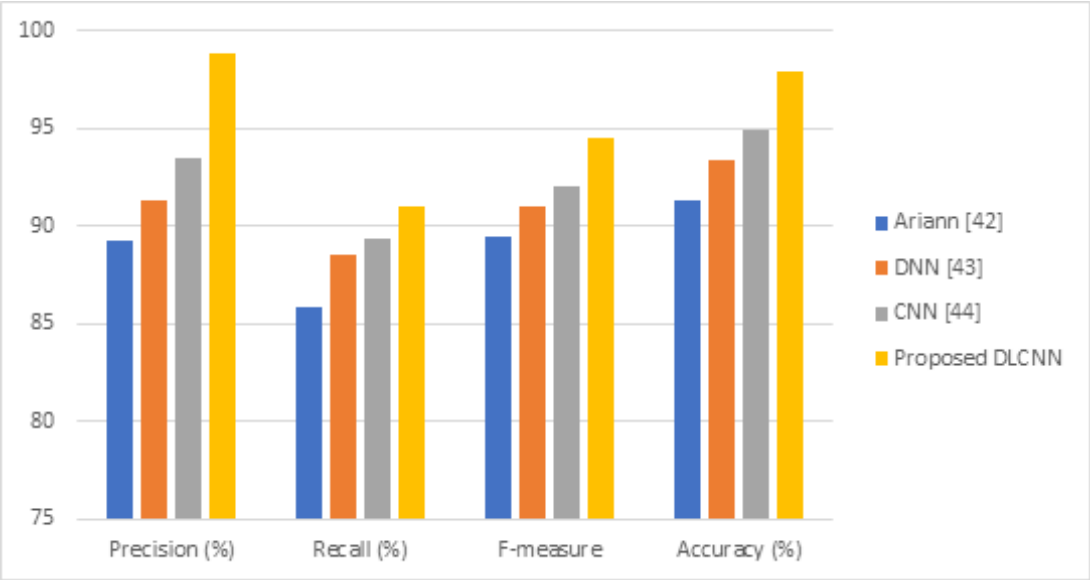


Figure 3.6: Attack detection Performance comparison of various LWC models

used for authentication in situations where an IoT device is primarily associated with a control center, such as in an environment that uses cloud computing. After that, this control center will be able to provide another key pair for D2D connection based on the pre-trained DLCNN that will be implanted into each unique IoT device. This will allow for more secure and private communications between the IoT devices. The already pre-trained model processes the new communication request sent by the IoT device. Next, the DLCNN model determines whether the new communication request is normal or includes any attack signatures based on whether it contains any of those signatures. At long last, a safe way to communicate between IoT devices, from the source to the destination, all while being monitored by a system that uses deep learning. The results of the simulation show that the proposed approach is more effective in terms of performance than the conventional techniques for metrics such as ADT, encryption time, decryption time, and attack detection confusion matrix. This conclusion was reached as a direct consequence of the simulation's findings. However, this work can be extended with ABE for more security with reduced time complexity.

Chapter 4

Hybrid LWC with Attribute-Based Encryption for secure and scalable IoT system

4.1 Introduction

IoT technologies and wireless communications have grown exponentially in recent years, and users have increasingly turned to less-weight and more compact computer devices. The reasons for this are that these technologies are less expensive, more compact, more powerful, and efficient enough to handle in their respective capacities. Additionally, it is good knowledge that resource-constrained technologies, such as RFID tags, contactless smart cards, smartphones, wireless patient monitoring systems, and wireless sensor networks, are prone to higher security problems. One of the most significant advancements in information and communication technology Information and Communication Technology (ICT) is reducing the size of individual components while preserving the functionality they had in the past. Because of this, advancements in information and communications technology are making it possible for new services to emerge that use the shrinking size of computer equipment.

The future of information and communications technology in this race comprises the consolidation of numerous applications into a single universally small device-And the development of several restricted devices capable of talking with one another across a network. The previous work focused on attack detection using DLCNN, but they did not implement the attribute-based policy mechanism, which can reduce the time complexities.

In [94] authors described ciphertext-policy weighted ABE, also known as CP-WABE, as having the ability to provide data security in the IoT. However, there are still a lot of problems, such as inflexibility, inadequate processing capabilities, and poor storage economy when comparing properties. They offer a unique access policy expression approach that uses 0-1 coding technology to address these problems. Using this strategy as a foundation, a CP-WABE that is adaptable and effective is built for the IoHT. It is possible to compare weighted qualities using their system, but their system also supports weighted attributes. In addition, they utilize offline/online encryption and technology that is outsourced for decryption to guarantee that the scheme can be executed on an inefficient IoT terminal. The theoretical and empirical evaluations demonstrate that their plan is superior to others in its efficacy and practicability. In addition, the findings of the security investigation point to the fact that their method is safe even when subjected to a chosen-plaintext attack.

Based on a policy-hiding attribute-based keyword search and data sharing scheme Policy-Hiding Attribute-Based Keyword Search and Data Sharing Scheme (PH-ABKS-DS) environment, the authors of the work [95] developed a trilinear pairing map with a rank of 3 on limited free R-modules (R is a commutative ring). The map has a limited number of free R-modules. This map was constructed via a policy-hiding attribute-based keyword search and a data-sharing method. Setting up multiple keys was used to establish independence from hashing. A digital signature technique is not required for user validation, which is another plus for this method. On the other hand, these systems have issues with maintaining policies and producing tokens, making them less than ideal.

In addition, Reversible Multi-Authority-Based ABE (RMA-ABE) [96] has been developed as a solution to the expressiveness concerns that are inherent in the standard Revocable Attribute-based Encryption with Data Integrity (RABE-DI) and Unbounded and Efficient Revocable based ABE (UER-ABE) systems. These issues were resolved by reversing the authorization granted by the RMA-ABE. This strategy implements not one but two straightforward cryptographic systems, one of which is not dependent on any hash function, while the other is dependent on such functions. In addition, revocable CP-ABE [97] has been developed as a solution to the computational complexity problems associated with conventional revocable methods. These problems are intrinsic to classic

revocable techniques. This technique significantly lessens the risk of a repayable chosen-ciphertext attack. Because it uses self-pairing in conjunction with ABE, this approach subsequently makes use of the fact that it is more lightweight than the one that came before it.

In addition, the secure decision of membership Secure Decision of Membership (SDM) protocols is implemented in the Dual Membership-based ABE (DM-ABE) [98]. After the pairs have been formed correctly, they can form fields of finite order. These fields were sufficiently big to make solving the discrete logarithm issue computationally challenging while simultaneously being tiny enough to allow for cheap calculations.

As a result, the primary emphasis of this study is on implementing the LWC-ABE technique, which uses the ChaCha and Playfair encryptions. The proposed LWC-ABE approach offers several advantages, such as high expressiveness, the ability to change access policies, support for large attribute domains, and white box traceability. However, one limitation of this approach is that it relies on multiple trusted authority environments, which can hinder the flexibility of IoT servers and devices to modify their access policies. The LWC-ABE approach provides a high level of expressiveness, allowing for fine-grained access control based on attributes. It supports the dynamic modification of access policies, enabling changes to be made as needed in IoT environments. The approach also handles a wide range of attribute domains, accommodating diverse sets of attributes in access policies. The proposed method resulted in higher security standards than state-of-the-art approaches.

4.2 LWC-ABE Method

LWC-ABE is a method that enables private communication between multiple parties over a communication network, protecting the information from eavesdroppers. It involves encryption and decryption processes, where a plaintext message is encoded into ciphertext during encryption and decoded back into the original message during decryption. Cryptographic systems typically rely on either a pair of keys (public and private) or a single shared key to perform encryption and decryption operations. The proposed LWC-ABE approach offers several key features. It provides high expressiveness, allowing

for fine-grained access control and flexible definition of access policies.

Access policies can be modified or updated, providing adaptability in different scenarios. The approach supports broad attribute domains, accommodating diverse attributes that can be used to define access policies. Another important characteristic of the LWC-ABE approach is white box traceability. This feature enables tracking and auditing of access and usage of encrypted data, enhancing accountability and providing a clear audit trail. So, the LWC-ABE offers strong security, flexibility in access control, and the ability to track and monitor access to sensitive information.

- A big attribute domain: There is a correlation between the number of authorized institutions and the size of the public parameters; however, this correlation does not rise linearly concerning the characteristics. If the system is constructed, then there may or may not be any need to update the properties of the system.
- Modification of the policy: To maintain compliance with increasingly stringent safety requirements, the data owners (s) are always producing new ciphertexts and updating the policy access requirements. In addition, following the updated policy, the data owners may modify the data access characteristics flexibly so that it better meets their requirements.
- White box traceability: The system can keep an eye out for nasty users who distribute private keys in an unauthorized manner. This characteristic is referred to as "white box traceability." Because white box traceability produces a list of users together with those individuals' access privileges, it was possible to determine which users lacked authorization with a small amount of computing effort.
- Many approved authorities: The problems with data integrity are solved by implementing a system with numerous authorities, which simultaneously solves the problem of the insufficient credibility of a single authority. By implementing a system with several authorities, the issues about the integrity of the data were resolved.
- Expressivity: It permits any repetitive access structure in addition to a flexible access control access strategy.

Figure 4.1 depicts the proposed LWC-ABE framework, and its important operational

components include the trusted party, the system party, the data consumers, the data owners, the attribute authority, and the cloud storage providers Cloud Storage Providers (CSP). These components make up the framework. The system setup method must first be called for the public parameters to be defined. In addition, certain parameters accessible to the public are first distributed to trusted parties, data consumers, data owners, and attribute authorities.

In the proposed method, attribute authorities are crucial in generating public keys and distributing them to trusted parties, data users, and owners. These attribute authorities assign attributes to data users based on their requests, ensuring only authorized users can access specific attributes. The data owner encrypts it using the ChaCha encryption algorithm and sends it to the CSP. The CSP further encrypts the data using the Playfair algorithm before storing it in cloud storage. The data owner also generates a policy updation key, which is provided to the CSP when there is a need to change the access policy. The data owner can choose either ChaCha or Playfair encryption for their data, and they can even use both encryption techniques simultaneously.

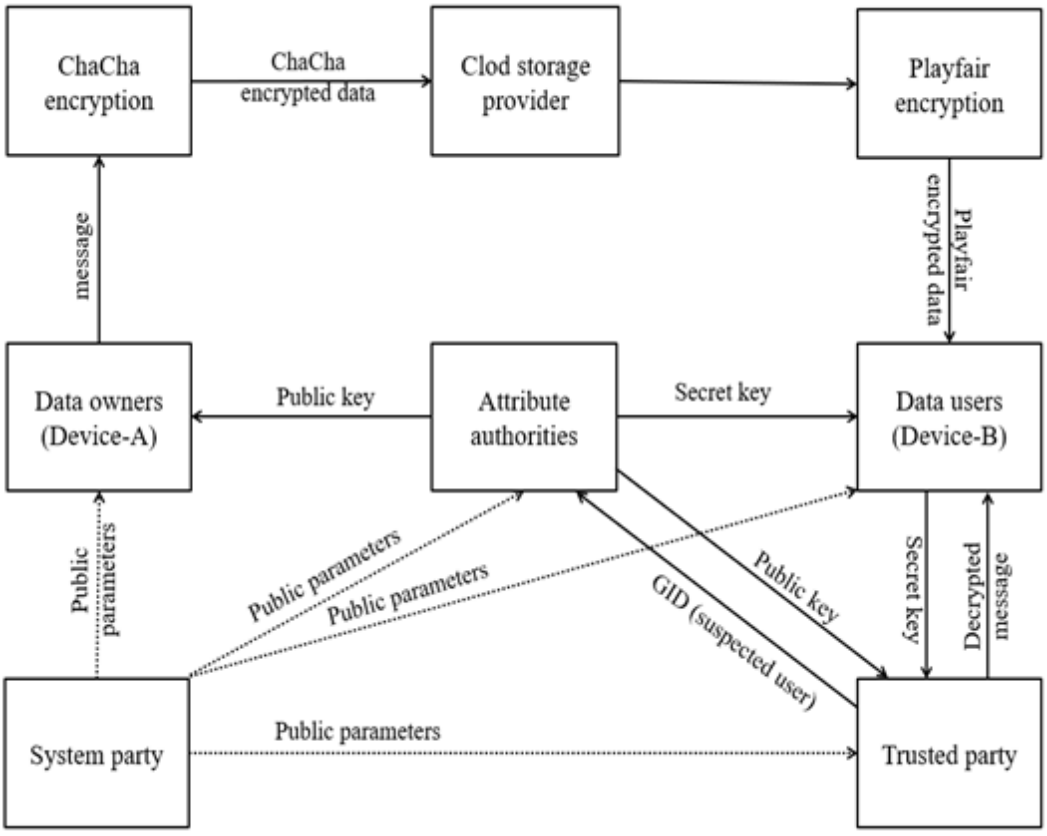


Figure 4.1: Model for the LWC-ABE system

Based on the selected encryption method, the ciphertext in the cloud storage is modified accordingly. When data users want to access the data, they transmit their secret key to a trusted party with the correct secret key generation capability. The trusted party performs the decryption operation using the secret key, resulting in the final decoded message. In case of disputes or suspicions, the trusted party utilizes the tracing algorithm and sends the User's ID (GID) of the suspected user to the attribute authorities for further investigation. So, this approach utilizes ABE, multiple encryption algorithms, and the involvement of attribute authorities and trusted parties to ensure secure data access, encryption flexibility, and traceability in case of disputes or suspicions.

4.2.1 ABE

ABE is a cryptographic scheme that provides fine-grained access control over encrypted data. It allows data owners to define access policies based on attributes, and only users possessing the required attributes can decrypt and access the data. ABE combines the properties of public-key encryption and access control, enabling secure and flexible data sharing in various scenarios.

Key Generation: The ABE scheme involves key generation for both data owners and users. Data owners generate a master secret key and ABE key that defines the access policy for the encrypted data. Users obtain their own secret keys, which are based on their attributes and the access policy defined by the data owner.

Encryption: To encrypt data using ABE, the data owner defines an access policy specifying which attributes are required to access the data. The data is encrypted using the ABE key and the access policy. The ciphertext is generated such that only users possessing the attributes specified in the access policy can decrypt the data.

Decryption: Users who possess the required attributes can decrypt the ciphertext and access the data. The decryption process involves verifying the user's attributes against the access policy specified in the ciphertext. If the user's attributes satisfy the policy, they can use their secret key to decrypt the ciphertext and obtain the original data.

Access Control: ABE provides fine-grained access control, allowing data owners to define complex access policies based on attributes. Attributes can represent various

characteristics such as user roles, clearances, time-based attributes, or any other user-defined attributes. Users must possess the required attributes to decrypt and access the data, ensuring that only authorized users can access specific data.

Scalability: ABE can support scalable access control in scenarios involving large-scale data sharing. Data owners can define access policies based on attributes, and users can be dynamically assigned attributes without requiring re-encryption of the data. This flexibility enables efficient and dynamic access control management.

Security: ABE schemes are designed to provide security guarantees such as confidentiality, data integrity, and access control. The security of ABE relies on the hardness of certain mathematical problems, such as the bilinear pairing or the Decisional Bilinear Diffie-Hellman assumption. Proper implementation and parameter selection are crucial to ensure the desired security properties.

Challenges: ABE also presents certain challenges. It requires a trusted authority or a central entity responsible for managing attribute assignment and key generation. The complexity of access policies and attribute management can introduce overhead in terms of key size, encryption, and decryption time. Additionally, revocation of user access or attribute changes can be challenging in certain ABE schemes.

Use Cases: ABE has applications in various domains, including secure data sharing in cloud computing, secure data sharing in IoT environments, secure healthcare data access, secure content distribution, and secure group communication, among others. ABE's flexible access control mechanisms make it suitable for scenarios where fine-grained control over data access is required

4.2.2 ChaCha Encryption

The ChaCha algorithm is widely used in various application fields, including mobile networks, wireless communications, etc. One of the key advantages of LWC-ABE is its utilization of the ChaCha encryption algorithm, which enables the creation of a new keystream generator for key generation. This integration of ChaCha raises the bar for security and reduces the overall complexity of the encryption process. In the context of IoT, where connected devices store sensitive data, there are specific resource requirements

and limitations on CPU power, energy consumption, and available bandwidth. LWC-ABE addresses these challenges by leveraging the efficiency of the ChaCha algorithm.

ChaCha is often employed in counter mode for symmetric encryption, as it effectively meets the criteria for such applications. Stream ciphers, including ChaCha, are designed with multiple rounds, with each round enhancing the security confidence but potentially reducing the processing capacity for a given time interval. During ChaCha's encryption process, the original data is combined with the keystream through an XOR operation. This XOR operation serves as the fundamental operation for producing the encrypted data. To achieve this, the ChaCha algorithm employs three lightweight operations: addition, XOR, and rotation of 32-bit data. The rotation operation involves shifting the bits by an endless number of positions, introducing additional complexity to the encryption process.

One crucial component of the ChaCha algorithm is the Quarter Round Function (QRF), which integrates the lightweight operations of addition, XOR, and rotation. The QRF is responsible for combining these operations in a specific way, ensuring the effectiveness and security of the encryption process. By utilizing the QRF, the ChaCha algorithm achieves the generation of ciphertext by combining the original data with the keystream through XOR operations and leveraging lightweight operations such as addition, XOR, and rotation. This approach significantly contributes to the secure encryption of data.

Furthermore, the QRF is utilized in each cycle of the ChaCha algorithm to update the state matrix, further enhancing the encryption process's security. The entire ChaCha-based LWC process involves ten cycles of operation, efficiently applying the algorithm to encrypt data securely and reliably.

Finally, the integration of the ChaCha algorithm into LWC-ABE brings numerous benefits, including enhanced security, reduced complexity, and efficient encryption in IoT environments. The lightweight operations of addition, XOR, and rotation, along with the QRF, contribute to the generation of ciphertext and ensure the confidentiality and integrity of data. Through its effective utilization, ChaCha-based LWC provides a robust encryption solution suitable for IoT applications.

The suggested procedure for generating ChaCha keystreams is outlined in Table 4.1

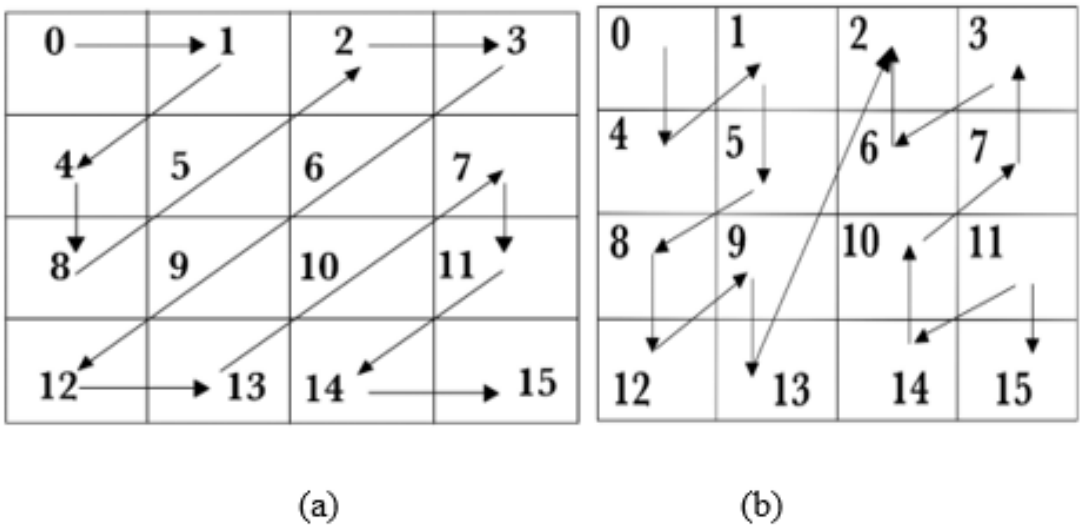


Figure 4.2: Input forms, (a) Zigzag form, (b) Alternative form

and presented for each round. The input matrix (I) has 512 bits and 16 seeds, each with 32 bits. In all, the matrix has 512 bits.

The input matrix includes the various keys ($k1...k8$), each of which has a size of 256 bits, the block message counter ($b1, b2$), each of which has a size of 64 bits, and the constants [$c1...c4$], each of which has a size of 192 bits, in addition to the nonce ($n1, n2$), which has the same size. In addition, the QRF technique used to produce the keystream is included in Table 4.2. At this point, it will produce the keystream by performing addition, XOR, and rotation operations.

Generate the rotation constants using the first four bits of input I_a, I_b, I_c, and I_d. Conventional techniques employ the rotation numbers 16, 12, 8, and 7 instead of the first four input bits. The input seeds should then be applied to the QRF in a zigzag pattern, as illustrated in Figure 4.2 (a), rather than in a column-wise manner as was previously done. In the proposed approach, a new sequence of updates for applying the input seeds is used, as depicted in Figure 4.2 (b). Instead of applying the input seeds in a row-by-row fashion, they are applied in an alternative form. This alternative sequence of updates results in a higher level of input dispersion, which enhances the defense against critical attacks.

Input: Consider 512 bits of input I with 16 seeds $[I = [I_o, I_1, \ldots, I_{15}]$.
Output: keystream with 512 bits.
Step 1: Initialize the round for keystream generation.
Step 2: Apply the 4 bytes of input to the QRF algorithm as presented in Table 4.2.
Step 3:Apply 32-bit input seeds in Zigzag form, as shown in Figure 4.2 (a). $[K_0, K_1, K_4, K_8] = QRF(I_0, I_1, I_4, I_8)$ $[K_5, K_2, K_3, K_6] = QRF(I_5, I_2, I_3, I_6)$ $[K_9, K_{12}, K_{13}, K_{10}] = QRF(I_9, I_{12}, I_{13}, I_{10})$ $[K_7, K_{11}, K_{14}, K_{15}] = QRF(I_7, I_{11}, I_{14}, I_{15})$
Step 4:Apply 32-bit input seeds in Alternate form as shown in Figure 4.2 (b) $[K_0, K_4, K_1, K_5] = QRF(I_0, I_4, I_1, I_5)$ $[K_8, K_{12}, K_9, K_{13}] = QRF(I_8, I_{12}, I_9, I_{13})$ $[K_2, K_6, K_3, K_7] = QRF(I_2, I_6, I_3, I_7)$ $[K_{10}, K_{14}, K_{11}, K_{15}] = QRF(I_{10}, I_{14}, I_{11}, I_{15})$
Step 5: Increment the round.
Step 6: Repeat steps 2 to 5 until the ten rounds are completed.
Step 7: The data presented in the $[K_0, K_1, \ldots, K_{15}]$ vectors are the final keystream.

Table 4.1: Initialization algorithm

Input: Input seeds I_a, I_b, I_c, I_d
Output: Keystream seed K_a, K_b, K_c, K_d
Step 1: The rotation constants $(I_{aR}, I_{bR}, I_{cR}, I_{dR})$ are developed as follows: $I_{aR} = I_a[3:0], \quad I_{bR} = I_b[3:0], \quad I_{cR} = I_c[3:0], \quad I_{dR} = I_d[3:0]$
Step 2: Generate the keystream seeds using a dual function: $K_a = I_a + I_b; \quad K_d = (I_d \oplus I_a) \lll I_{aR}$ $K_c = I_c + I_d; \quad K_b = (I_b \oplus I_c) \lll I_{bR}$ $K_a = I_a + I_b; \quad K_d = (I_d \oplus I_a) \lll I_{cR}$ $K_c = I_c + I_d; \quad K_b = (I_b \oplus I_c) \lll I_{dR}$

Table 4.2: QRF Algorithm

4.2.3 Privileged Encryption

The Playfair cipher is a multi-alphabet letter encryption cipher that turns plaintext letters into ciphertext letters by treating them as distinct units and using the ciphertext letters as input. To carry out the encryption procedure, the Playfair cipher uses the Polybius Square, which also functions as a key. The Polybius square has a matrix size of 5 by five and has 25 different elements. The alphabet should not be duplicated throughout the square. In addition, the Polybius square algorithm does not include the letter J, resulting in bits of plaintext overlapping. Because of this, the letter "J" will be changed to "I" if it appears in the keystream. In addition, there should be no repetition of any characters inside Polybius square. The next steps of the encryption procedure are then carried out:

Step 1: The plain text is broken into many diagraphs, which are created by combining two characters from the plaintext. In addition, the letter "Z" is added to the digraph if there is just one letter of the alphabet that is odd. Consider the word "$INSTRUMENTS$" as an example of plaintext comprising 11 letters. Therefore, the fictitious letter "Z" is appended to the LSB of the plaintext, which ultimately produces the output "$INSTRUMENTS$".

Step 2: If a digraph contains repeated letters or the same letters appearing twice side by side, the unknown letter "X" is assigned to the digraph's least significant bit (LSB) position. This is done to ensure the digraph is valid and to avoid confusion during encryption or decryption processes. Replacing the repeated letters with "X" in such cases maintains the integrity and correctness of the cryptographic operation. If we take the word "$COMMUNICATE$" as an example and assume it as plaintext, we may break it down into component digraphs: CO, OM, MM, MU, UN, NI, IC, CA, AT, TE. Since the MM digraph has the same letters as the MX digraph, it has been transformed.

Step 3: As an example, think of the word "$MONARCH$" as a Polybius square, as seen in Figure 4.3 (a), as this serves as a key text. The blank spaces in the square are filled with unique alphabets that are then arranged alphabetically.

In the first row of this box, jot down the letters of the term that was provided, reading from left to right. If the keyword contains any letters that are repeated, we

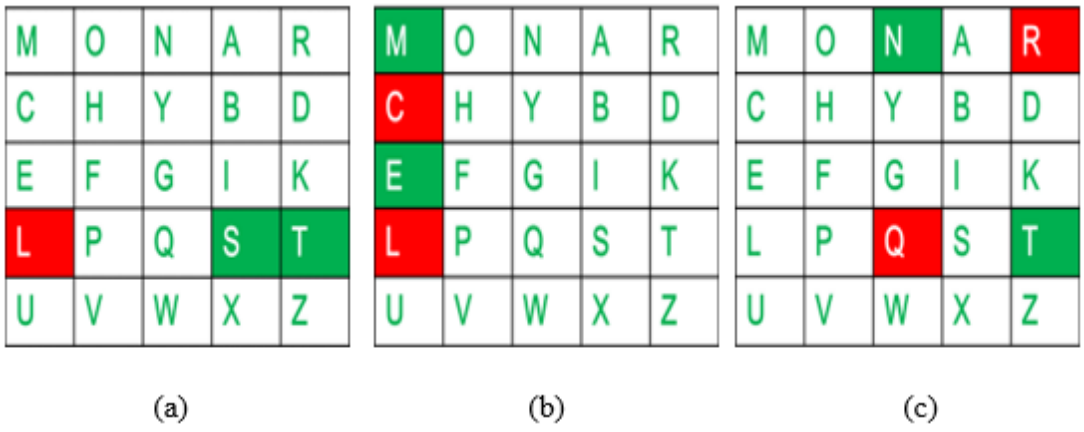


Figure 4.3: Polybius Square, (a) row-based ciphertext generation, (b) column-based ciphertext generation, (c) horizontal opposite corner-based ciphertext generation

should avoid using those letters. This suggests that each letter should be read just once before being discarded. After that, complete the sentence by writing the remaining letters alphabetically.

Step 4: If the digraph from the plaintext appears in the same row of the Polybius square, the ciphertext is created by considering the letters immediately to the right of the digraph. In cases where there are no letters on the right side, the first letter of the same row is used instead. Let us consider the example of the plaintext "$INSTRUMENTS$" and the digraph "ST". If we look at the Polybius square in Figure 4.3(a), we can see that the digraph "ST" is found in the fourth row. Following the rule above, the letter "S" in the plaintext is transformed into "T" in the ciphertext. Similarly, the letter "T" in the plaintext is transformed into "L" in the ciphertext. Therefore, the resulting ciphertext for the digraph "ST" in the given plaintext would be "TL" based on the Polybius square encryption scheme.

Step 5: If the digraph from the plaintext occurs in the same column of the Polybius square, then the ciphertext is constructed by considering the letters directly below the digraph. If the letters are not presented in the intended order, the letter that comes first in the same column must be utilized. Consider "$INSTRUMENTS$" to be the plaintext; the "ME" digraph was in the first column of the square, as shown in Figure 4.3(b). Consider "$INSTRUMENTS$" to be the plaintext. Therefore, the ciphertext representation of the letter "M" in plaintext is the letter "C", and the ciphertext representation of the letter

"E" in plaintext is the letter "L".

Consider the $M * N$ sub-matrix as the next stage in the process if none of the situations described in steps five or step 6 have happened and the digraph letters are arranged in separate columns and rows. In addition to this, the digraph letters need to be included inside the $M * N$ matrix. In the end, the ciphertext for that ciphertext is formed by always considering the letters in the horizontal corners that are opposite one another in the plain text. Because the "NT" digraph is missing from both the single row and the column of squares in Figure 4.3(c), "$INSTRUMENTS$" should be plaintext. Therefore, the components "N" and "T" are used to construct the submatrix that is four by 3.

Finally, the ciphertext is constructed for the letter "N" in the plaintext by considering the horizontal opposite corner letter to be an "R", and ciphertext is made for the letter "T" in the plaintext by taking into consideration the horizontal opposite corner letter to be a "Q".

Step 7: It is time to repeat the procedure with the remaining digraphs to produce the ciphertext.

Step 8: The steps involved in decryption are identical to those involved in encryption. However, they are completed in the opposite order. The decryption of the cipher was symmetric (go left along rows and up along columns). The receiver of plain text has the same capabilities.

4.3 Results and Discussion

This section aims to offer a comprehensive analysis, present simulation results, and compare the performance of the proposed LWC-ABE method with state-of-the-art approaches. Various performance metrics are utilized to evaluate the effectiveness of the proposed method and its comparison with existing methods. The time taken to detect attacks, known as the ADT, is an essential metric used to assess the efficiency of the proposed method in identifying and classifying attacks. By measuring the duration, it takes for the system to detect and respond to different types of attacks, the responsiveness and

effectiveness of the method can be evaluated.

4.3.1 Influence on the Amount of time required for Encryption and Decryption

The time required for encryption and decryption indicates the duration needed to perform the respective encryption and decryption operations. This study measured the encryption and decryption times for ten users with varying message lengths. Table 4.3 presents the results of the encryption and decryption times for the proposed LWC-ABE method compared to other ABE techniques, including CP-WABE [96], DM-ABE [95], DM-ABE [94], DM-ABE [97], and CP-ABE [98].

The findings demonstrate that the proposed LWC-ABE method requires less time for encryption and decryption than these other ABE techniques. This improved efficiency can be attributed to the use of a combination of the Playfair and ChaCha encryption algorithms in the proposed method, which enables the generation of keys in a timely manner. By leveraging the strengths of these algorithms, the proposed method achieves faster encryption and decryption operations, enhancing the system's overall performance.

Method	**Encryption time (seconds)**	**Decryption time (seconds)**
Proposed LWC-ABE	0.000835	0.000310
CP-WABE [96]	0.09651	0.09585
DM-ABE [95]	0.04104	0.03624
DM-ABE [94]	0.06252	0.01845
DM-ABE [97]	0.02686	0.007186
CP-ABE [98]	0.002000	0.0025

Table 4.3: Performance comparison of encryption and decryption times

Figure 4.4 andTable 4.4 illustrates the calculation of the encryption time for ten random users in an IoT environment. This graphical representation provides a visual understanding of the encryption time variation across different user scenarios, highlighting the efficiency of the proposed LWC-ABE method. These findings demonstrate that the proposed LWC-ABE method outperforms other ABE techniques regarding encryption and decryption times, offering improved efficiency and responsiveness in secure data

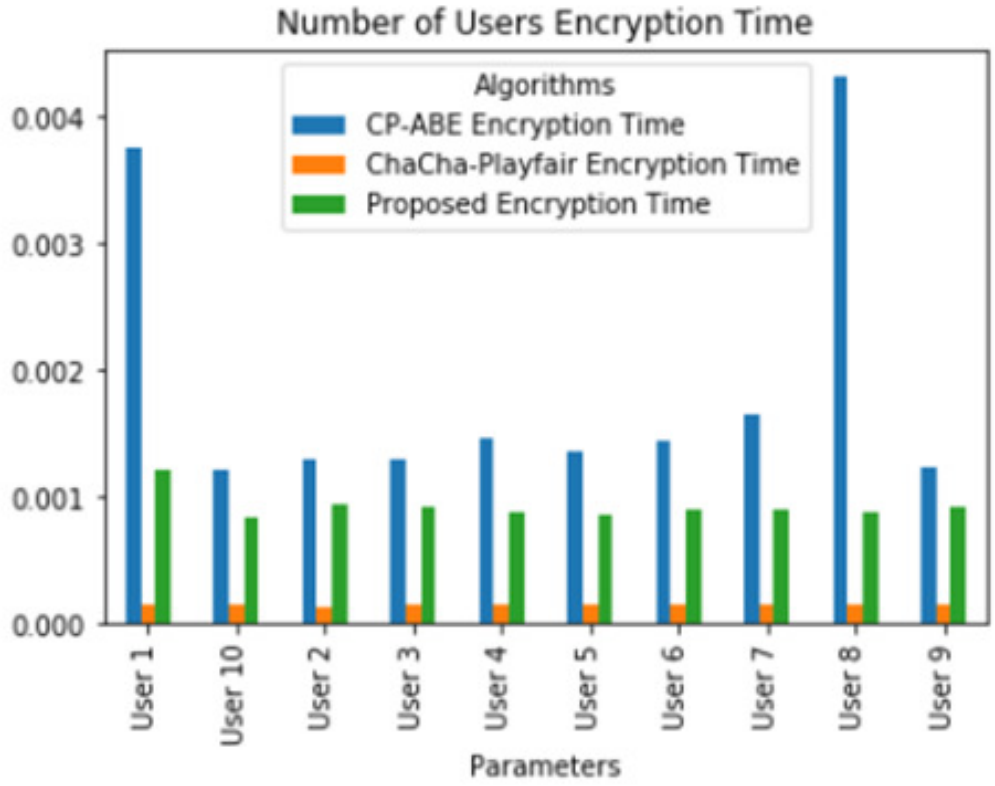


Figure 4.4: Analysis of encryption time for ten users

communication within IoT environments.

Figure 4.5 and Table 4.5 displays a similar prediction of the Amount of time required to decrypt data for ten users randomly selected from the same IoT network. In the situation with many users and authorities, the suggested LWC-ABE approach led to a reduced Amount of time needed for encryption and decryption. The traditional DM-ABE [97] and CP-ABE [98] techniques are having problems with the access policy, which has led to a strange rise in the Amount of time needed to generate a key for each user and has caused an increase in the Amount of time needed for cryptographic operations.

Table 4.6 presents the comparison of encryption and decryption times for the proposed LWC-ABE method and competing algorithms, including CP-WABE [96], DM-ABE [95], DM-ABE [94], DM-ABE [97], and CP-ABE [98], across different message sizes. The findings demonstrate that the proposed LWC-ABE method achieves faster encryption and decryption times than these competing algorithms.

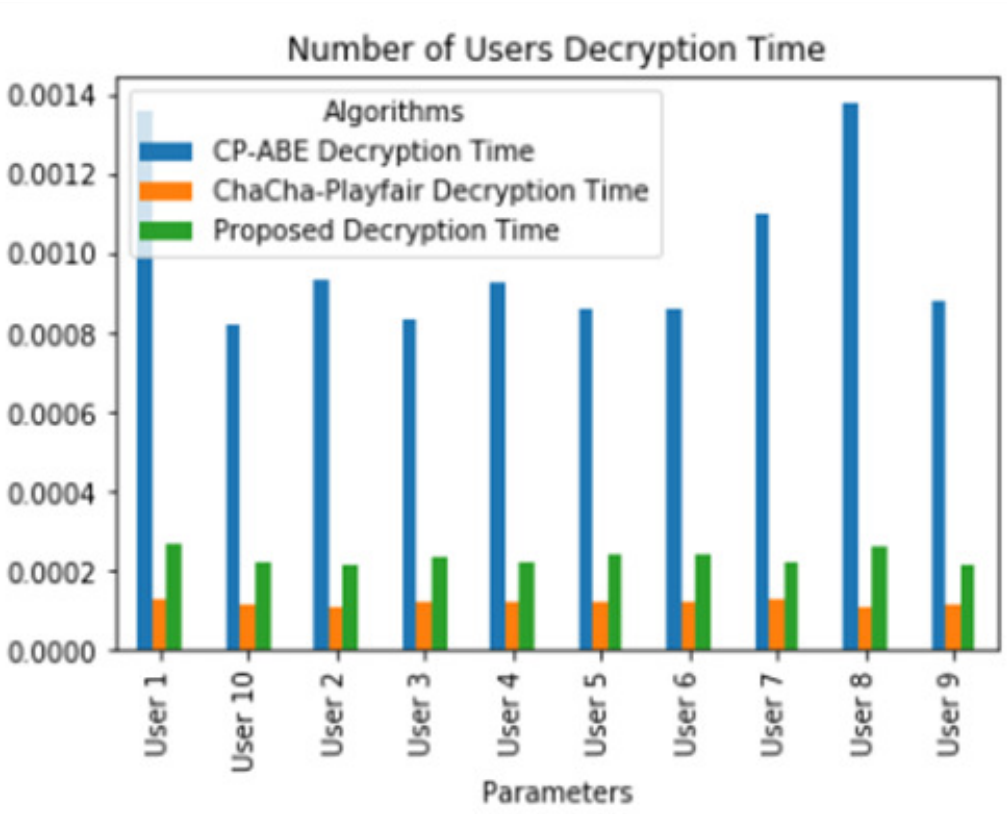


Figure 4.5: Analysis of decryption times for ten users

Users	**CP-ABE**	**ChaCha - playfair**	**Proposed**
User 1	0.0033	0.00015	0.0011
User 10	0.0010	0.00016	0.00076
User 2	0.0011	0.00014	0.00085
User 3	0.0011	0.00016	0.00082
User 4	0.0013	0.00016	0.00080
User 5	0.0012	0.00016	0.00078
User 6	0.0013	0.00016	0.00082
User 7	0.0014	0.00016	0.00018
User 8	0.0038	0.00016	0.00080
User 9	0.0011	0.00016	0.00083

Table 4.4: Performance of encryption time analysis for ten users.

Users	CP-ABE	ChaCha – playfair	Proposed
User 1	0.0013	0.00012	0.00026
User 10	0.00079	0.00011	0.00021
User 2	0.00090	0.00010	0.00020
User 3	0.00081	0.00011	0.00022
User 4	0.00090	0.00011	0.00021
User 5	0.00083	0.00011	0.00023
User 6	0.00083	0.00011	0.00023
User 7	0.00106	0.00012	0.00021
User 8	0.00133	0.00010	0.00025
User 9	0.00085	0.00011	0.00020

Table 4.5: Performance of decryption time analysis for ten users.

Method	ETVMS=100	DTVMS=100	ETVMS=200	DTVMS= 200
Proposed LWC-ABE	0.000134	0.000123	0.000102	0.000142
CP-WABE [96]	0.0766	0.09259	0.06851	0.09399
DM-ABE [95]	0.0667	0.07253	0.05963	0.07916
DM-ABE [94]	0.0487	0.06855	0.04818	0.06019
DM-ABE [97]	0.0150	0.05701	0.05049	0.04051
CP-ABE [98]	0.000867	0.0020	0.00125	0.000892

Table 4.6: Comparison of encryption and decryption timings dependent on message size

4.4 Summary

This chapter uses a versatile and helpful LWC-ABE approach to remove the uncommon attacks produced in an IoT context. Furthermore, using LWC-ABE in the IoT network reduces hardware resource utilization, including power consumption, while also meeting higher security requirements. This is achieved by leveraging the ChaCha and Playfair encryption algorithms, specifically chosen for their efficiency and security characteristics. Therefore, to maintain compliance with the higher security requirements, the data owners continually alter the policy access specifications and produce a variety of ciphertexts. In addition, according to the revised policy, the data owners may also flexibly

adjust the data access attributes to better suit their needs. In addition, an environment with different trusted authorities was implemented in the IoT network. This created a bottleneck in the IoT servers, but it gave IoT devices the freedom to vary their admission procedure. The simulation findings indicate that the implementation of the LWC-ABE presented resulted in shorter durations spent encrypting and decrypting data for multi-user, variable-message-size situations compared to standard methods. This work was expanded to include the use of hybrid encryption methods to improve the security of IoT.

Chapter 5

Secure And Scalable IoMT Using Ensemble LWC Model

5.1 Introduction

The IoMT defines the notion of linked devices and objects of any kind that is wired, wireless, or connected to the internet. Because these technologies are utilized for various goals, including transportation, communication, the growth of businesses, and education, the popularity of the notion has expanded over the years. The IoMT developed hyper-connectivity, which is a demonstration that demonstrates how people and organizations may easily interact with each other from their respective faraway places. There are already around 26.66 billion IoT devices in use throughout the globe. This widespread investigation of the IoMT's usefulness started in 2011 with the automation of homes, the installation of smart energy meters, and the use of wearable devices. The exploration of the IoT has been helpful to businesses in a variety of ways, including the improvement of corporate strategy, the use of blockchain for the recording of transactions and the monitoring of assets, and market research. The advent of automated services has also improved people's quality of life because of the IoT. Nevertheless, the unchecked development of these technologies has created security and privacy difficulties.

Using an innovative cryptographic model with optimization strategies, the authors of [99] investigated ways to ensure the safety of medical images sent over the internet. In ECC, the ideal Key will be selected with hybrid swarm optimization-based key generation

Hybrid Swarm Optimization-Based Key Generation (HSO-KG), also known as grasshopper optimization and particle swarm optimization. This method is used to generate keys. Both terms refer to the same process. Because of this, the encrypting and decrypting operations will be carried out with the greatest possible degree of security.

The authors of the research offered a comparative analysis and performance assessment of three trustworthy candidate encryption algorithms [100]. AES, SPECK, and SIMON were the names of these algorithms. Simulations of the algorithms were run and examined side-by-side in detail to establish whether one demonstrated the most desirable characteristics and ought to be considered for use in a medical application.

To encrypt medical photographs in a way that is both efficient and secure, the authors of the research [101] proposed using a lightweight cryptosystem that they referred to as HBC-LWC. This system was built using the Henon chaotic map, Brownian motion, and Chen's chaotic system as its foundations. The Brownian motion also played a role. The effectiveness of this method is evaluated by considering several different attacks. The authors of [102] described a lightweight selective encryption Light Weight Selective Encryption (LWSE) approach to encrypting the edge maps of medical pictures. A technique for detecting edges is used to begin extracting the edge map. After that, a chaotic map produces a sizable key space. A one-time pad encryption scheme was presented in this study to encrypt the significant identified picture blocks correspondingly.

However, the conventional encryption models failed to provide maximum security. Therefore, this work focused on implementing the LW-MIC system using ELWC protocols. Initially, the medical image data from users is converted into digital data. Then, ELWC operation is applied to vector data, which implements play-fair and Cha-Cha-based encryption algorithms. So, secured image data of users are transmitted over the IoMT environment. Finally, the ELWC decryption algorithms restore the original image data at the receiver (doctor) side.

5.2 Proposed Method

This section gives a detailed analysis of the proposed LW-MIC performance. The survey shows that the combination of play-fair and Cha-Cha encryption algorithms is not used. Figure 5.1 shows the block diagram of the proposed LW-MIC. The LW-MIC system contains the application environment (indicated by the orange color outer line) and proposed framework (indicated by a dotted line). The application environment is mainly used to establish networking connections between patients and doctors. The data (medical images, medical statistics) generated from patients are transferred to the IoMT cloud using a networking environment, achieved by different wireless communication services. Similarly, the doctor receives the patient's data using a server environment. This application environment is a bi-directional communication medium, so the patient and doctor can communicate in a secure environment.

Here, this security is achieved by the proposed LW-MIC protocols. Initially, a patient's medical data is converted into a uniform vector with digital data. Then, the ELWC approach implements both the play-fair and Cha-Cha algorithms on vector data one after another. In the last step, the encrypted picture data is moved to the environment of the IoMT. After that, the ELWC decryption procedure is carried out at the receiver end, ultimately restoring the initial picture. The recovered and encrypted picture is sent to the medical practitioner via the application environment in the last step. Usually, the doctor will perform the diagnosis operation and send the resulting outcome to the patient in a secure environment.

5.2.1 ChaCha Encryption and Decryption

The ELWC is the hybrid process of encryption, which performs the combination of ChaCha and Playfair encryption algorithms jointly. So, this hybrid combination provides more security than individual algorithms. This section gives a detailed analysis of the ChaCha encryption algorithm procedure. Table 5.1 shows the proposed ChaCha image encryption algorithm. The ChaCha decryption algorithm is exactly the opposite of the encryption process.

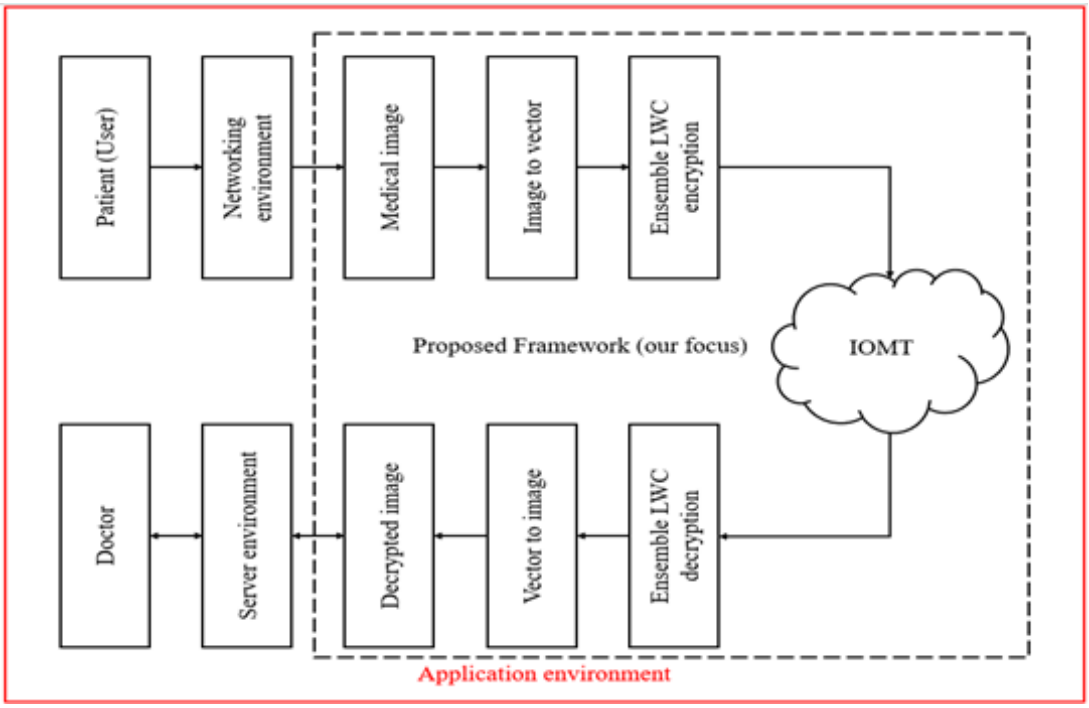


Figure 5.1: Proposed LW-MIC block diagram

5.2.2 Playfair encryption

This section gives the Playfair algorithm's detailed procedure for image encryption and decryption. The ChaCha encryption algorithm is applied as input to the Playfair algorithm.

Playfair image encryption

Step 1: Read the source image and convert it into red, green, and blue planes based on three separate matrices.

Step 2: If each plane contains an odd number of columns or rows, then add extra zeros to columns or rows. So, the image is generated with even dimensions.

Step 3: Create the 16 x 16 key matrix using any integers in the range from 0 to 255.

Step 4: For each separate matrix, perform steps 5 to 9.

Step 5: If the values are in different rows and columns, then the pair should be replaced with the values at the opposite corners of the rectangle produced by the original pair. The pair should remain unchanged if the values are in the same row and column. The order of the values must be maintained.

Input: Source medical image (I)
Output: Cipher image
Step 1: Read the source medical image and convert it into red, green, and blue planes based on three separate matrices.
Step 2: Decompose each color plane matrix into 16 bytes, i.e., $I_0, I_1, \ldots, I_{16}$.
Step 3: Initialize the round, and each round generates keystream using Quarter Round Function (QRF) process. It generates the rotation round keys I_{aR}, I_{bR}, I_{cR} and I_{dR} for data shifting. $I_{aR} = I_a[3:0], \quad I_{bR} = I_b[3:0], \quad I_{cR} = I_c[3:0], \quad I_{dR} = I_d[3:0]$ Then, produce the keystream utilizing dual function by applying rotation round keys on input bytes (I_a, I_b, I_c, I_d). $K_a = I_a + I_b; \quad K_d = (I_d \oplus I_a) \lll I_{aR}$ $K_c = I_c + I_d; \quad K_b = (I_b \oplus I_c) \lll I_{bR}$ $K_a = I_a + I_b; \quad K_d = (I_d \oplus I_a) \lll I_{cR}$ $K_c = I_c + I_d; \quad K_b = (I_b \oplus I_c) \lll I_{dR}$ Here, K_a, K_b, K_c, and K_d are the resultant keystreams.
Step 4: Perform the Zigzag form and Alternate form-based keystream generation using QRF rules.
Step 5: Concatenate the outcomes generated using Zigzag and Alternate form outcomes.
Step 6: Repeat the process performed in steps 2 to 6 until the ten rounds are completed.
Step 7: The resultant data generated from round 10 is considered the final keystream.
Step 8: Convert the matrix to an image, which generates the cipher image.

Table 5.1: Proposed ChaCha image encryption algorithm

Step 6: If the values are found on the same row of the matrix, replace them with the values that are found to their immediate right (which will result in the matrix being read from the right to the left).

Step 7: Replace the values with those directly below them if they exist in the same matrix column.

Step 8: Use the secret Key to produce a mask composed of a random permutation of the integers ranging from 0 to 255.

Step 9: Combine, using XOR, the obtained scrambled picture with the created random mask.

Step 10: The combined image from the red, green, and blue planes is considered a cipher image.

Playfair image decryption

Step 1: Read the Cipher picture as a series of red, green, and blue matrices.

Step 2: Use the hidden Key to construct a mask composed of a random permutation of the integers ranging from 0 to 255.

Step 3: Using the produced random mask, do an XOR operation on the red color plane of the cipher picture.

Step 4: Using the secret Key, create a Key Square, a 16 by 16 matrix consisting of random integer integers between 0 and 255.

Step 5: Carry out the procedures listed below for each pair of the XORed red plane that resulted in the cipher image: (a) If the values are in different rows and columns, replace the pair with the values that are in the opposite corners of the rectangle produced by the original pair. This step is only necessary if the values are in different rows and columns. The order of the values must be maintained. (b) If the values are located on the same row of the matrix, replace them with the values located to their immediate right properly. This will cause the matrix to wrap around to the left side of the page. (c) Replace the values in the matrix with those directly below them in the correct order if they are placed in the same column (wrapping around to the top side of the column).

Step 6: Repeat steps 3 to 5 to decipher the cipher picture using the green and blue color planes.

Step 7: Make the picture that was generated into the Plain image.

5.3 Results and Discussion

This section presents LW-MIC analysis, replication, and performance comparisons with current methods. Performance metrics: improved PSNR, SSIM, MSE, encryption, decryption, and computation time.

5.3.1 Subjective performance

Figure 5.2 presents the subjective performance of the proposed LW-MIC method. Here, Figure 5.2(a) represents the original medical images of the patient, where two sets of brain MRI and chest-x-ray images are considered. The encrypted data generated using the proposed LW-MIC approach contains cipher text. Figure 5.2(b) shows the decrypted images generated using the existing HSO-KG [99] approach. However, these images contain brightness and contrast illumination problems and higher noise artifacts, proving that the existing HSO-KG cannot recover the original image from encrypted data. Finally, Figure 5.2(c) shows recovered images using the proposed LW-MIC approach, which looks like the original image. Therefore, the suggested LW-MIC strategy produced improved visual performance compared to the usual HSO-KG approach.

5.3.2 Objective performance

Table 5.2 compares the image quality performance, where the proposed LW-MIC method resulted in improved quality metrics as compared to conventional approaches such as HSO-KG, AES-SPECK [100], HBC-LWC [101], and LWSE [102]. Here, the proposed LW-MIC resulted in increased decrypted image quality (i.e., PSNR), decrypted image similarity with the original image (i.e., SSIM), and reduced pixel changes in the decrypted image in comparison with the original image (i.e., MSE).

Table 5.3 compares the time complexity performance of various methods, such as HSO-KG , AES-SPECK , HBC-LWC , and LWSE . Here, the encryption time, decryption time, and overall computation time of the proposed LW-MIC were reduced due to the low-complex design style. As the time complexity reduces, the patient data is encrypted quickly, and the doctor receives the decrypted data quickly, which reduces the processing

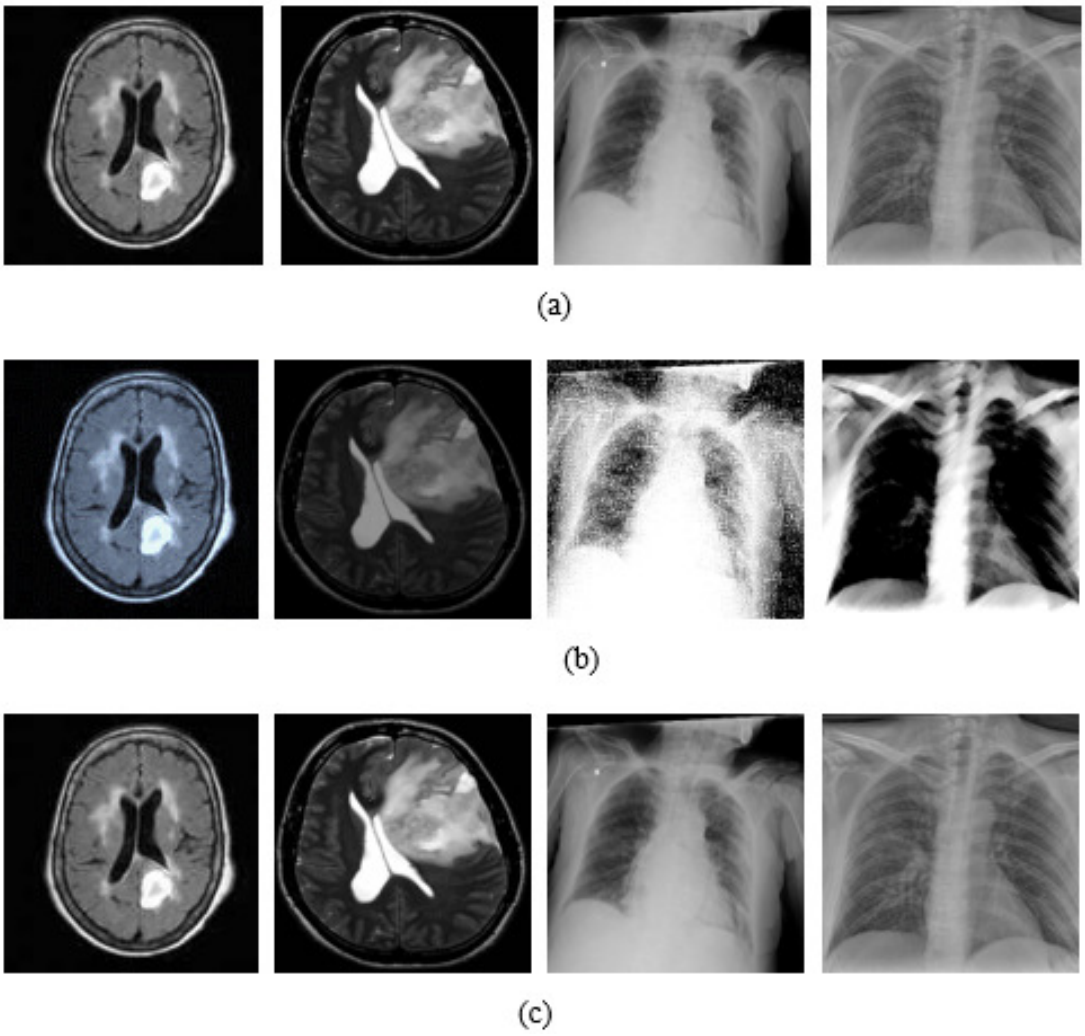


Figure 5.2: Subjective performance of the proposed method (a) Source images (b) Decrypted images using existing HSO-KG (c)Decrypted images using proposed LW-MIC.

time in the IoMT environment.

5.4 Summary

The primary emphasis of this study was developing an LW-MIC system using an ensemble of LWC protocols. The user's analogue medical imaging data is initially transformed into its digital equivalent. After that, the play-fair and Cha-Cha-based encryption algorithms were implemented using the ELWC operation applied to the vector data. Therefore, users' picture data are encrypted before being sent across IoMT's infrastructure. Finally, the ELWC decryption methods are implemented to return the original

Method	PSNR (dB)	SSIM	MSE
HSO-KG [99]	32.397	0.7240	0.745
AES-SPECK [100]	36.485	0.7834	0.425
HBC-LWC [101]	41.226	0.845	0.274
LWSE [102]	43.858	0.9134	0.1371
Proposed LW-MIC	100	1.0	0

Table 5.2: Image quality performance comparison of various methods

Method	Encryption time	Decryption time	Overall computation time
HSO-KG [99]	0.08246	0.0703	0.09756
AES-SPECK [100]	0.0713	0.0526	0.08645
HBC-LWC [101]	0.0613	0.0354	0.07264
LWSE [102]	0.0143	0.03481	0.06246
Proposed LW-MIC	0.003813	0.004404	0.008216

Table 5.3: Time complexity (in seconds) performance of various methods

picture data on the receiver's (the doctor's) side. Compared to the state-of-the-art approaches, the simulation results showed that the suggested LW-MIC system produced better picture encryption while simultaneously reducing the amount of time and complexity involved. Further, this work can be extended to implement a real-time secured medical data transmission system with prediction models.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

This research study initially introduced a ULWC that coupled a DLCNN approach with an IoT environment and used a symmetric key-based communication protocol. In this work, the logistic map process was used to implement the primary delegation strategy. Within the framework of this method, the creation of random numbers is used to come up with an initial value for a parameter. As a direct result of this fact, the control center oversees configuring each IoT device to its default state. The proposed method utilizes a randomly generated key for authentication in scenarios where an IoT device is primarily connected to a control center, such as in a cloud computing environment.

This control center provides another key pair for D2D connections based on a pre-trained DLCNN embedded in each IoT device. This enhances the security of communication between devices. When an IoT device sends a new communication request, the DLCNN model, which has been pre-trained, processes the request. It analyzes the request and predicts whether it is a normal request or contains any attack signatures. This prediction is based on the model's ability to identify patterns and anomalies in the communication data. With this approach, IoT devices can securely connect while being monitored by a system that leverages machine learning techniques. The simulation results indicate that the proposed method outperforms current techniques regarding various performance metrics, including ADT, encryption time, decryption time, and the attack detection confusion matrix. This suggests that the proposed approach offers im-

proved efficiency and accuracy in detecting and responding to potential attacks in IoT communication scenarios. These results were obtained by running the method through the simulation.

Second, this research built a varied and beneficial LWC-ABE strategy to eradicate the attacks generated in the IoT environment. In addition, LWC-ABE reduces hardware resources such as power consumption and implements improved security needs in the IoT network by using the ChaCha and Playfair encryptions. These benefits come from LWC-ABE's utilization of the ChaCha and Playfair algorithms. These benefits come from the use of the ChaCha and Playfair encryptions. The use of the appropriate algorithms brings about the desired results. The method that has been proposed makes it possible to simultaneously update policies, revoke attributes, and outsource decryption characteristics. Because of this, for the data owners to remain in compliance with the greater security standards, they must continuously change the policy access specifications and develop a range of ciphertexts. In addition, the newly amended policy allows the data owners to flexibly modify the data access qualities to meet their requirements better. In addition, a set consisting of many distinct authorities that were relied upon was included in the IoT network. This resulted in a bottleneck in the IoT servers and provided IoT devices with the flexibility to adjust their access policy as they saw fit. The simulation outcomes suggest that implementing the LWC-ABE resulted in shorter periods spent encrypting and decrypting data compared to traditional techniques in cases involving many users and varied message sizes. This was the case regardless of whether the data was encrypted or decrypted. To strengthen protection, the parameters of this investigation were broadened to include the use of hybrid encryption strategies.

Finally, developing an LW-MIC system using an ensemble of LWC protocols was the primary emphasis of this study. The user's medical imaging data is initially transformed into its digital equivalent. After that, the play-fair and Cha-Cha-based encryption algorithms were implemented using the ELWC operation applied to the vector data. Therefore, users' picture data are encrypted before being sent across IoMT's infrastructure. Finally, the ELWC decryption methods are implemented to return the original picture data on the receiver's (the doctor's) side. Compared to the state-of-the-art approaches, the simulation results showed that the suggested LW-MIC system produced better picture

encryption while simultaneously reducing the amount of time and complexity involved. Further, this work can be extended to implement a real-time secured medical data transmission system with prediction models.

6.2 Future Scope

The future directions of ULWC are expected to focus on several areas: post-quantum cryptography, homomorphic encryption, quantum-resistant cryptography, privacy-preserving cryptography, and multivariate cryptography.

A kind of encryption known as post-quantum cryptography is intended to be safe against attacks carried out by quantum computers. It is anticipated that quantum computers will be able to break a significant number of the current public-key cryptosystems, including RSA and ECC. Researchers are now developing post-quantum cryptography systems immune to quantum attacks to solve this challenge. In the future, ultra-lightweight post-quantum cryptographic schemes will be developed to secure IoT devices against quantum attacks.

Calculations were performed on data that has been encrypted using a method known as homomorphic encryption. This eliminates the requirement to decode the data to do the calculations on the encrypted data. This is useful for IoT devices that compute sensitive data, such as medical records or financial transactions. However, homomorphic encryption is computationally expensive and requires much memory. To address this problem, researchers are developing lightweight homomorphic encryption schemes that can be used on resource-constrained devices.

The term "quantum-resistant cryptography" refers to a form of cryptographic protocol that is supposed to be impenetrable by quantum computers and other similar devices. Quantum-resistant cryptography, on the other hand, is meant to be safe against attacks by quantum computers even in the present day, in contrast to post-quantum cryptography, which is planned to be secure against attacks by quantum computers in the future. In the not-too-distant future, ultra-lightweight quantum-resistant cryptographic methods suitable for usage in devices with limited resources will be created.

A kind of cryptography known as privacy-preserving cryptography enables calculations to be carried out on encrypted data without disclosing any of the associated information in the process. IoT devices that need to calculate sensitive data without disclosing the data itself benefit from this feature. In the future, ultra-lightweight privacy-preserving cryptographic schemes will be developed for IoT devices.

Multivariate cryptography is a type of cryptography based on solving systems of polynomial equations. It is resistant to attacks by quantum computers and is computationally efficient. Ultra-lightweight multivariate cryptographic schemes will be developed for use in resource-constrained devices.

So, the future of ULWC will focus on developing cryptographic algorithms and protocols designed to meet the unique needs of resource-constrained devices while also providing strong security guarantees. These include post-quantum cryptography, lightweight homomorphic encryption, quantum-resistant cryptography, privacy-preserving cryptography, and multivariate cryptography.

Publications

List of International Journals:

1. Mounika Jammula, Venkata Mani Vakamulla, Sai Krishna Kondoju,"Artificial intelligence framework-based ultra-lightweight communication protocol for prediction of attacks in Internet of Things environment", Vol.34, Pages: e4680, Year: 2023/1, Transactions on Emerging Telecommunications Technologies, John Wiley & Sons, Ltd.**(SCI)**

2. Mounika Jammula, Venkata Mani Vakamulla, Sai Krishna Kondoju,"Hybrid lightweight cryptography with attribute-based encryption standard for secure and scalable IoT system", vol 34, year 2022, Taylor & Francis.**(SCI)**

3. M Jammula, VM Vakamulla, SK Kondoju,"Performance Evaluation of Lightweight Cryptographic Algorithms for Heterogeneous IoT Environment", Journal of Interconnection Networks, pages-2141031, Jan 2022.

List of International Conferences:

1. Mounika Jammula; Venkata Mani Vakamulla; Sai Krishna Kondoju,"Secure and Scalable Internet of Medical Things using Ensemble Lightweight Cryptographic Model", Pg 982-987, IEEE 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS), 14-16 June 2023, Coimbatore, India.

Bibliography

[1] S. Sutar and P. Mekala, "An extensive review on iot security challenges and lwc implementation on tiny hardware for node level security evaluation," *International Journal of Next-Generation Computing*, vol. 13, no. 1, 2022.

[2] V. Bhagat, S. Kumar, S. K. Gupta, and M. K. Chaube, "Lightweight cryptographic algorithms based on different model architectures: A systematic review and futuristic applications," *Concurrency and Computation: Practice and Experience*, vol. 35, no. 1, p. e7425, 2023.

[3] R. S. Salman, A. K. Farhan, and A. Shakir, "Lightweight modifications in the advanced encryption standard (aes) for iot applications: a comparative survey," in *2022 International Conference on Computer Science and Software Engineering (CSASE)*. IEEE, 2022, pp. 325–330.

[4] V. Jayaprakash and A. K. Tyagi, "Security optimization of resource-constrained internet of healthcare things (ioht) devices using lightweight cryptography," in *Information Security Practices for the Internet of Things, 5G, and Next-Generation Wireless Networks*. IGI Global, 2022, pp. 179–209.

[5] Y. A. Birgani, S. Timarchi, and A. Khalid, "Ultra-lightweight fpga-based rc5 designs via data-dependent rotation block optimization," *Microprocessors and Microsystems*, vol. 93, p. 104588, 2022.

[6] N. Thangamani and M. Murugappan, "A lightweight cryptography technique with random pattern generation," *Wireless Personal Communications*, vol. 104, pp. 1409–1432, 2019.

[7] I. El Gaabouri, M. Senhadji, and M. Belkasmi, "A survey on lightweight cryptography approach for iot devices security," in *2022 5th International Conference on Networking, Information Systems and Security: Envisage Intelligent Systems in 5g//6G-based Interconnected Digital Worlds (NISS)*. IEEE, 2022, pp. 1–8.

[8] M. Rana, Q. Mamun, and R. Islam, "Lightweight cryptography in iot networks: A survey," *Future Generation Computer Systems*, vol. 129, pp. 77–89, 2022.

[9] J. B. Awotunde, S. Misra, and Q. T. Pham, "A secure framework for internet of medical things security based system using lightweight cryptography enabled blockchain," in *International Conference on Future Data and Security Engineering*. Springer, 2022, pp. 258–272.

[10] N. A. Gunathilake, W. J. Buchanan, and R. Asif, "Next generation lightweight cryptography for smart iot devices:: implementation, challenges and applications," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. IEEE, 2019, pp. 707–710.

[11] D. Bakkiam Deebak and F. AL-Turjman, "Lightweight privacy-aware secure authentication scheme for cyber-physical systems in the edge intelligence era," *Concurrency and Computation: Practice and Experience*, vol. 35, no. 13, p. e6510, 2023.

[12] M. A. F. Al-Husainy, B. Al-Shargabi, and S. Aljawarneh, "Lightweight cryptography system for iot devices using dna," *Computers and Electrical Engineering*, vol. 95, p. 107418, 2021.

[13] R. Chatterjee, R. Chakraborty, and J. Mondal, "Design of lightweight cryptographic model for end-to-end encryption in iot domain," *IRO Journal on Sustainable Wireless Systems*, vol. 1, no. 4, pp. 215–224, 2019.

[14] J. Wang, Y. Zhu *et al.*, "Secure two-factor lightweight authentication protocol using self-certified public key cryptography for multi-server 5g networks," *Journal of Network and Computer Applications*, vol. 161, p. 102660, 2020.

[15] B. Ying and A. Nayak, "Lightweight remote user authentication protocol for multi-server 5g networks using self-certified public key cryptography," *Journal of network and computer applications*, vol. 131, pp. 66–74, 2019.

[16] E. Lara, L. Aguilar, and J. A. García, "Lightweight authentication protocol using self-certified public keys for wireless body area networks in health-care applications," *IEEE Access*, vol. 9, pp. 79 196–79 213, 2021.

[17] B. M. Alshammari, R. Guesmi, T. Guesmi, H. Alsaif, and A. Alzamil, "Implementing a symmetric lightweight cryptosystem in highly constrained iot devices by using a chaotic s-box," *Symmetry*, vol. 13, no. 1, p. 129, 2021.

[18] R. Rohit, "Design and cryptanalysis of lightweight symmetric key primitives," 2020.

[19] A. Biryukov and L. Perrin, "State of the art in lightweight symmetric cryptography," *Cryptology ePrint Archive*, 2017.

[20] M. Jammula, V. M. Vakamulla, and S. K. Kondoju, "Performance evaluation of lightweight cryptographic algorithms for heterogeneous iot environment," *Journal of Interconnection Networks*, vol. 22, no. Supp01, p. 2141031, 2022.

[21] H. Noura, R. Couturier, C. Pham, and A. Chehab, "Lightweight stream cipher scheme for resource-constrained iot devices," in *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2019, pp. 1–8.

[22] O. Kuznetsov, O. Potii, A. Perepelitsyn, D. Ivanenko, and N. Poluyanenko, "Lightweight stream ciphers for green it engineering," *Green IT Engineering: Social, Business and Industrial Applications*, pp. 113–137, 2019.

[23] B. Aboushosha, R. A. Ramadan, A. D. Dwivedi, A. El-Sayed, and M. M. Dessouky, "Slim: A lightweight block cipher for internet of health things," *IEEE Access*, vol. 8, pp. 203 747–203 757, 2020.

[24] D. Sehrawat, N. S. Gill, and M. Devi, "Comparative analysis of lightweight block ciphers in iot-enabled smart environment," in *2019 6th International Conference on Signal Processing and Integrated Networks (SPIN)*. IEEE, 2019, pp. 915–920.

[25] A. Andrushkevych, Y. Gorbenko, O. Kuznetsov, R. Oliynykov, and M. Rodinko, "A prospective lightweight block cipher for green it engineering," *Green IT Engineering: Social, Business and Industrial Applications*, pp. 95–112, 2019.

[26] A. Biswas, A. Majumdar, S. Nath, A. Dutta, and K. Baishnab, "Lrbc: a lightweight block cipher design for resource constrained iot devices," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–15, 2020.

[27] D. Dinu, Y. L. Corre, D. Khovratovich, L. Perrin, J. Großschädl, and A. Biryukov, "Triathlon of lightweight block ciphers for the internet of things," *Journal of Cryptographic Engineering*, vol. 9, pp. 283–302, 2019.

[28] N. Nabeel, M. H. Habaebi, and M. R. Islam, "Security analysis of lnmnt-lightweight crypto hash function for iot," *IEEE Access*, vol. 9, pp. 165 754–165 765, 2021.

[29] ——, "Security analysis of lnmnt-lightweight crypto hash function for iot," *IEEE Access*, vol. 9, pp. 165 754–165 765, 2021.

[30] D. N. Gupta and R. Kumar, "Sponge based lightweight cryptographic hash functions for iot applications," in *2021 International Conference on Intelligent Technologies (CONIT)*. IEEE, 2021, pp. 1–5.

[31] R. Kuang, D. Lou, A. He, and A. Conlon, "Quantum safe lightweight cryptography with quantum permutation pad," in *2021 IEEE 6th international conference on computer and communication systems (ICCCS)*. IEEE, 2021, pp. 790–795.

[32] H. Faria and J. M. Valença, "Post-quantum authentication with lightweight cryptographic primitives," *Cryptology ePrint Archive*, 2021.

[33] K. B. Jang, M. J. Sim, and H. J. Seo, "Design of a lightweight security protocol using post quantum cryptography," *KIPS Transactions on Computer and Communication Systems*, vol. 9, no. 8, pp. 165–170, 2020.

[34] H. Cheng, J. Großschädl, P. B. Rønne, and P. Y. Ryan, "Avrntru: Lightweight ntru-based post-quantum cryptography for 8-bit avr microcontrollers," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 1272–1277.

[35] Y. Li, P. Zhang, and R. Huang, "Lightweight quantum encryption for secure transmission of power data in smart grid," *IEEE Access*, vol. 7, pp. 36 285–36 293, 2019.

[36] 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," *International Journal of intelligent networks*, vol. 3, pp. 16–30, 2022.

[37] S. Li, S. Zhao, G. Min, L. Qi, and G. Liu, "Lightweight privacy-preserving scheme using homomorphic encryption in industrial internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 14 542–14 550, 2021.

[38] K. Mandal and G. Gong, "Homomorphic evaluation of lightweight cipher boolean circuits," in *International Symposium on Foundations and Practice of Security*. Springer, 2021, pp. 63–74.

[39] K. Chait, A. Laouid, L. Laouamer, and M. Kara, "A multi-key based lightweight additive homomorphic encryption scheme," in *2021 International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP)*. IEEE, 2021, pp. 1–6.

[40] N. Mohankumar, M. Jayakumar, and M. Nirmala Devi, "Lightweight logic obfuscation in combinational circuits for improved security—an analysis," in *Expert Clouds and Applications: Proceedings of ICOECA 2021*. Springer, 2022, pp. 215–225.

[41] V. S. Rathor and G. Sharma, "A lightweight robust logic locking technique to thwart sensitization and cone-based attacks," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 2, pp. 811–822, 2019.

[42] N. Mohankumar, M. Jayakumar, and M. Nirmala Devi, "Lightweight logic obfuscation in combinational circuits for improved security—an analysis," in *Expert Clouds and Applications: Proceedings of ICOECA 2021*. Springer, 2022, pp. 215–225.

[43] K. Z. Azar, F. Farahmand, H. M. Kamali, S. Roshanisefat, H. Homayoun, W. Diehl, K. Gaj, and A. Sasan, "{COMA}: Communication and obfuscation management architecture," in *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*, 2019, pp. 181–195.

[44] V. S. Rathor and G. Sharma, "A lightweight robust logic locking technique to thwart sensitization and cone-based attacks," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 2, pp. 811–822, 2019.

[45] M. A. F. Al-Husainy, B. Al-Shargabi, and S. Aljawarneh, "Lightweight cryptography system for iot devices using dna," *Computers and Electrical Engineering*, vol. 95, p. 107418, 2021.

[46] J.-P. Kaps, W. Diehl, M. Tempelmeier, E. Homsirikamol, and K. Gaj, "Hardware api for lightweight cryptography," *URL https://cryptography. gmu. edu/athena/index. php*, pp. 1–26, 2019.

[47] M. S. Turan, K. McKay, D. Chang, C. Calik, L. Bassham, J. Kang, J. Kelsey *et al.*, "Status report on the second round of the nist lightweight cryptography standardization process," *National Institute of Standards and Technology Internal Report*, vol. 8369, no. 10.6028, 2021.

[48] E. Tehrani, T. Graba, A. S. Merabet, and J.-L. Danger, "Risc-v extension for lightweight cryptography," in *2020 23rd Euromicro Conference on Digital System Design (DSD)*. IEEE, 2020, pp. 222–228.

[49] M. K. Hasan, M. Shafiq, S. Islam, B. Pandey, Y. A. Baker El-Ebiary, N. S. Nafi, R. Ciro Rodriguez, and D. E. Vargas, "Lightweight cryptographic algorithms for guessing attack protection in complex internet of things applications," *Complexity*, vol. 2021, pp. 1–13, 2021.

[50] O. A. Khashan, R. Ahmad, and N. M. Khafajah, "An automated lightweight encryption scheme for secure and energy-efficient communication in wireless sensor networks," *Ad Hoc Networks*, vol. 115, p. 102448, 2021.

[51] N. A. Gunathilake, W. J. Buchanan, and R. Asif, "Next generation lightweight cryptography for smart iot devices:: implementation, challenges and applications," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. IEEE, 2019, pp. 707–710.

[52] N. A. Gunathilake, A. Al-Dubai, and W. J. Buchana, "Recent advances and trends in lightweight cryptography for iot security," in *2020 16th International Conference on Network and Service Management (CNSM)*. IEEE, 2020, pp. 1–5.

[53] I. Salam, T. H. Ooi, L. Xue, W.-C. Yau, J. Pieprzyk, and R. C.-W. Phan, "Random differential fault attacks on the lightweight authenticated encryption stream cipher grain-128aead," *IEEE Access*, vol. 9, pp. 72 568–72 586, 2021.

[54] V. A. Thakor, M. A. Razzaque, and M. R. Khandaker, "Lightweight cryptography algorithms for resource-constrained iot devices: A review, comparison and research opportunities," *IEEE Access*, vol. 9, pp. 28 177–28 193, 2021.

[55] T. K. Goyal, V. Sahula, and D. Kumawat, "Energy efficient lightweight cryptography algorithms for iot devices," *IETE Journal of Research*, vol. 68, no. 3, pp. 1722–1735, 2022.

[56] M. S. Turan, K. McKay, D. Chang, C. Calik, L. Bassham, J. Kang, J. Kelsey *et al.*, "Status report on the second round of the nist lightweight cryptography standardization process," *National Institute of Standards and Technology Internal Report*, vol. 8369, no. 10.6028, 2021.

[57] V. Rao and K. Prema, "A review on lightweight cryptography for internet-of-things based applications," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 8835–8857, 2021.

[58] J. Sun, H. Xiong, X. Liu, Y. Zhang, X. Nie, and R. H. Deng, "Lightweight and privacy-aware fine-grained access control for iot-oriented smart health," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6566–6575, 2020.

[59] F. Thabit, S. Alhomdy, A. H. Al-Ahdal, and S. Jagtap, "A new lightweight cryptographic algorithm for enhancing data security in cloud computing," *Global Transitions Proceedings*, vol. 2, no. 1, pp. 91–99, 2021.

[60] B. Aslan, F. Yavuzer Aslan, and M. T. Sakallı, "Energy consumption analysis of lightweight cryptographic algorithms that can be used in the security of internet of things applications," *Security and Communication Networks*, vol. 2020, pp. 1–15, 2020.

[61] L. Ning, Y. Ali, H. Ke, S. Nazir, and Z. Huanli, "A hybrid mcdm approach of selecting lightweight cryptographic cipher based on iso and nist lightweight cryp-

tography security requirements for internet of health things," *IEEE Access*, vol. 8, pp. 220 165–220 187, 2020.

[62] F. Thabit, S. Alhomdy, A. H. Al-Ahdal, and S. Jagtap, "A new lightweight cryptographic algorithm for enhancing data security in cloud computing," *Global Transitions Proceedings*, vol. 2, no. 1, pp. 91–99, 2021.

[63] A. Fotovvat, G. M. Rahman, S. S. Vedaei, and K. A. Wahid, "Comparative performance analysis of lightweight cryptography algorithms for iot sensor nodes," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 8279–8290, 2020.

[64] Y. Justindhas and P. Jeyanthi, "Secured model for internet of things (iot) to monitor smart field data with integrated real-time cloud using lightweight cryptography," *IETE Journal of Research*, pp. 1–14, 2021.

[65] B. Rezvani, F. Coleman, S. Sachin, and W. Diehl, "Hardware implementations of nist lightweight cryptographic candidates: A first look," *Cryptology ePrint Archive*, 2019.

[66] N. Uke, "Healthcare 4.0 enabled lightweight security provisions for medical data processing," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 11, pp. 165–173, 2021.

[67] J. Kaur, M. M. Kermani, and R. Azarderakhsh, "Hardware constructions for lightweight cryptographic block cipher qarma with error detection mechanisms," *IEEE transactions on emerging topics in Computing*, vol. 10, no. 1, pp. 514–519, 2020.

[68] L. Xiong, X. Han, C.-N. Yang, and Y.-Q. Shi, "Robust reversible watermarking in encrypted image with secure multi-party based on lightweight cryptography," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 1, pp. 75–91, 2021.

[69] R. Chatterjee, R. Chakraborty, and J. Mondal, "Design of lightweight cryptographic model for end-to-end encryption in iot domain," *IRO Journal on Sustainable Wireless Systems*, vol. 1, no. 4, pp. 215–224, 2019.

[70] J. Kaur, M. M. Kermani, and R. Azarderakhsh, "Hardware constructions for lightweight cryptographic block cipher qarma with error detection mechanisms," *IEEE transactions on emerging topics in Computing*, vol. 10, no. 1, pp. 514–519, 2020.

[71] M. K. Hasan, M. Shafiq, S. Islam, B. Pandey, Y. A. Baker El-Ebiary, N. S. Nafi, R. Ciro Rodriguez, and D. E. Vargas, "Lightweight cryptographic algorithms for guessing attack protection in complex internet of things applications," *Complexity*, vol. 2021, pp. 1–13, 2021.

[72] R. Bikmukhamedov and A. Nadeev, "Lightweight machine learning classifiers of iot traffic flows," in *2019 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO)*. IEEE, 2019, pp. 1–5.

[73] H. A. Wahsheh and M. S. Al-Zahrani, "Qr codes cryptography: A lightweight paradigm," in *International Conference on Information Systems and Intelligent Applications*. Springer, 2022, pp. 649–658.

[74] R. Bhaskaran, R. Karuppathal, M. Karthick, J. Vijayalakshmi, S. Kadry, and Y. Nam, "Blockchain enabled optimal lightweight cryptography based image encryption technique for iiot." *Intelligent Automation & Soft Computing*, vol. 33, no. 3, 2022.

[75] N. Thangamani and M. Murugappan, "A lightweight cryptography technique with random pattern generation," *Wireless Personal Communications*, vol. 104, pp. 1409–1432, 2019.

[76] R. R. Irshad, S. S. Sohail, S. Hussain, D. Ø. Madsen, M. A. Ahmed, A. A. Alattab, O. A. S. Alsaiari, K. A. A. Norain, and A. A. A. Ahmed, "A multi-objective bee foraging learning-based particle swarm optimization algorithm for enhancing the security of healthcare data in cloud system," *IEEE Access*, 2023.

[77] N. A. Gunathilake, A. Al-Dubai, and W. J. Buchana, "Recent advances and trends in lightweight cryptography for iot security," in *2020 16th International Conference on Network and Service Management (CNSM)*. IEEE, 2020, pp. 1–5.

[78] S. Atiewi, "Amer al-rahayfeh; muderalmiani; salman yussof; omar alfandi; ahed-abugabah; yaserjararweh,"scalable and secure big data iot system based on multifactor authentication and lightweight cryptography"," *IEEE Access*, vol. 8, pp. 113 498–113 511, 2019.

[79] A. Alahdal and N. K. Deshmukh, "A systematic technical survey of lightweight cryptography on iot environment," *International Journal of Scientific & Technology Research*, vol. 9, no. 3, 2020.

[80] M. A. Latif, M. B. Ahmad, and M. K. Khan, "A review on key management and lightweight cryptography for iot," in *2020 Global Conference on Wireless and Optical Technologies (GCWOT)*. IEEE, 2020, pp. 1–7.

[81] M. Rana, Q. Mamun, and R. Islam, "Current lightweight cryptography protocols in smart city iot networks: a survey," *arXiv preprint arXiv:2010.00852*, 2020.

[82] K. Arai, S. Kapoor, and R. Bhatia, *Proceedings of the Future Technologies Conference (FTC) 2020, Volume 3*. Springer Nature, 2020, vol. 1290.

[83] P. Shah, M. Arora, and K. Adhvaryu, "Lightweight cryptography algorithms in iot-a study," in *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*. IEEE, 2020, pp. 332–336.

[84] V. A. Thakor, M. A. Razzaque, and M. R. Khandaker, "Lightweight cryptography for iot: A state-of-the-art," *arXiv preprint arXiv:2006.13813*, 2020.

[85] A. Patil, S. Banerjee, and G. Borkar, "A survey on securing smart gadgets using lightweight cryptography," in *Proceedings of International Conference on Wireless Communication: ICWiCOM 2019*. Springer, 2020, pp. 503–515.

[86] X. Luo, L. Yin, C. Li, C. Wang, F. Fang, C. Zhu, and Z. Tian, "A lightweight privacy-preserving communication protocol for heterogeneous iot environment," *IEEE Access*, vol. 8, pp. 67 192–67 204, 2020.

[87] V. Rao and K. Prema, "A review on lightweight cryptography for internet-of-things based applications," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 8835–8857, 2021.

[88] Q. Huang, L. Wang, and Y. Yang, "Decent: Secure and fine-grained data access control with policy updating for constrained iot devices," *World Wide Web*, vol. 21, pp. 151–167, 2018.

[89] G. Ramu, "A secure cloud framework to share ehrs using modified cp-abe and the attribute bloom filter," *Education and Information Technologies*, vol. 23, no. 5, pp. 2213–2233, 2018.

[90] S. Atiewi, A. Al-Rahayfeh, M. Almiani, S. Yussof, O. Alfandi, A. Abugabah, and Y. Jararweh, "Scalable and secure big data iot system based on multifactor authentication and lightweight cryptography," *IEEE Access*, vol. 8, pp. 113 498–113 511, 2020.

[91] T. Ryffel, P. Tholoniat, D. Pointcheval, and F. Bach, "Ariann: Low-interaction privacy-preserving deep learning via function secret sharing," *arXiv preprint arXiv:2006.04593*, 2020.

[92] G. Liu, J. Lu, H. Li, P. Tang, and W. Qiu, "Preimage attacks against lightweight scheme xoodyak based on deep learning," in *Advances in Information and Communication: Proceedings of the 2021 Future of Information and Communication Conference (FICC), Volume 2*. Springer, 2021, pp. 637–648.

[93] M. F. Idris, J. S. Teh, J. L. S. Yan, and W.-Z. Yeoh, "A deep learning approach for active s-box prediction of lightweight generalized feistel block ciphers," *IEEE Access*, vol. 9, pp. 104 205–104 216, 2021.

[94] H. Li, K. Yu, B. Liu, C. Feng, Z. Qin, and G. Srivastava, "An efficient ciphertext-policy weighted attribute-based encryption for the internet of health things," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 5, pp. 1949–1960, 2021.

[95] P. Zeng, Z. Zhang, R. Lu, and K.-K. R. Choo, "Efficient policy-hiding and large universe attribute-based encryption with public traceability for internet of medical things," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 963–10 972, 2021.

[96] Y. Ming, B. He, and C. Wang, "Efficient revocable multi-authority attribute-based encryption for cloud storage," *IEEE Access*, vol. 9, pp. 42 593–42 603, 2021.

[97] R. Guo, G. Yang, H. Shi, Y. Zhang, and D. Zheng, "O 3-r-cp-abe: An efficient and revocable attribute-based encryption scheme in the cloud-assisted iomt system," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 8949–8963, 2021.

[98] H. Lu, R. Yu, Y. Zhu, X. He, K. Liang, and W. C.-C. Chu, "Policy-driven data sharing over attribute-based encryption supporting dual membership," *Journal of Systems and Software*, vol. 188, p. 111271, 2022.

[99] M. Elhoseny, K. Shankar, S. Lakshmanaprabu, A. Maseleno, and N. Arunkumar, "Hybrid optimization with cryptography encryption for medical image security in internet of things," *Neural computing and applications*, vol. 32, pp. 10 979–10 993, 2020.

[100] N. Alassaf and A. Gutub, "Simulating light-weight-cryptography implementation for iot healthcare data security applications," *International Journal of E-Health and Medical Communications (IJEHMC)*, vol. 10, no. 4, pp. 1–15, 2019.

[101] F. Masood, M. Driss, W. Boulila, J. Ahmad, S. U. Rehman, S. U. Jan, A. Qayyum, and W. J. Buchanan, "A lightweight chaos-based medical image encryption scheme using random shuffling and xor operations," *Wireless Personal Communications*, vol. 127, no. 2, pp. 1405–1432, 2022.

[102] O. A. Khashan and M. AlShaikh, "Edge-based lightweight selective encryption scheme for digital medical images," *Multimedia Tools and Applications*, vol. 79, no. 35-36, pp. 26 369–26 388, 2020.