An Efficient Approach to Design a New Asymmetric Key Encryption Algorithm Using Elliptic Curves

By

Baseerat Hayat

NATIONAL UNIVERSITY OF MODERN LANGUAGES

ISLAMABAD

November, 2025

An Efficient Approach to Design a New Asymmetric Key Encryption Algorithm Using Elliptic Curves

By

Baseerat Hayat

MS-Math, National University of Modern Languages, Islamabad, 2025

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

In **Mathematics**

To

FACULTY OF ENGINEERING & COMPUTING

NATIONAL UNIVERSITY OF MODERN LANGUAGES ISLAMABAD

THESIS AND DEFENSE APPROVAL FORM

The undersigned certify that they have read the following thesis, examined the defense, are satisfied with overall exam performance, and recommend the thesis to the Faculty of Engineering and Computing for acceptance.

Thesis Title: An Efficient Approach to Design a New Asymmetric Key Encryption Algorithm Using Elliptic Curves

Submitted By: Baseerat Hayat **Registration #:**81 MS/MATH/S23

Master of Science in Mathematics

Title of the Degree

Mathematics

Name of Discipline

Dr. Ghulam Murtaza _____

Name of Research Supervisor Signature of Research Supervisor

Dr. Anum Naseem _____

Name of HOD (Math) Signature of HOD

Dr. Noman Malik _____

Name of Dean (FEC) Signature of Dean (FEC)

November, 2025

AUTHOR'S DECLARATION

I Baseerat Hayat

Daughter of Khizar Hayat

Discipline Mathematics

Candidate of Master of Science in Mathematics at the National University of Modern Languages do hereby declare that the thesis An Efficient Approach to Design a New Asymmetric Key Encryption Algorithm Using Elliptic Curves submitted by me in partial fulfillment of MS degree, is my original work and has not been submitted or published earlier. I also solemnly declare that it shall not, in the future, be submitted by me for obtaining any other degree from this or any other university or institution. I also understand that if evidence of plagiarism is found in my thesis/dissertation at any stage, even after the award of a degree, the work may be cancelled and the degree revoked.

Signature of Candidate

Baseerat Hayat

Name of Candidate

November 24, 2025

Date

ABSTRACT

Title: An Efficient Approach to Design a New Asymmetric Key Encryption Algorithm Using Elliptic Curves

The Diffie-Hellman Key (DHK) protocol is highly effective for asymmetric keys, but if the parameters are not appropriately selected, it may be vulnerable to brute force attacks. Our research attempts to enhance the elliptic curve cryptosystem (ECC)-based asymmetric key scheme. In the improved DHK, the sender and recipient sharing method is utilized, in which they agree on an elliptic curve, but generator G is kept secret. We have proposed an ECC PRNG key exchange protocol that is more reliable than the previous one, as G is kept secret. Additionally, pseudo-random numbers are subjected to the suggested scheme. For the diffusion in this new public key technique, pseudo-random numbers are utilized. The system enhances its security against known or selected plaintext attacks by using a pseudo-random number stream generated from ECC to perform modular encryption. The suggested algorithm's security and simulation demonstrate its effectiveness, adaptability to various threats, and potential for practical use.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ECC	-	Elliptic Curve Cryptography
RSA	-	Rivest-Shamir-Adleman
DH	-	Diffie-Hellman
AES	-	Advanced Encryption Standard
ECDLP	-	Elliptic Curve Discrete Logarithm Problem
DSA	-	Digital Signature Algorithm
NIST	-	National Institute of Standards and Technology
ECDH	-	Elliptic Curve Diffie-Hellman
S-box	-	Substitution box
PRNG	-	Pseudo-random Number Generator
3DES	-	Triple Data Encryption Standard
DSS	-	Digital Signature Standard
NPCR	-	Number of Pixels Change Rate
UACI	-	Unified Averaged Changed Intensity
CML	-	Couple map lattices
NL	-	Nonlinearity
AC	-	Autocorrelation
LAP	-	Linear Approximation Probability
DAP	-	Differential Approximation Probability
SAC	-	Strict Avalanche Criterion
BIC	-	Bit Independence Criterion
SHA-256	-	Secure Hash Algorithm 256-bit

LIST OF SYMBOLS

E	-	Elliptic curve
$\mathbb{F}_p$	-	Finite field of prime order p
a, b	-	Curve parameters (coefficients)
U, V, W	-	Points on the elliptic curve
$U + V$	-	Elliptic curve point addition
G	-	Base point (generator)
n	-	Order of the base point G
(G, n, a, b, p)	-	Domain parameters for ECC over $\mathbb{F}_p$

ACKNOWLEDGMENT

First of all, I wish to express my gratitude and deep appreciation to Almighty Allah, who made this study possible and successful. This study would not be accomplished unless the honest espousal was extended from several sources for which I would like to express my sincere thankfulness and gratitude. Yet, there were significant contributors to my attained success and I cannot forget their input, especially my research supervisor, Dr. Ghulam Murtaza, who did not leave any stone unturned to guide me during my research journey. I really appreciate our HOD, Dr. Anum Naseem providing us with a research environment and kind support. I really want to say thanks to our respected teachers, Dr. Muhammad Rizwan, Dr. Sadia Riaz, Dr. Hadia Tariq, Dr. Shabeela Malik, and other teachers, for their guidance and support.

I shall also acknowledge the extended assistance from the administration of the Department of Mathematics, who supported me all through my research experience and simplified the challenges I faced. For all whom I did not mention but shall not neglect their significant contribution, thanks for everything.

DEDICATION

This thesis work is dedicated to my parents, family, and my teachers throughout my education career who have not only loved me unconditionally but whose good examples have taught me to work hard for the things that I aspire to achieve.

CHAPTER 1

INTRODUCTION AND LITERATURE REVIEW

In this chapter we discuss types of cryptography, the use of elliptic curves, and their advantages and disadvantages in cryptography. Further, we discuss the Diffie-Hellman key exchange and its computation over elliptic curves.

1.1 Background of Cryptography

For a long time, cryptography has been a widely used method of encoding messages. From ancient ciphers to modern cryptographic algorithms, the fundamental goal remains the same that is to use it for the confidentiality of data [1]. In the 1970's the most important advancement come to light was public-key cryptography which added the idea of public and private keys to the process of encryption. RSA and Diffie-Hellman behind them use a concept that their solutions take too long, for example, the factoring of integers or discrete logarithms. Cryptography has been known to play a crucial role in today's society and virtually in every electronic application and service such as secure email, online cash transaction, block chain technology and even in smart IoT devices [2].

Cryptographic techniques can be largely separated into two groups. These two are symmetric and asymmetric key encryption, commonly use in the today's world. Symmetric key cryptography uses a single secret key for both encryption and decryption in the advanced encryption standard AES. Although they are effective, symmetric key systems would prove problematic in key distribution and management. Two mathematically related keys are used in asymmetric key encryption, one key for sending the message and the other key for receiving the message. This is used for key exchange and can therefore be used in today's communication systems [3].

1.2 Elliptic Curve Cryptography

Since beginning, two main cryptosystems such as RSA and El-Gamal have appeared to be resistant to all attacks. Because of this, these two public key cryptosystems are currently the most widely used and regarded. One can make advantage of both digital signatures and encryption/decryption cryptosystems. It should be safe to utilize implementations of such cryptosystems since they are covered by all significant security standards. Elliptic curve cryptography (ECC) were created by Miller and Koblitz in 1985 and have gained popularity due to their efficiency and security. They are particularly relevant in today's fast-paced information technology, where mobile phones and handhelds require secure conversations. ECC's ideal qualities stem from the absence of sub exponential techniques in the elliptic curve discrete logarithmic problem, allowing for shorter keys for higher security levels [4].

Many contemporary cryptographic systems favor ECC over more conventional techniques like RSA or DSA because it provides robust security with significantly lower key sizes. ECC also allows for flexibility in secure system design by supporting several cryptographic protocols, including elliptic curve Diffie-Hellman, which is used for secure key exchange, and elliptic curve based DSA, which is used for digital signatures. These benefits, along with the support of prominent standards bodies like NIST, have made ECC the go-to option for putting safe, effective cryptography systems into place.

1.2.1 Advantages of Elliptic Curve Cryptography

1. **Smaller Key Sizes:** ECC is capable of providing an equivalent amount of security for different key sizes compared to RSA. For instance, an ECC key of 256 bits has the same strength as RSA key with 3072 bits [5]. Thus, reduction in key size results in improved computational cost and less demands on storage area.

2. **Efficiency:** Some ECC functions like scalar multiplication are less computationally expensive than RSA's modular exponentiation and therefore make ECC ideal for use in constrained devices.

3. **Security:** The ECDLP is more complex than the mathematical computations which are applied in RSA and Diffie-Hellman and is very resistant to cryptographic attacks.

4. **Scalability:** ECC has flexibility between devices and at the same time is applicable for use in the IoT devices for communication securely and also in blockchain systems.

1.2.2 Challenges in Elliptic Curve Cryptography

1. **Computational Cost:** Though ECC is faster than RSA, the operations on elliptic curves such as scalar point multiplication can put much load on devices with low computational capabilities.

2. **Implementation Vulnerabilities:** Unfortunately, if ECC is implemented in a wrong way, there is a threat of side-channel attacks, when an attacker engages the information leakage during the cryptographic operations implementation.

3. **Quantum Threat:** Some studies suggest that Goppal's attacks can be overcome using Shor's algorithm that breaks ECDLP which it an as liability of using ECC. Nevertheless, ECC remains safe from classical computing attack kinds.

4. **Key Mapping:** Converting plaintext to some points on the elliptic curve is quite demanding, and correct functional mapping techniques must be developed.

1.3 Comparison between RSA vs ECC

1. **Foundation of Mathematics:** RSA is based on the difficulty of factoring large integers, whereas ECC requires the complexity of the ECDLP.

2. **Security and Key sizes:** ECC offers security that is comparable to RSA, as seen in **Table 1.1** and **Figure 1.1**, but with much reduced key sizes.
   Effective scalar multiplication computation is essential to an ECC's performance. The

Table 1.1: Comparison of ECC and RSA Key Sizes at Equivalent Bit-Level Security

Bit-Level Security	**Key Size (in bits)**	
	ECC	**RSA**
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

security level of a 160-bit ECC key is equal to that of a 1024-bit RSA/DSA key. noted that 2048 bits were needed for really important keys and 1024 bits were needed for corporate

use in order to implement RSA security. Therefore, it is clear that ECC is superior to RSA since it can offer the same level of security with a shorter key length. The key sizes used in RSA and ECC are displayed in **Table 1.1**. A secure cryptosystem of ECC requires a key size of at least 160 bits.

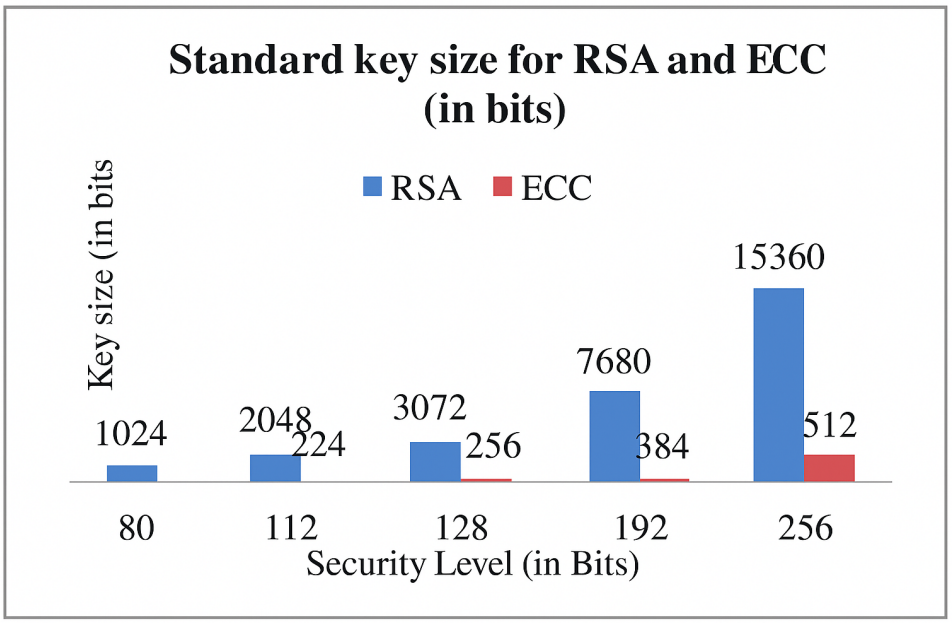


Figure 1.1: NIST's suggested security bit level

This suggests that for the same level of security, ECC can use more parameters than RSA. For example, RSA requires a key size of 2048 bits to achieve a security level of 112 bits, as shown in **Table 1.1** and **Figure 1.1**, while ECC requires a key size of 224 bits [6].

3. **Competence:** To generate keys, encrypt data, and decrypt it, ECC is far quicker than RSA, particularly on low-power devices like smart cards and smartphones. Because RSA heavily relies on modular exponentiation, it becomes slower as key sizes increase.

1.4 Diffie-Hellman Key Exchange

Two unknown parties can collaborate to generate a secure channel using a shared secret key by employing the Diffie-Hellman key exchange technique. The multiplicative group of integers modulo p, where p is a prime number and g is a primitive root modulo p, is used in the simplest and original implementation of the protocol. Diffie-Hellman key is illustrated in **Figure 1.2**.

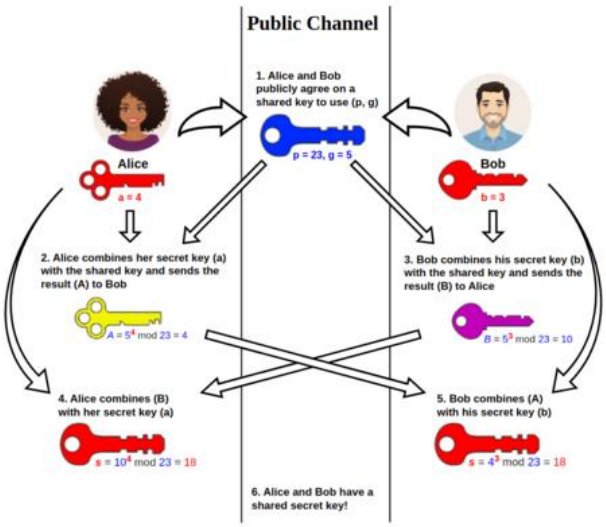


Figure 1.2: Diffie-Hellmen Protocol

1.5 Literature Review

Since ancient times, the technique of secret writing, or cryptography, has been used to conceal information or maintain the security of messages. [6] contrasted RSA with ECC. When it comes to key size utilization, ECC is lower while RSA is larger. Consequently, for the same security level, RSA takes longer to upload, encrypt, and decrypt than ECC. A number of encryption models are put out to secure the transport of crucial information by rendering it unintelligible. These models are based on chaotic maps, algebraic systems, and EC's. Modern cryptography is the design, development, and analysis of various mathematical techniques to ensure secure communication in the presence of opponent [7]. Asymmetric keys: RSA, ECC, and were used by H T Loriya et al. [8] and compared the sizes of their keys. In devices with limited resources, Bafandehkar et al. The concept of EC's in cryptography was initially proposed by Miller [9]. ECC is more efficient and provides better security with a smaller key size than traditional cryptosystems like the RSA protocol. Koblitz introduced the idea of the discrete logarithm problem, which is employed in Diffie-Hellman key cryptography, to the EC group [10]. Neal et al. [10] developed the concept of discrete logarithm problem for the generation of extremely secure and repid security systems. ECC deals with the construction of secure cryptosystems that ensure the protection of sensitive information. Numerous data security crypto-systems have been designed based on various mathematical structures and EC. In 1949, Shannon [11] made a significant contribution by establishing that to achieve high security in a cryptosystems, It must be able to cause data to become confused and diffuse up to a certain point. Neal [12]

designed an EC based cryptosystem over a finite field. Amara and Siad in [13] compared ECC with RSA. Furthermore, Hayat and Azam [14] presented a new method for image encryption that makes use of ECs. With a limit on the number of points on an elliptic curve, a hybrid image encryption method [14] based on a dynamic S-box and pseudo-random numbers (PRN) over an ordered elliptic curve has already been presented. Author et al. [15] presented the original Diffie-Hellman key exchange protocol, which is not unique to elliptic curves. The discrete log problem is transferred into the elliptic curve group in ECDH, the elliptic curve comparable of this basic concept. The ECDH key exchange technique uses ECC to securely transfer cryptographic keys between two parties. The public and private keys are generated by ECDH using elliptic curve mathematics. While the public key is made available to everyone, the private key is kept confidential. For the exchange of keys, the two sides share public keys and calculate a shared secret using their private keys. The shared secret can then be used to encrypt and decode messages between the two parties [16]. In [17], the author addresses elliptic curve operations and optimizations that are useful for ECDH and related protocols, although it is primarily concerned with pairs. The elliptic curve is used by many academics to create cipher images for image encryption systems. Each original image pixel [18] is transformed into the elliptic curve points (x, y) in a new image encryption process. A cipher image pixel is created using these elliptic curve points. When compared to other systems, the system offers tiny block size, great speed, and high level of safety. However, there aren't many curve points produced. If an attacker is able to predict the elliptic curve's basic parameters, they can also obtain these points. Therefore, great security may not be provided by mapping the image pixel process to the curve points. In order to improve the speed of data encryption and decryption and address the issue of key distribution, a combination of enhanced AES and ECC [19] encryption algorithms was previously developed. A new mapping scheme [20] has been developed for text messages, separating them into characters, transforming them into hexadecimal values, and calculating elliptic curve points. These points are appended to the sender's private key for encryption. However, this method's implementation is limited to text and should be used for audio and video data testing.

CHAPTER 2

PRELIMINARIES

This chapter provides a theoretical background on cryptographic methods, including elliptic curve and ECC-based pseudo-random number generators. Understanding these preliminary steps is crucial for designing, implementing, and assessing the proposed ECC-based image encryption system.

2.1 Cryptography

The art and study of secure communication methods in the presence of attackers is known as cryptography. To guarantee confidentiality, integrity, and authenticity, it involves converting data into a format that is unreadable (ciphertext), which only authorized parties can access. This is accomplished by encoding and decoding data using mathematical methods and keys.

2.2 Types of Cryptography

Cryptography can be divided into two main categories based on how encryption techniques are classified: symmetric and asymmetric key-based systems.

2.2.1 Symmetric Key Cryptography

Symmetric key cryptography, also known as secret key cryptography, involves data encryption and decryption using the same secret key, which must be kept secret by both sender and recipient. Examples include AES, DES, and 3DES.

2.2.2 Asymmetric Key Cryptography

Asymmetric cryptography, also known as public-key cryptography, involves two mathematically linked keys, a public key and a private key. The private key is confidential, while the public key can be shared freely. This approach ensures safe communication through protocols like

ElGamal, ECC, RSA, and DSS.

2.3 Elliptic Curve Basics

For a prime p, an elliptic curve over a field F_p is described as

$$E_{(a,b,p)} = \{(x,y) \in \mathbb{F}_p \times \mathbb{F}_p \mid y^2 \equiv x^3 + ax + b \pmod{p}\} \cup \{O\} \tag{2.1}$$

It is assumed that $4a^3 + 27b^2 \not\equiv 0 \ (mod p)$ and that $a, b \in \mathbb{F}_p$. O is a point at infinity. Let $U(x_1, y_1), V(x_2, y_2) \in \mathbb{E}_(p, a, b)$, $-U$ is calculated as

$$-U = \begin{cases} (x, p-y) & if \quad U \neq O \\ O & otherwise \end{cases} \tag{2.2}$$

If $U(x_1, y_1), V(x_2, y_2) \in \mathbb{E}_(p, a, b)$ *then* $W = U + V$ is provided as follows:

$$U + V = \begin{cases} U & \text{if } V = O \\ O & \text{if } U = -V \\ V & \text{if } U = O \\ W(x_3, y_3) & \text{if } Otherwise \end{cases} \tag{2.3}$$

In this case $(x_3, y_3) = (m^2 - x_1 - x_2, m(x_1 - x_3) - y_1)(mod\ p)$ where

$$m = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } U \neq V \\ \frac{3x_1^2 + a}{2y_1} & \text{if } U = V \text{ and } y_1 \neq 0 \end{cases} \tag{2.4}$$

Additionally, a graphical representation of the addition, point doubling, point at infinity when both y coordinates are zero and point at infinity when the coordinates are exact mirror images of one another is provided in **Figure 2.1**.

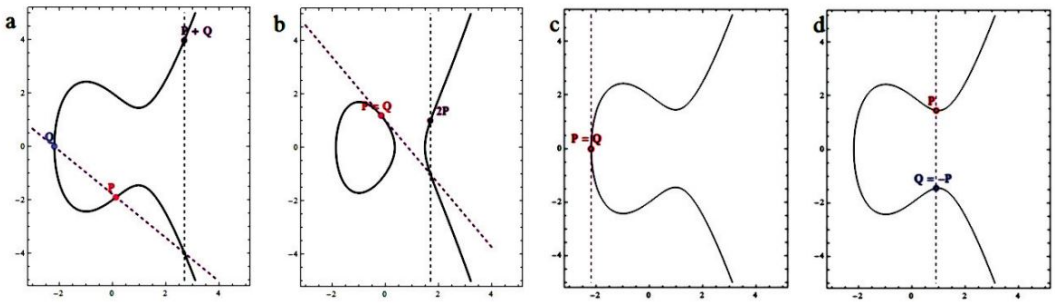


Figure 2.1: Fig (a) represents Point doubling, Fig (b) represents point addition, Fig (c) represents point at infinity when both y coordinates are zero, and Fig (d) represents point at infinity when the coordinates are exact mirror images of one another

Since $U \in E_{(a,b,p)}$ *and* $k \in F_p$, the elliptic curve scalar operation is

$$k\,U = \begin{cases} O \quad if \quad k = 0, \\ U + (k-1)U \quad otherwise. \end{cases} \tag{2.5}$$

The most basic scalar multiplication procedure is the double-and-add algorithm. The total number of points for a $E_{(a,b,p)}$ is represented by the notation $\#E_{(a,b,p)}$. Equation (2.6) uses Hasse's inequality to ensure that the points boundaries on the $E_{(a,b,p)}$,

$$\left|\#E(\mathbb{F}_p) - (p+1)\right| \leq 2\sqrt{p} \tag{2.6}$$

Eq. (2.7) defines a Mordell-Elliptic Curve (MEC).

$$E_{(p,a,b)} = \{(x,y) \in \mathbb{F}_p^2 \mid y^2 \equiv x^3 + b \pmod{p}, b \in (\mathbb{F}_p \backslash \{0\}) \cup \{O\}[21] \tag{2.7}$$

2.4 Elliptic Curve Logarithm Problem

In cryptography, the elliptic curve logarithm problem (ECDLP) is a mathematical problem. The task is to discover the integer n such that $V = nU$ given an elliptic curve defined over a finite field, a point U on the curve, and another point V that is a multiple of U. It basically asks for the "discrete logarithm" of V with regard to the elliptic curve's base point U [22].

2.5 Elliptic Curve Diffie Hellman (ECDH)

An ECC-based key exchange protocol that enables two parties to safely create a shared secret over an unprotected channel [23].

2.6 Public and Private Keys in ECC

Private key: An integer chosen at random acts like the private key.

Generation: Firstly we choose elliptic curve and a base point G of a prime order n then we generate a random integer a such that $1 \leq a \leq n-1$. we use a as a private key.

Public key: A point on the elliptic curve that is obtained from the private key is the public key.

Generation: The private key a is multiplied by the curve's base point (generator point) G to determine the public key P: $P = a \times G$ (where $\times$ stands for elliptic curve point multiplication) [24].

2.7 Concept of Base Point or Generator Point

In ECC algorithms, a predetermined point G on the elliptic curve is used to generate keys.

Example: The base point G in the secp256k1 curve, which is used in Bitcoin, is a particular point that is identified by its x and y coordinates [25].

2.8 Diffie-Hellman Protocol

A cryptographic technique called the Diffie-Hellman protocol enables two people to create a shared secret key via an unsecured channel, which can then be utilized for secure communication. It prevents intercepting by facilitating key exchange without requiring the transmission of the actual key [26].

2.9 Concept of Key Generation in ECC

In elliptic curve cryptography, key generation is the process of creating a private key and corresponding public key. First, an elliptic curve over a finite field and a base point G with a big prime order are selected. The private key is a randomly chosen big number a, where $1 < a < n$, and must be kept secret. The elliptic curve point $P = a \cdot G$ is then obtained by multiplying the base point by the private key to generate the public key. Because of the difficulty of the ECDLP, it is computationally impossible to determine the private key from the public key, making ECC a secure and efficient cryptographic system [27].

2.10 Key Generated through the ECDH Protocol

Consider the public keys for the elliptic curve $y^2 = x^3 + ax + b$ over finite field $\mathbb{F}_p$, where G is a generator, $P_a = n_a G$ and $P_b = n_b G$, which are generated by the sender's $n_a G$ and the recipient's $n_b G$, respectively. The shared secret key is generated as $n_b P_a$ by the sender and $n_a P_b$ by the recipient [28]. Consequently,

$$n_a n_b G = n_a P_b = n_b P_a = n_b n_a G \tag{2.8}$$

2.11 Pseudo-random Number Generation

An algorithm that generates a random, deterministic series of numbers.

Use of ECC in PRNG

1. For strong security with small key sizes
2. For robust algebraic structure
3. For unpredictability and high entropy
4. Ideal for environments with limited resources [29]

2.12 S-box (substitution box)

A fundamental part of block ciphers, an S-box (Substitution box) substitutes input bits with output bits in a nonlinear manner to create confusion and prevent cryptanalysis. It functions

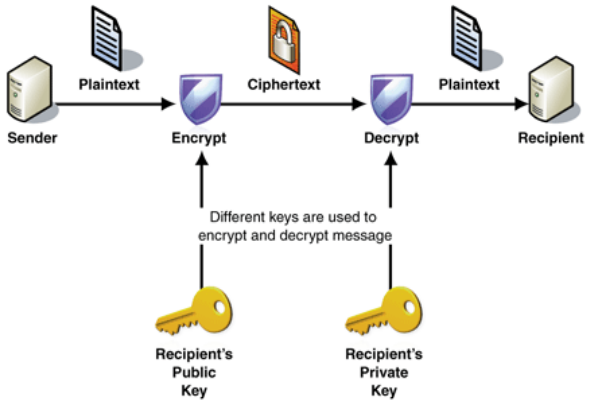


Figure 2.2: Encryption and Decryption Process

as a lookup table, changing data in a way that is hard to undo without the secret key. It can be dynamically generated (like in the AES) or fixed (like in the DES).

2.13 Cryptosystem

A tool for converting plain text to cipher text and vice versa is called cryptosystem. A cryptosystem is able to convert plain text into cipher text and vice versa. Depending on the keys used for encryption and decryption, either it is a symmetric or asymmetric key. The encryption and decryption process is illustrated in **Figure 2.2**.

Encryption is the process of transforming a plain text into a form that is unreadable whereas decryption is the process of transforming an unreadable message (cipher text) into a clear and readable format. The encryption and decryption procedures are managed by one or more cryptographic keys. The parameters that determine a particular involuntary transformation in this system are called a set of keys. One or more cryptographic keys govern the encryption and decryption process. In general, the following mathematical explanation can be used to describe the encryption and decryption process:

$EK(S) = C$ **(Encryption Process)**

$DK(C) = S$ **(Decryption Process)**

Using the key K, we declare the message S to be message C. Then, in the decryption process, we use the key K to execute the encrypted message C and obtain the original message, S [30].

2.14 Cryptographic Analysis

2.14.1 Differential Cryptanalysis

Differential cryptanalysis is a technique used to decipher and analyze cryptographic algorithms, particularly block ciphers. It examines how variations in input pairs impact output pairs,

helping attackers determine secret keys. If a small alteration in plain images results in different cipher images, encryption algorithms are resistant to differential attacks. Methods like NPCR and UACI are used to assess algorithmic sensitivity against differential attacks.
The ciphered images C_I and C'_I of $I_{N\times M}$ and $I'_{N\times M}$ plain-images are produced by altering a single pixel in UACI-NPCR analysis. Equations (2.9) and (2.10) are used to calculate the NPCR and UACI.

$$NPCR = \frac{1}{N\times M}\sum_{i,j} D(i,j)\times 100 \tag{2.9}$$

$$UACI = \sum_{i,j}\frac{|I_C(i,j) - I'_C(i,j)|}{N\times M\times 255}\times 100 \tag{2.10}$$

where if $C_I(i,j) - C'_I(i,j) \neq 0$ and $D(i,j) = 0$ otherwise, then $D(i,j) = 1$ [31].

2.14.2 Test of Entropy

The most crucial metric for measuring the degree of unpredictability in a dataset of images is entropy. It displays the distribution of pixel intensity in an image. If I were a grayscale image, the entropy S equation would be

$$S = -\sum_{i=0}^{255} P(x_i)\log_2(P(x_i)) \tag{2.11}$$

High entropy indicates high randomness in image data [31].

2.14.3 Histogram Test

In cryptography, a histogram test is a statistical method for determining whether encrypted data is random or not. It determines if the distribution of byte or symbol frequencies in cipher text looks uniform, which is an ideal quality of secure encryption. A histogram test graphs the frequency of each symbol or byte in cipher text. If a histogram appears too "structured" or "non-random," it can indicate patterns or vulnerabilities in the encryption algorithm. Good encryption must hide all patterns of the original plain text. **Figure 2.3** show the uniform distribution proves strong encryption [31].

2.14.4 Correlation Test for Image Encryption

Pixel correlation is typically high in plain images. Creating encrypted images with minimal correlation is essential for a perfect cryptosystem. Pixel's correlation coefficient is computed as

$$\sigma_{xy} = \frac{\sum_{i=1}^{M}(x_i - F(x))(y_i - F(y))}{\sqrt{\sum_{i=1}^{M}(x_i - F(x))^2\sum_{i=1}^{M}(y_i - F(y))^2}} \tag{2.12}$$

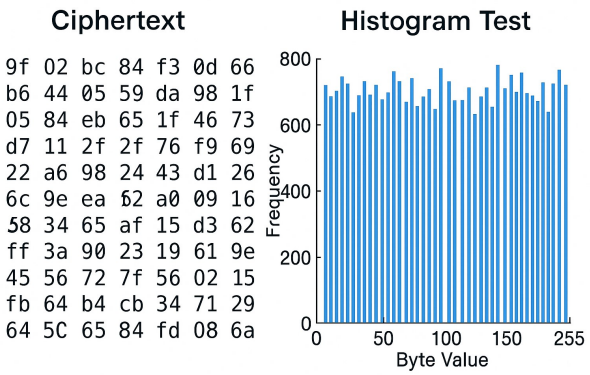


Figure 2.3: On left figure showing cipher text and right side showing uniform distribution proves strong Encryption

Under the requirement that

$F(x) = \frac{1}{M}\sum_{i=1}^{M} x_i$

M is the data size in this case. F is the expected value opertaor and σ is the correlation coefficient [31].

CHAPTER 3

A NOVEL IMAGE ENCRYPTION SCHEME BASED ON ELLIPTIC CURVES AND COUPLED MAP LATTICES

3.1 Overview

Research in [31] discusses the security of images and data sharing over the Internet, specifically digital images. Traditional encryption algorithms have limitations and inefficiencies. A new image encryption approach based on elliptic curves (ECs) and coupled map lattices (CMLs) is developed for real-time transmission of images. The method is resistant to brute-force attacks and has a large keyspace. The algorithm can encrypt an image of size 256×256 in just 0.641 seconds.

3.2 Pseudo-random Numbers Generation

In this article the author generated 2D PRNGs by using the following parameters

A prime p' such that p is equivalent to 2 *mod* 3

Selection of b' is selected from the range $[1, p'-1]$ *and* $m_x, m_y \in [1, p']$

Assume that $\mathbb{Z} \times \mathbb{Z}$ contains M as a subset.

Define transformation $\alpha_{M,p',b',m_x,m_y,}$ from M to $[0,\ m_x - 1]$ such that for each $(a_1, a_2) \in M$ and $((a, a_1 + a_2)\ mod\ m_y) \in E_{,0,b',p'}$ it states this

$$\alpha_{M,p',b',m_x,m_y}(a_1, a_2) = a\ (mod\ m_x) \tag{3.1}$$

Author use the following fixed parameters to examine the sequence's unpredictable behavior α_{M,p',b',m_x,m_y} for images of various sizes: $p' = 1048847, b' = 1, m_y = p'$ *and* $m_x = 256$. Entropy

and period analyses of PRNs produced using our approach are provided in **Table 3.1**. To create

Table 3.1: Comparison of entropy and period values for various 256×256 plain-images

Image	**Entropy**	**Period**
All White	7.9972	65536
All Black	7.9971	65536
Lena	7.9930	65536
Mandrill	7.9932	65536
Pepper	7.9936	65536
Cameraman	7.9941	65536
Upper Bound of Entropy		8.0000

PRNs, a collection of plain images (such as peppers, Mandrill, Lena, and Cameraman) is utilized in a set $M \subset \mathbb{Z} \times \mathbb{Z}$ *in* $\alpha_{M,m_x,m_y,p',b'}$

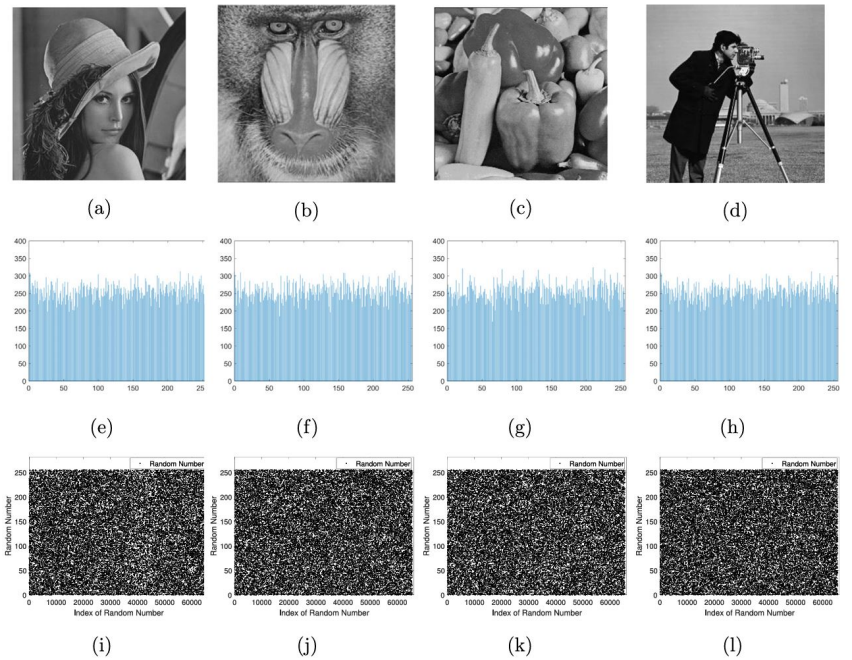


Figure 3.1: using the variables $(p' = 1048847, b' = 1, m_y = p, m_x = 256)$ pseudo random number generated from the sequence α_{M,p',b',m_x,m_y} are evaluated for four grayscale images of size 256×256 Peppers, Mandrill, Lena, and Cameraman Subfigures (a)–(d) represent the source images;(e)–(h) display histograms of the generated PRNs;(i)–(l) visualize the PRNs themselves for each corresponding image.

Figure 3.1 show the images of PRN and Histogram. **Figure 3.2** shows PRN's graphs for images as well as all-white and all-black histograms.

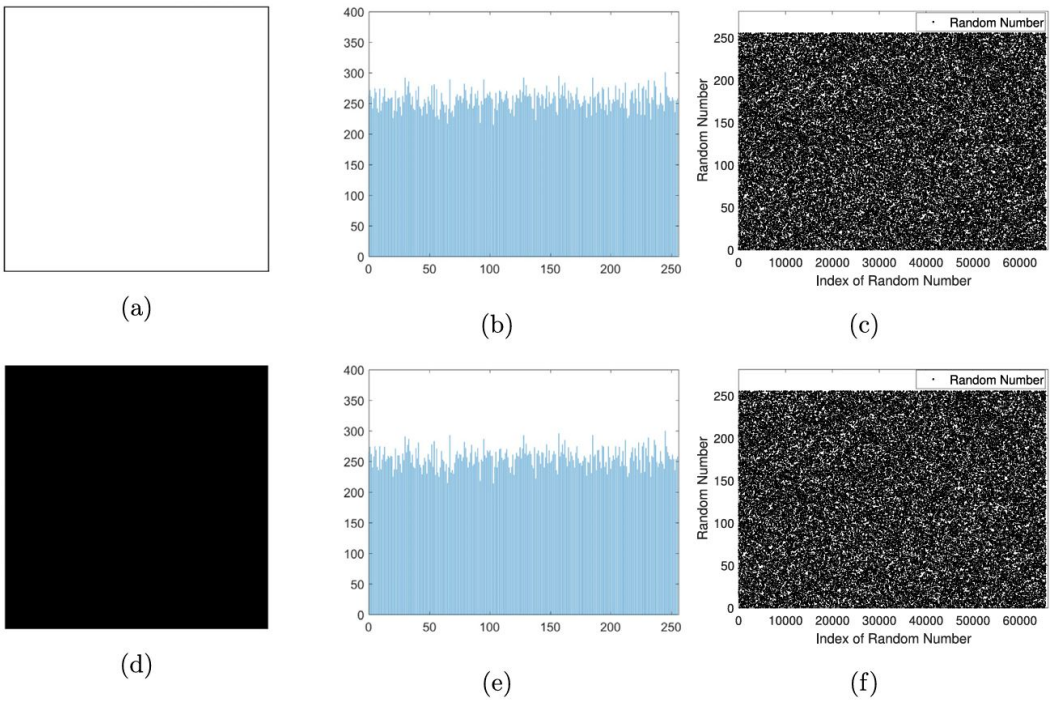


Figure 3.2: For images where every pixel is set to 255 (all-white) or 0 (all-black) for parameters of PRNs analysis $(b' = 1, m_y = p, m_x = 256, p' = 1048847)$ (a) All-white plain-image (b)PRN histogram produced from the image (a) ;(c)PRN's produced from the image in (a) ;(d) All-black original-image;(e)PRN histogram produced from the image in (d); (f) pseudo random numbers produced using the image in (d)

The PRNs in **Table 3.1** have entropy values that are approximately around the upper bound. Additionally, each gray image's PRN histogram is consistent, as shown in **Figure 3.1 and 3.2**. High periods and entropy values are produced by the suggested PRN generator, as **Table 3.1** demonstrates. Thus, in a plain image, it can produce high diffusion.

3.3 Couple Map Lattice

A logistic map is one where $\lambda x_n(1-x_n) = x_{n+1}$ for $(0,1) \in x_n$ and $\lambda \in (0,4]$. The definition of a CML-system is

$$x_{n+1}(j) = (1-\xi) f(x_n(j)) + \frac{\xi}{2}[f(x_n(j+1)) + f(x_n(j-1))] \tag{3.2}$$

where the lattice site index is $j = 1, 2, ..., \tau$, the coupling constant is $\xi \in [0\ 1]$, the time variable is n, the real mapping is f, and τ indicates the complete number of lattices that need to be formed.

It is assumed that $x_n(0) = x_n(\tau)$.

3.4 A Dynamic S-box Generator that Makes Use of a Couple Map Lattices and an Elliptic Curve

A chaotic substitution box generator uses an elliptic curve and couple map lattice system to construct $m \times m$ substitution boxes, selecting variables from $E_{a,b,p}$ and its generator G, and a set T and string S is proposed in this section.

- **Establishing an EC and CML-systems' parameters:** Choose G as a generator of the parameters a, b, p and the EC $E_{a,p,b}$. Determine x_i, λ, ξ and $x_j = mod(z + x_j - 1, 1), j = 1, 2, ..., \tau$. The total lattices is τ, and for each lattice τ, z is utilized to construct the initial parameter x_j.

- **Point generation on an EC:** Use G to create a sequence of points on the $E_{a,p,b}$. With $0 < l \leq m$, Suppose $S = \langle G \rangle$ so that $|S| = 2l$. Set up $n = m - l$, as well as select T as a set where $T = \{t_1, ..., t_n\} \subset \{0, 2m-1\}$, $t_j \neq t_i$ for $i \neq j$, meaning that $T = 2m(mod\ 2m - i)\ for\ i = 1, ..., n$.

- **Creation of a integer series depend on EC:** With $S_i = S_y(mod\ t_i)$, $i = 1, ..., n$, and S_y being the EC points where Y-coordinate in S, for example. Concatenate $(||)$ each S_i to create a collection X, so that $X = [S_1||S_2||...||S_n]$.

- **Chaotic system iteration:** To obtain a sequence unaffected by the initial conditions, Eq. (3.2) iterates the CML-system $2m + \alpha$ times for $\alpha \geq 10$, removing the initial α iterations. Transform every CML iteration into an integer number using the formula:

$$Y_i(j) = [U_i(j) \times 10^{12}] \tag{3.3}$$

- **Designing an integer sequence based on CML:** Define an integer sequence $Y_j \subset [0, 2m - 1]$ for every lattice j for a $m \times m$ S-box by using modulo $2m$ as
$Y(j) = [Y_i(j)\ mod\ 2^m\ , j = 1, 2, ..., \tau, i = 1, ...2^m$.

- **S-box generation:** By switching the entries of the original S-box $C_0 = \{0, 1, ..., 2m-1\}$, create candidate S-boxes B_i, $i = 1, ..., 2m$. Make use of sets Y and X to iteratively conduct the exchange operation $(\leftrightarrow)$ on C_0 for the S-boxes B_i, that is,
$C_0(Y(i) + 1) \leftrightarrow C_0(X(i) + 1)$

The last B_i is the wanted S-box. **Figure 3.3** shows the flow chart for our S-box generator.

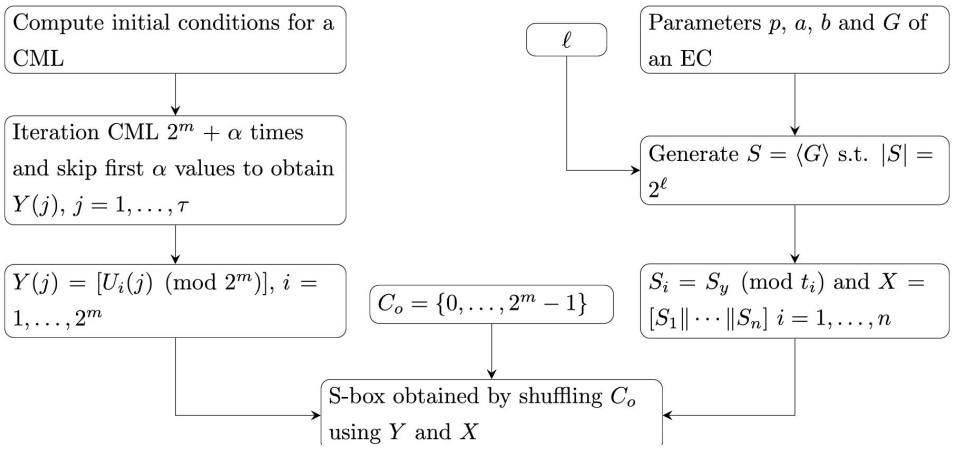


Figure 3.3: Flow chart of the generator

3.4.1 Review of S-box Generator

We will talk about the generator's experimental analysis in this section. To do this, we utilize the following 256-bit parameters for an $E_{a,b,p}$,
p=115792089210356248762697446949407573530086143415290314195533631308867097853951,
a=115792089210356248762697446949407573530086143415290314195533631308867097853948,
b=115792089210356248762697446949407573530086143415290314195533631308867097853951,
G=[48439561293906451759052585252797914202762949526041747995844080717082404635286,
36134250956749795798585127919587881956611106672985015071877198253568414405109].
Using a $E_{a,b,p}$ with the previously mentioned parameters and a couple map lattice with various initially conditions $x_0 \in (0,1)$, and setting additional criteria as $\xi = 0.4254$, $\lambda = 3.9575$, $\alpha = 10$, $l = 7$, $z = 0.7500$ *and* $T = \{256, 255\}$,we produced a random collection of 10,000 S-boxes. An substitution box produced by suggested generator is displayed in **Table 3.2**. The performance of this generator for image encryption is assessed using the following tests:

- **Sensitivity:** Sensitivity is the impact of input on the generator of an substitution box output. A generator of an S-box with high sensitivity is essential for the encryption of images. since it strengthens the ability of an encryption algorithm to withstand differential assaults. The generated S-box is greatly affected when the light changes in either x_0 or ξ. Therefore, we may conclude that the generator is effective and appropriate for use in encryption applications.

- **Critical point:** There is no critical point because the suggested generator generates a substitute box for every suitable set of variables. This generator function expedites the

Table 3.2: Our method produced an S-box with the following parameters: $T = \{256, 255\}$, $x_0 = 0.7500$, $\lambda = 3.9575$,$\alpha = 10$, $\ell = 7$, $z = 0.7500$ and $\xi = 0.4254$ where a, b, p, and G are the same as those provided as above section

53	85	207	89	213	173	66	80	162	132	142	93	232	9	105	59
151	17	33	166	23	64	199	81	171	189	95	217	233	20	245	148
209	231	11	92	156	6	178	114	45	146	253	19	241	160	228	78
158	188	120	73	67	195	83	116	41	239	186	130	227	119	123	236
192	51	36	161	57	140	205	94	220	170	60	118	152	62	40	99
5	237	167	4	226	179	121	55	110	149	187	22	169	76	229	70
46	240	176	39	63	27	234	117	164	90	144	182	30	102	97	242
112	65	68	122	155	180	72	211	135	196	200	183	103	141	150	247
107	125	3	98	28	230	104	204	218	16	197	214	185	249	101	1
208	7	24	246	193	82	91	250	201	153	71	133	86	108	49	216
221	26	21	58	168	255	106	42	29	75	154	0	202	136	18	111
244	165	69	198	87	177	113	181	61	243	52	2	203	25	235	8
77	157	137	225	88	163	223	212	96	210	147	109	134	159	175	252
38	139	48	138	12	124	115	174	251	128	222	184	44	84	191	143
32	74	131	10	129	215	34	248	79	219	145	190	206	54	238	254
15	172	47	100	43	126	35	127	224	56	37	13	14	194	31	50

encryption process.

- **Cryptography Output strength: Table 3.2** lists our generator's NL (minimum, average, and maximum) as well as the possible schemes for 10,000 S-boxes. The findings demonstrate that the recently constructed generator is capable to create S-boxes with an average NL of 97.70 and $80 \leq NL \leq 106$. Thus, the substitution box generation technique outperforms the generators in [14, 32, 33, 34, 35], as seen in **Table 3.3**. NL outcomes The paper is taken from [14, 32, 33, 34] for 10,000 S-boxes [36]. **Table 3.3** provides the

Table 3.3: NL evaluation of 10,000 S-boxes.

Generator	**NL**		
	maximum	**average**	**minimum**
suggested	106	97.70	80
Ref. [14]	102	92.05	64
Ref. [34]	104	97.45	82
Ref. [32]	104	84.64	52
Ref. [35]	104	99.27	84
Ref. [33]	106	90.20	0

performance analysis using S-boxes in [34, 37, 38, 39, 40, 41, 42, 43, 44]. compared to existing S box's, our S box improved non linearity in [34, 37, 38, 39, 40, 41, 42, 43, 44], and its BIC-NL is superior to that in [38, 39, 41, 43]. Linear approximation probability characterizes the S box linearity of 0.148 and a differential approximation probability of 0.047 [45, 46].

3.5 Encrypting and Decrypting Images

Alice sends Bob an image, and Bob sends a simple image of $I_{u\times v}$. The hash value of I is determined, and CML-systems are initialized. Two chaotic sequences are generated, and a diffused image D is formed. An encrypted image is created by shuffled rows and columns.

1. **Computation of SHA-256**

   Suppose we have a image I with an m×n dimension (256×256 pixels). we compute SHA-256 hash of image I. This produces 32 bytes, or a 256-bit output. In second step we divide the hash into thirty-two parts the names of each byte are $h_1, h_2, ., h_{32}$. These integers

Table 3.4: comparative analysis of S-box based on existing schemes

S-box	AC	DAP	LAP	NL	SAC Min	SAC Max	BIC Min	BIC Max	NL Avg
Suggested	254	0.047	0.148	106	0.406	0.641	0.459	0.523	98
[34]	253	0.054	0.141	104	0.406	0.594	0.461	0.522	98
[38]	255	0.039	0.152	100	0.391	0.586	0.468	0.537	100
[38]	255	0.047	0.125	96	0.422	0.609	0.471	0.547	96
[39]	254	0.054	0.133	98	0.422	0.609	0.477	0.535	94
[40]	255	0.039	0.125	96	0.359	0.609	0.477	0.541	98
[41]	254	0.039	0.149	102	0.422	0.594	0.461	0.527	96
[42]	255	0.039	0.145	104	0.391	0.625	0.471	0.531	98
[37]	254	0.047	0.125	102	0.422	0.641	0.477	0.533	100
[43]	255	0.039	0.133	104	0.359	0.609	0.457	0.535	96
[44]	254	0.039	0.133	104	0.438	0.641	0.475	0.547	98

range from 0 to 255 and are 8-bit.Convert them to decimal if they are hexadecimal. Now we have 32 numbers in a set $h_1, h_2, ., h_{32}$. These will be used to generate the chaotic map parameters in Step 2 of image encryption and decryption. The image content is tightly linked to the hash, which is a secret key generator. As a result, if the image is modified, the hash is also changed, which changes the encryption output. ensures image content sensitivity, which is a key component of secure encryption.

2. **Choosing parameters to generate permutations in the CML system:**
   We generate $s(u,a,b,p)$ *and* $s(v,a,b,p)$, two permutations on $[1,v]$ and $[1,u]$, consequently. Pick two subsets for $D_u = \{h_1, h_2, ..., h_{16}\}$ *and* $D_v = \{h_{17}, h_{18}, ..h_{32}\}$ *of* $\{h_1, h_2, ..., h_{32}\}$. Now we calculate using Eqs.(3.4) and (3.5) of two sets of parameters. $d_u = \{K_1^{(u)}, K_2^{(u)}, K_3^{(u)}, K_4^{(u)}\}$ *and* $d_v = \{K_5^{(u)}, K_6^{(u)}, K_7^{(u)}, K_8^{(u)}\}$. Compute parameters k_i, $i = 1,2,...,8$ as follows: $K_i = \sum_{i=s}^{r} h_i$, whereas $r = 1+4(\text{i}-1)$ while $s = 4\text{i}$. If q is the set $\{h_1, h_2, ..., h_{32}\}$ and the hash values of the arithmetic mean, then the members of sets d_u d_v are calculated based on

$$\begin{cases} K_1^{(u)} = 3.75 + mod(q, 0.25 + \frac{k_1}{2^n}) \text{ and} \\ K_i^{(u)} = mod(q, 1 + \frac{k_1}{2^n}), for\ i = 2, .., 4. \end{cases} \tag{3.4}$$

$$\begin{cases} K_5^{(v)} = 3.75 + mod(q, 0.25 + \frac{K_5}{2^n}) \; and \\ K_i^{(v)} = mod(q, 1 + \frac{k_i}{2^n}), for \; i = 6,7,8. \end{cases} \tag{3.5}$$

3. **Permutation construction**

   Establish the following beginning conditions for the CML. Using the permutation $S(a,b,u,p)$: Applying the variables of the elliptic curve (a,b,p,G), create a set of points $S = \langle G \rangle$ G on an elliptic curve: $\lambda = K_1^{(u)}, \varepsilon = K_2^{(u)}, x_o = K_3^{(u)}$ *and* $z = K_4^{(u)}$. Now use the procedure as above to create an substitution box $S(a,b,p,u)$. Establish the following beginning conditions for a CML about the permutation process $S(v,a,b)$: Using $\lambda = K_5^{(v)}, \varepsilon = K_6^{(v)}, x_o = K_7^{(v)}$ *and* $z = K_8^{(v)}$ while parameters of an EC (a,b,v,p), Use the above procedure to create an substitution box $S(v,a,b,p)$. Suppose λ *and* σ be used for efficiency represent $S(p,a,v,b)$ and $S(p,u,a,b)$, consequently.

4. **The creation of PRN variables:**

   Assume that a precomputed $E_{0,b',p'}$ for a parameter $b' \in [1, p'-1]$ and a prime $p \equiv 2 \; (mod 3)$ is agreed upon by the sender and the recipient. Pick $m_x = h_i \times h_j$ and $m_y \in \{h_i \cdot h_j \mid i,j \in [1,32], \; h_i \neq 0, \; h_j \neq 0\}$. Lastly, calculate these equations for the plain-image p the sequence is γ_{P,p',b',m_x,m_y}

5. **Diffusion locally**

   Make a diffused image D of the plain-image I such that $d(i;D) = (d(i;I) + \alpha_{b',p',m_x,m_y,P}(i, d(i;P)))\; (mod|s|)$ for each integer $i \leq uv$.

6. **Diffusion globally**

   From top to bottom, let $R_i, i \in [1,u]$ represent the i^{th} row of D. Next, We get an image $\sigma(D)$ so that the i^{th}, the i^{th} row of $\sigma(D)$ where $i \in [1,u]$ is $R_{\sigma(i)}$. Going from left to right, suppose $C_i, i \in [1,v]$ represent the i^{th} column of $\sigma(D)$. $\lambda(\sigma(D))$ is the cipher text of I is then obtained so that the i^{th}, $i \in [1,u]$ and $C_{\lambda(i)}$ is the column of $\lambda(\sigma(D))$. **Figure 3.4** showing the encryption algorithm flowchart and an illustration of our encryption method is displayed in **Figure 3.5** for a 4 by 4 image.

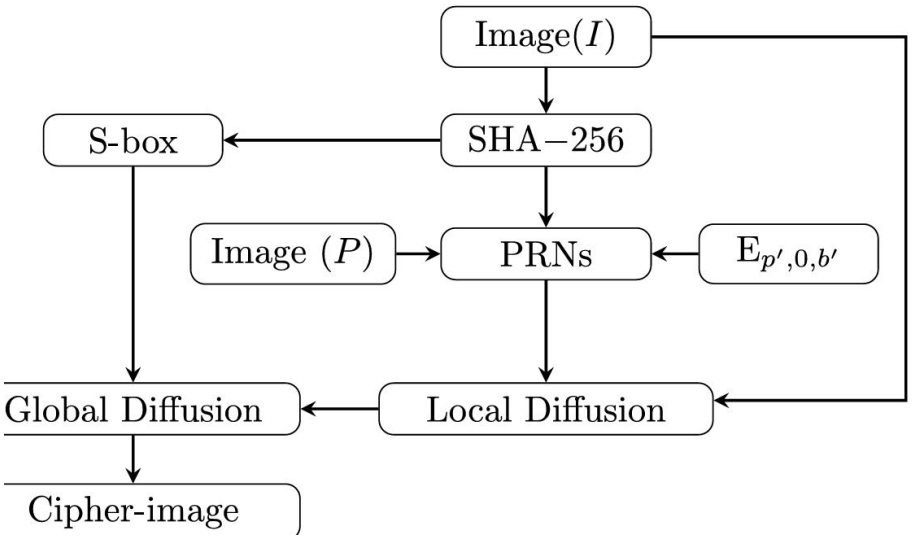


Figure 3.4: Represents the encryption algorithm's flowchart.

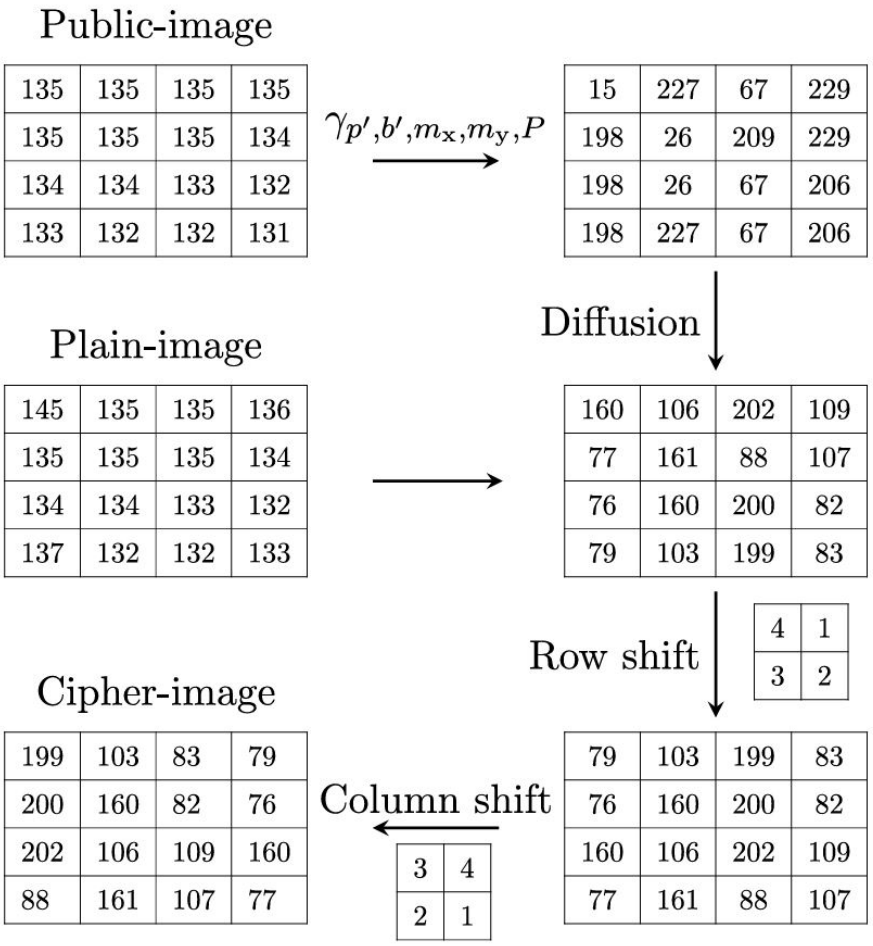


Figure 3.5: An illustration of our encryption method is displayed for a 4 by 4 image.

3.5.1 Process of Decryption

In contrast to the encryption approach, this scheme allows for decoding. λ^{-1} and σ^{-1}, the reverse substitution boxes must be understood for this. The CML and $E_{p,a,b}$ parameters allow for the complete derivation of these S-boxes. Alice is able to obtain the plain image $I_{u\times v}$ by doing these steps.

1. Determine $I's$ hash value.

2. Use the method outlined above to create the permutations σ and λ. Find the inverse S-boxes for σ^{-1} and λ^{-1} of σ and λ, respectively.then use Equation (3.6) to obtain $d(i,D)$.

$$d(i,D) = \sigma^{-1}(\lambda^{-1}(C_I)) \tag{3.6}$$

3. For the plain-image P, calculate the sequence γ_{P,p',b',m_x,m_y}.

4. For each integer $i \leq uv$, find the plain-image. $d(i;I) = (d(i;D) + \gamma_{P,b',p',m_x,m_y}(i, d(i;P)))\ (mod|s|)$ is the condition.

3.6 Analysis of Security

Now, we use the standard 256×256 Lena image and every image from the USC-SIPI database for encryption. Every image in the USC-SIPI database has the dimensions $k \times k = 256, 512, 1024$. Every image is subjected to security analysis. We executed the encryption schemes using MATLAB R2017a. Every experiment uses a PC with an Intel(R) Core(TM) i5 processor running at 3.20 GHz, Microsoft Windows 10/64 bit, and 8.00 GB of RAM. The public image P utilized for analysis in this section is an all-white image. The image of Lena, sized 256×256 used in this section has a hash value of 663124595124170100185186724251131341822297710310
392243116172166113131491
52221972. For a $E_{a,b,p}$, the parameter set (a,b,p,G) is similar to the one above.

3.6.1 Difference based Cryptoanalysis

If a small alteration in plain images results in noticeably different cipher images, So, the encryption algorithm is resistant to differential attacks from a cryptographic perspective. The UACI [47] and NPCR [47] are two techniques for evaluating how sensitive encryption algorithms are to algorithmic assaults that target differentials. Differential attacks are essentially pointless if a minor alteration to a plain image does not result in inconsistent or varied encryption behavior

across the cipher image. To generate the ciphered images C_I and C'_I of $I_{N\times M}$ and $I'_{N\times M}$ plain-images, a single pixel is altered in UACI-NPCR analysis. Equations (3.7) and (3.8) are used to calculate the NPCR and UACI.

$$NPCR = \frac{1}{N\times M}\sum_{i,j} D(i,j)\times 100 \tag{3.7}$$

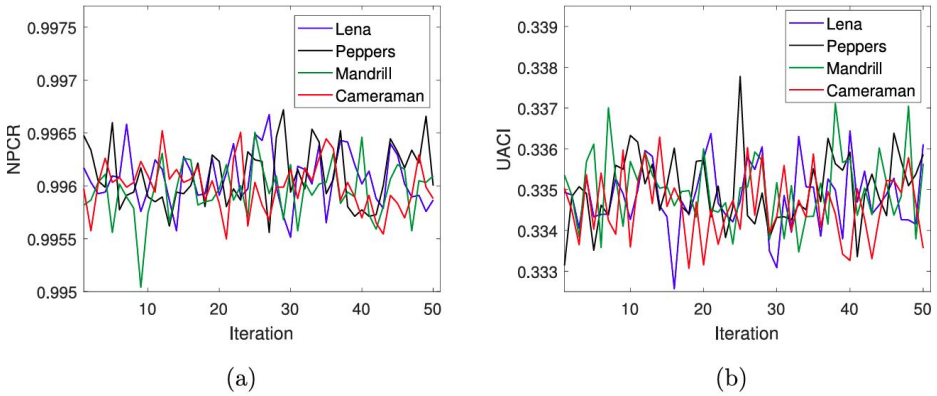
$$UACI = \sum_{i,j}\frac{|I_C(i,j) - I'_C(i,j)|}{N\times M\times 255}\times 100 \tag{3.8}$$

where if $C_I(i,j) - C'_I(i,j) \neq 0$ and $D(i,j) = 0$ otherwise, then $D(i,j) = 1$. The proposed method performed 50 NPCR and UACI tests on various images by altering a single pixel at various plain image locations. **Table 3.5** displays the average NPCR/UCAI value.

Table 3.5: NPCR and UACI values for each image

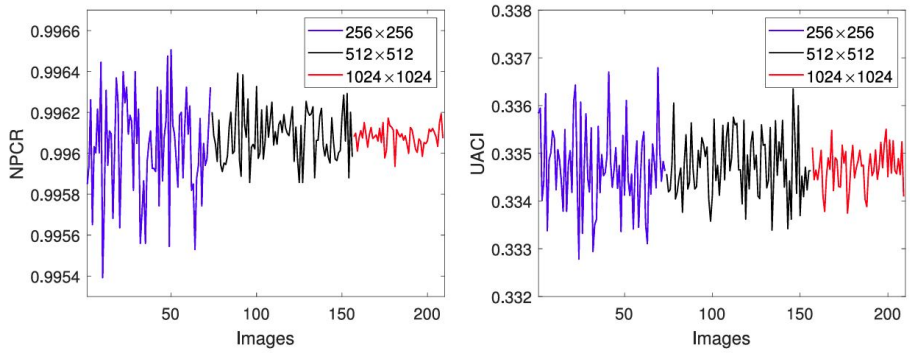
Metric	**Image**	**Value (%)**
NPCR	Peppers	99.61
NPCR	Mandrill	99.60
NPCR	Lena	99.61
NPCR	Cameraman	99.60
UACI	Peppers	33.50
UACI	Mandrill	33.51
UACI	Lena	33.49
UACI	Cameraman	33.44

Figure 3.6 displays the numbers for each iteration. The schemes in [48] are not as good as our results in Table Comparison for Lena gray scale image. Additionally, we perform the NPCR/UACI test on every plain image found in the USC-SIPI database. **Figure 3.7** show the plotted findings on paper.



(a) (b)

Figure 3.6: Figures (a) and (b) show the NPCR and UACI metrics, which were calculated over 50 runs for various images



(a) (b)

Figure 3.7: (a)NPCR evaluation across images of varying sizes (b) UACI evaluation across images of varying sizes

3.6.2 Cryptoanalysis Using Statistics

An encryption system can be used for encryption in real time if it passes widely recognized tests like correlation, histogram, and entropy. Below is a detailed discussion of each test and the related findings.

Test of Entropy

The most crucial metric for measuring the degree of unpredictability in a dataset of images is entropy. It illustrates the unpredictability of pixel intensity in an image dataset. If I were a

grayscale image, the entropy S equation would be

$$S = -\sum_{i=0}^{255} P(x_i)\log_2(P(x_i)) \tag{3.9}$$

High entropy indicates high randomness in image data. Following **Table 3.6** provides a list of multiple images' entropy. When compared to methods in [48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58], entropy is better than the entropy of lena encrypted image shown in **Table 3.7**. Additionally, we evaluate the suggested technique on every plain image in the USC-SIPI collection. The entropy ranges from $7.995 \leq H(I) \leq 7.9999$. The results are shown in **Figure 3.8**. As a result, the suggested technique generates a lot of randomness.

Table 3.6: Image Entropy Analysis

Plain-image	**Peppers**	**Mandrill**	**Lena**	**Cameraman**
Plain image Entropy	7.5571	7.2641	7.4204	7.1048
Cipher image Entropy	7.9976	7.9971	7.9975	7.9972

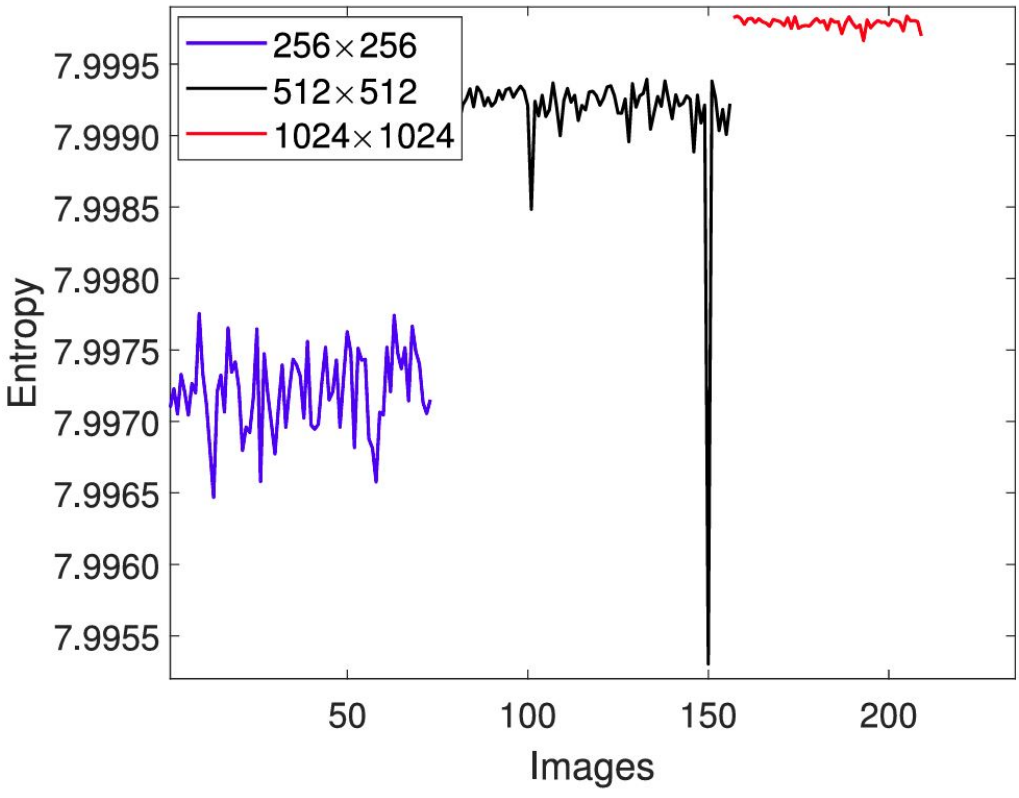


Figure 3.8: The entropy distribution of images with varying sizes

Table 3.7: Lena Grayscale Image comparison

Algorithm	**Entropy**	**UACI**	**Corr-H**	**Corr-D**	**Corr-V**	**NPCR**
presented	7.9975	33.670	0.0030	0.0096	0.0026	99.610
[59]	7.9975	33.472	-0.0018	-0.0009	0.0011	99.614
[49]	7.9965	33.392	0.0029	-0.0003	0.0080	99.617
[50]	7.9972	33.423	0.0069	0.0075	0.0479	99.625
[60]	7.9977	33.413	0.0003	-0.0003	-0.0000	99.621
[48]	7.9974	33.463	0.0004	0.0051	0.0051	99.606
[61]	7.9976	33.451	-0.0018	0.0040	-0.0006	99.609
[51]	7.9967	34.080	-0.0003	-0.0066	-0.0013	99.580
[52]	7.9970	33.505	0.0119	0.0011	0.0092	99.594
[53]	7.9971	33.456	-0.0029	0.0004	-0.0017	99.599
[54]	7.9970	33.419	0.0086	0.0009	0.0024	99.605
[55]	7.9962	33.384	0.0015	0.0057	0.0041	99.633
[56]	7.9970	33.546	-0.0016	-0.0026	0.0043	99.614
[57]	7.9973	33.458	-0.0023	-0.0029	0.0016	99.626
[58]	7.9975	33.457	-0.0034	-0.0063	0.0013	99.621
[62]	7.9966	33.452	-0.0008	-0.0101	0.0014	99.617
[63]	7.9974	33.356	0.0102	0.0052	0.0067	99.580

Histogram Analysis

If the histograms of the ciphered images are consistent, the cipher method becomes secure. **Figure 3.9 (a)-(d)** show histograms of the plain images in **Figure 3.10 (a)-(d)**, while **Figure 3.10 (e)-(h)** show the histograms of their encrypted images. The suggested encryption scheme's security is demonstrated by the consistent distribution of encrypted image histograms.

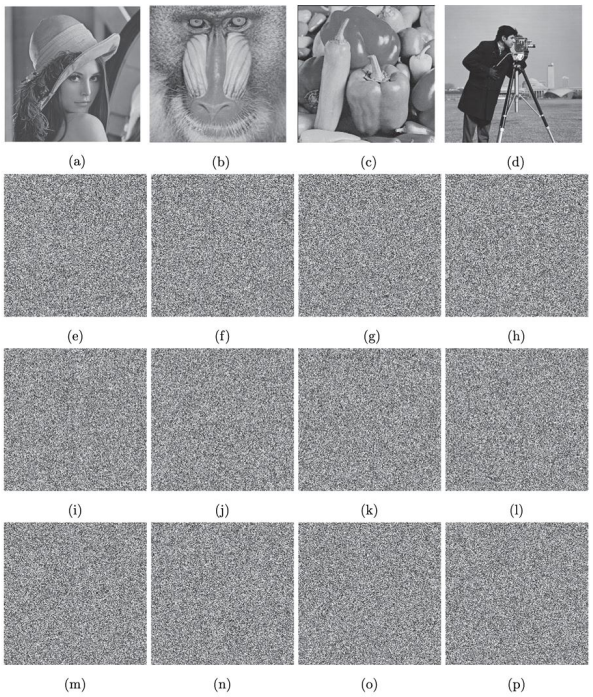


Figure 3.9: (a) Lena with $(m_x, m_y) = (19912, 40885)$; (b) Mandrill with $(m_x, m_y) = (39600, 54056)$; (c) Peppers with $(m_y, m_x) = (49494, 30600)$; (d) A cameraman with $(m_x, m_y) = (36084, 49952)$ while (e–h) Images of (a)–(d) that have been ciphered, where P is the image that is all white, (i)–(l) The ciphered images (a)–(d) whereas P is the image of lena; ciphered images (m)–(p), where P is the image of cameraman

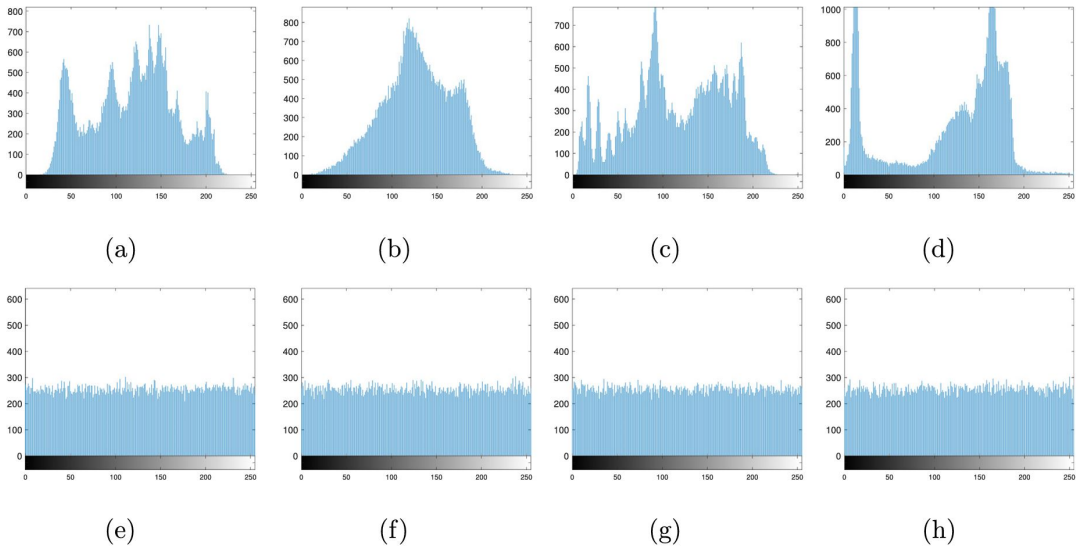


Figure 3.10: **Figure 3.8 (a-d)** displays the plain image histogram, while **Figure 3.8 (e-h)** displays the ciphertext histogram.

Correlation Test

Pixel correlations are typically high in plain images. Creating encrypted images with minimal correlation is essential for a perfect cryptosystem. Pixels' correlation coefficient is computed as

$$\sigma_{xy} = \frac{\sum_{i=1}^{M}(x_i - F(x))(y_i - F(y))}{\sqrt{\sum_{i=1}^{M}(x_i - F(x))^2 \sum_{i=1}^{M}(y_i - F(y))^2}} \tag{3.10}$$

Where

$$E(x) = \frac{1}{M}\sum_{i=1}^{M} x_i. \tag{3.11}$$

The data size in this case is M. F is the expected value opertaor and σ is the correlation coefficien. **Table 3.8** shows the correlation between various images. The results of **Table 3.7** is also comparable to this schemes. The correlation between each plain image within the USC-SIPI database is shown in **Figure 3.11**. The results shown in **Figure 3.11** and **Figure 3.12** show that the proposed approach passes the correlation test.

3.6.3 Key Analysis

Key space

If a cryptosystem's key space is larger than 2128, it passes the key space analysis. Our suggested cryptosystem uses SHA-256 and the 256-bit parameters a, p, b, G, b', P' as keys. As a

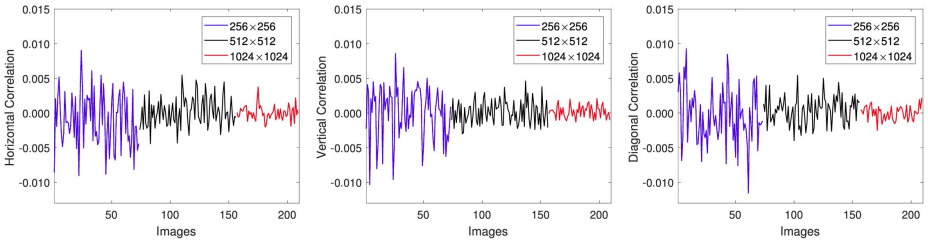


Figure 3.11: using the suggested encryption method, the horizontal vertical and diagonal relationship between two adjacent pixels were examined for images with sizes of 256 × 256, 512 × 512, and 1024 × 1024

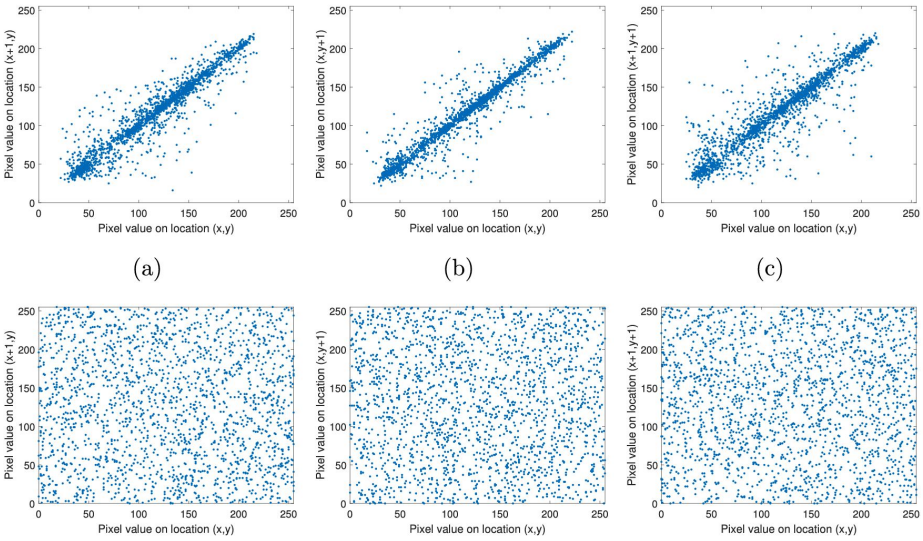


Figure 3.12: The pixel adjacency patterns of $Lena_{256\times256}$ are depicted in images (a)-(c) and its encrypted version (d)-(f) as shown in Fig. 3.8(b).

Table 3.8: Correlation Analysis for Original and Encrypted Images

Image	**Original Image**			**Encrypted Image**		
	Horizontal	**Vertical**	**Diagonal**	**Horizontal**	**Vertical**	**Diagonal**
Peppers	0.9429	0.9441	0.9025	−0.0042	0.0039	−0.0046
Cameraman	0.9331	0.9592	0.9075	0.0015	0.0023	0.0025
Lena	0.9390	0.9136	0.9680	0.0007	0.0022	0.0085
Mandrill	0.9048	0.8899	0.8199	0.0032	0.0042	0.0020

result, our crypto-system's key space is significantly bigger than 2128. Our encryption method is therefore impervious to brute-force attacks.

Sensitivity of keys

In order to obtain the real keys, opponents typically encrypt a plain image using multiple keys and compare it to the original encrypted image. A cryptosystem needs to be extremely sensitive to every key in order to attain higher security. On the basis of $I's$ hash value, we construct CML parameters. The cryptosystem is especially sensitive to keys due to the sensitivity of the CML-system to the initial conditions.

Known-plaintext and chosen-plaintext attack

Following the use of a sequence of PRNs in order to create diffusion in the plain image, the SHA-256 algorithm has the goal of creating confusion in a diffused image. Because the pair of S-boxes varies depending on the type of plain image, the strategy is immune to plaintext attacks. Someone outside the system cannot use a chosen-plain text attack to crack the cryptosystem. In order to prevent the attacker from using a chosen-plain text attack to discover any information about the final substitution box $S(u,p,a,b)$, the sets Y and X are arbitrary generated over the set $\mathbb{Z}_{2^n}$. These sequences' output, $\alpha_{p',b',m_x,m_y,A}$, is highly reliant on m_x and m_y, which makes it challenging for the attacker to locate secret keys. Because of this, the technique is very resistant to chosen-plaintext attacks. **Table 3.9** demonstrates that the NPCR and UACI are extremely near to optimal values and **Figure 3.13** represents that histogram are uniform . Thus, the suggested system has good resistance against known-plain text and chosen-plain text attacks.

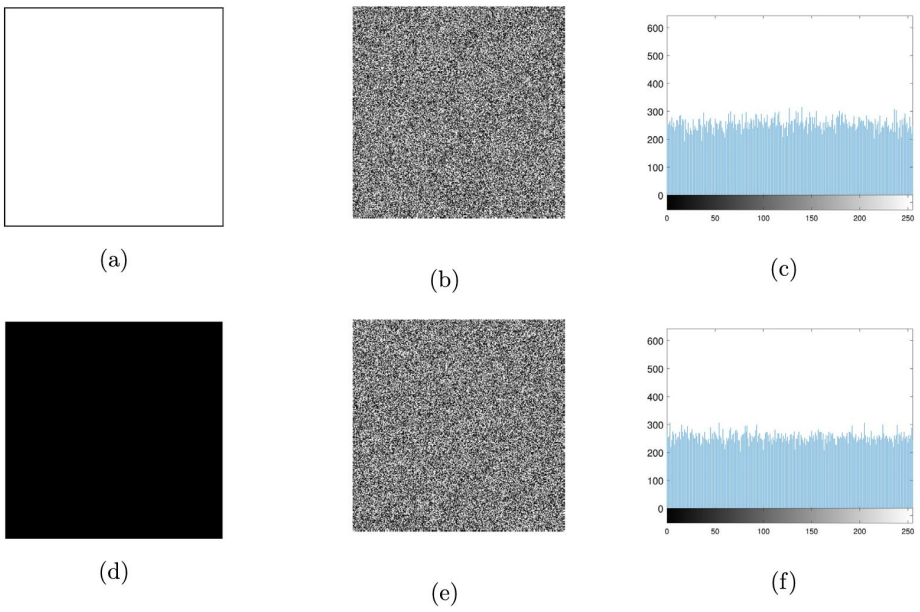


Figure 3.13: (a) All-white image $(m_x, m_y) = (24806, 33807)$; (b) encrypted-image; (c) Histograms; (d)A black pixel located at $(20022, 46968)$; (e) encrypted-image; (f) Histogram.

Table 3.9: Analysis of Security

Metric Category	**All-black**	**All-white**
NPCR	99.63	99.68
UACI	33.60	33.45
Correlation - Horizontal	-0.0056	0.0028
Correlation - Vertical	0.0076	0.0050
Correlation - Diagonal	0.0046	-0.0002
Entropy	7.9941	7.9946

3.6.4 Computation Analysis

We set the total lattice configuration count in the CML-structure to $\tau = 8$ for analysis. The suggested scheme has a lower time complexity. The calculation of the the computation time is given in **Table 3.10**.

Table 3.10: Analysis of Lena images running on the same OS in terms of run time in seconds

size of image	Proposed system	[50]	[14]	[64]	[65]	[66]	[67]	[68]
256×256	0.641	1.286	8.744	3.035	0.834	2.913	1.654	15.45
512×512	1.311	4.333	2731.450	3.082	2.849	11.26	6.353	76.777

Table 3.11 compares the run times of our approach with those of algorithms in [60, 48, 69, 70, 71, 72] across various operating systems. Our encryption method encrypts grayscale Lena image (256×256) and high resolution image Lena (512×512) in 0.641 and 1.311 seconds, respectively. The suggested encryption system is significantly faster than the techniques in [50, 64, 67, 66, 65, 68, 73], as **Table 3.10** illustrates. In **Table 3.10**, we compared Lena images with sizes of 256×256 and 512×512. In a similar vein, **Table 3.11** results make it evident that

Table 3.11: Analysis of the $\text{Lena}_{256 \times 256}$ image's run time across various operating systems using related schemes.

Algorithm	Time (s)	MATLAB Version	CPU	RAM
Proposed	0.6460	R2017a	Intel i5, 3.2 GHz	8 GB
[48]	1.8936	R2016b	2.4 GHz	12 GB
[60]	1.1247	R2016a	3.0 GHz	8 GB
[69]	1.4816	7.14	Intel i7, 3.4 GHz	16 GB
[70]	2.2234	R2016a	Intel i7, 2.7 GHz	8 GB
[71]	2.4600	R2017a	2.8 GHz	8 GB
[72]	4.6880	R2016b	2.8 GHz	8 GB

the newly created cryptosystem outperforms the algorithms in [48, 60, 69, 70, 71, 72]. We find that our recommended methodology performs better than current methods and can be used for encryption of images in real time.

3.7 Discussion

A safe cryptosystem for encrypting digital images is suggested in this study. A suggested approach is predicated on a CML system and EC's. To make the plain image diffuse, a collection of key-dependent PRN's is built. To create image-sensitive dynamically substitution boxes from an elliptic curve and a couple map lattice, an effective S-box generator is created. The purpose

of the suggested S-boxes is to confuse the basic image. After that benefits of the S-box generator and encryption technique are demonstrated by experimental results:

- The suggested generator is more efficient than the generators in [14, 32, 33, 34], according to **Table 3.3**.
- The typical Lena image's NPCR and UACI values in **Table 3.7**.
- According to theory, the suggested cryptosystem is less complex in terms of time than the schemes in and requires less running time than the algorithms.
- The suggested approach is effective for EC's with big parameters.

Additionally, the suggested approach offers extra security features to defend against attacks using attacks with prior knowledge (KPA) and encryption oracle attacks (CPA) by leveraging PRNs' key-stream and key-dependent dynamic S-boxes. Also, the security against brute-force attacks is guaranteed by the large key space. According to the suggested article, we hope to use the cipher technique for upcoming visual data, even though we employ it for grayscale images in this article.

CHAPTER 4

AN EFFICIENT APPROACH TO DESIGN A NEW ASYMMETRIC KEY ENCRYPTION ALGORITHM USING ELLIPTIC CURVES

4.1 Overview

This chapter presents an asymmetric key encryption scheme using elliptic curves. It generates pseudo random numbers over finite fields, non-linear mixing, and modular exponentiation for greater entropy. Diffie-Hellman key exchange is generated using elliptic curves. A secure ECC-based pseudo random number generator is used in a third algorithm. The last algorithm uses ECC pseudo random number generator to encrypt images, featuring secure pixel permutation and bitwise XOR. This hybrid cryptosystem combines safe key exchange with elliptic curves and symmetric encryption.

4.2 Elliptic Curve PRNG Generator

- **Purpose of Algorithm:**

  To generate pseudo-random numbers from elliptic Curves over a finite field $\mathbb{F}_p$, followed by nonlinear mixing and modular exponentiation for greater entropy.

- **Parameter initialization**

  In our first algorithm p represents a prime number of a finite field, m_x, m_y, m_z represent the output moduli for the x, y, z axes, and N represents the number of pseudo-random points to draw.

- **Describe the Elliptic Curve:**

  we choose an elliptic curve is $y^2 = x^3 + ax + b \ mod \ p$ where $a, b \in \mathbb{F}_p$.

- **Assign Start Point:**

  Saves all the valid (x,y) points of the curve into $\mathbb{F}_p$

- **Establish ECC Points:**

  Repeat for every x in $\mathbb{F}_p$. Find R.H.S of

$$y^2 \equiv x^3 + ax + b \pmod{p}, \quad a, b \in \mathbb{F}_p$$

- **Ensure Enough Points:**

  Repeat list P with exactly N entries.

- **Create Random Third Coordinate:**

  In our algorithm, we extend the elliptic curve points to 3D by adding a random w coordinate. We increase entropy (randomness) by introducing a random w value, which makes it more difficult to reverse the output. Now we will discuss how the code operates. Firstly, we generate (x, y) points on the curve, then for every point (x, y, w), define a random w, then utilizing all three coordinates, we calculate the PRNG output:

  For X axis $(x + y + w)^4 \ mod \ m_x$

  For Y axis $(x.y + w)^5 \ mod \ m_y$

  For Z axis $(x \cdot y + w)^8 \ mod \ m_w$

- **PRNG Base Loop:**

  For each point:

  Combine x,y,w with nonlinear exponentiation.

  Employ modular operations to maintain finite limits.

  Outputs: X_i, Y_i, Z_i form a 3D pseudo-random numbers.

4.2.1 Output of ECC-based 3D PRNG Generator

A list of pseudo-random $3D$ points (X, Y, Z) calculated using elliptic curve mathematics, modular exponentiation, and non-linear operations is the algorithm's output. We present these inside a three-dimensional scatter plot to demonstrate their dispersion and randomness. With the addition of an arbitrary third component w, the coordinates are derived from elliptic curve

Algorithm 1 ECC-Based PRNG

Input: $p \geq m \times n$, m_x,m_y, $m_w \in [2, M]$, $N \in \mathbb{Z}^+$, Choose $EC \in (x^3 + ax + b) \bmod p$
Output: 3D points $(X_i, Y_i, Z_i)_{i=1}^{N}$
procedure GENERATE PRNG(p, m_x, m_y, m_w, N)
    $EC \leftarrow$ Generate-ECC-Points(p)
    Extend EC to N points with random w values
    for (x, y, w) in EC_{3D} **do**
        $X_i \leftarrow (x + y + w)^4 \mod mx$
        $Y_i \leftarrow (x.y + w)^5 \mod my$
        $Z_i \leftarrow (xy + w)^8 \mod mw$
    end for
    return $(X_i, Y_i, Z_i)_{i=1}^{N}$
end procedure

points that are valid across a finite field. A number of modular operations and entropy increasing transformations, including exponentiation, are used to create each output triple (X_i, Y_i, Z_i). Extreme randomness and suitability for simulation or cryptographic applications are shown by the final scatter plot's enormous dispersion and lack of observable patterns. **Figure 4.1** shows the distribution of ECC-based PRNG output graphically.

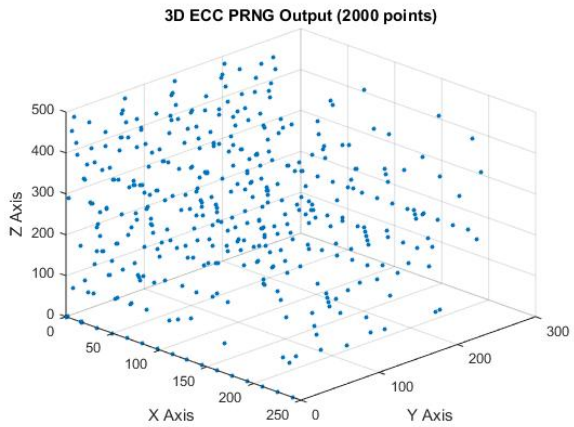


Figure 4.1: ECC-based 3D PRNG Generator

4.3 ECDH Algorithm

Alice and Bob can safely calculate a shared secret over an unprotected channel using elliptic curve cryptography by using the following algorithm, which realizes the Elliptic Curve Diffie-

Hellman key exchange protocol.

- **Parameter Initialization:**

  In this algorithm, the elliptic curve is defined on the field of prime number p, $\mathbb{F}_p$. a is a curve coefficient and b is implicit on the elliptic curve $y^2 = x^3 + ax + b \ mod \ p$. We create Base Points $G = (x_G, y_G) \in E(\mathbb{F}_p)$ on the curve that is utilized for key exchange.

- **Private key generation:**

  A private key is chosen by Alice $n_a \in [1, p-1]$.

  A private key is chosen by Bob $n_b \in [1, p-1]$

- **Function SM (scalar multiplication):**

  Applies the double-and-add procedure k times to add point U to itself. The purpose is that caculate $k.U$ on elliptic curve efficiently.

- **Elliptic curve function point addition:**

  On the elliptic curve calculate the sum of two points $U + V$. Manage many cases like

  Returns the other point if U or V is the identity point $(0,0)$.

  Using point doubling formula if $U = V$

  If $U \neq V$, apply the elliptic curve addition formula.

  Result is identity point if $U = -V$

- **Elliptic curve function (Modular inverse):**

  The modular inverse of a mod p is computed using the Extended Euclidean algorithm.

  Verifies the validity of divisions in point addition and doubling in $\mathbb{F}_p$.

- **Calculation of Public key:**

  Alice uses the following to create her public key: $P_A = n_a.G$

  Bob uses the following to create her public key: $P_B = n_b.G$.

- **Shared secret:**

  We compute shared secret by using $Shared \ secret = n_a.P_B = n_b.P_A$.

  So Alice obtain her public key by $S_A = n_a.P_B$ and Bob obtain his public key by $S_B = n_b.P_A$.

  The shared secret matches if $S_A = S_B$ and scalar multiplication is commutative.

Algorithm 2 ECDH Key Exchange (Compact)

Input: $p \in \mathbb{P},\ a,b \in [1,p-1],\ G \in EC(Fp)$
$n_A \leftarrow \text{Rand}(1,p-1),\ n_B \leftarrow \text{Rand}(1,p-1)$
Output: Generation of ECDH Keys
function SM(U,k)
    $W \leftarrow (0,0),\ V \leftarrow U$
    while $k > 0$ **do**
        if $k \bmod 2 = 1$ **then**
            $W \leftarrow \text{ADD}(W,V)$
        end if
        $V \leftarrow \text{ADD}(V,V),\ k \leftarrow \lfloor k/2 \rfloor$
    end while
    return W
end function

4.3.1 Results of ECDH Algorithm

ECDH algorithm is used to show the generation and agreement of public keys and shared secrets between two parties. For example, Alice and Bob randomly choose their private keys, Alice's public key is (3,91) and Bob's public key is (80,10). The shared secret between them is then matched, confirming the proper operation of the ECDH protocol. The shared secret generated by Alice and Bob will be identical, and the output will display a return message confirming the successful and secure key exchange.

4.4 ECC-Based Secure PRNG

The third pseudocode presents a complete and secure ECC-based PRNG system that uses elliptic curve cryptography along with Diffie-Hellman key exchange and hashing. The outcome is a high-entropy PRNG output appropriate for cryptographic usage and the additional benefit of ECC security and precomputation methods for performance enhancements.Explanation is given below:

- **Parameter Initialization:**

  First, we define an elliptic curve, $y^2 = x^3 + ax + b\ (mod\ p)$, over a finite field $\mathbb{F}_p$. and other parameters are constant where $p \in \mathbb{P}$ and $a,b \in [1,p-1]$ are curve parameters and Base point is $G = (x_G, y_G)$.

- **Hide generator point:**

  To create a hidden generator, multiply the base point G by a secret random scalar r: $G' = [r]G$. This improves randomization and privacy while also hiding the original generator.

- **Precompute the powers of Hidden generator:**

  Precompute $G'_i = 2^{i-1}G'$ for i=1 to i=32 which allows for scalar multiplication up to 32 bits. This increases scalar multiplication efficiency.

- **Create Pairs of ECC Keys:**

  Two private keys, n_A and n_B, should be chosen at random for Alice and Bob: $n_A, n_B \in_R [2, p-1]$.

  We calculate the public keys, $P_A = [n_A]G'$ and $P_B = [n_B]G'$, using private keys.

- **Implement the DH Key Exchange for ECC:**

  Both parties compute shared secret $S = n_A.P_B = n_B.P_A$. If both parties do the same calculation, then the key exchange will be successful.

- **The shared ECC point is hashed:**

  Combine the x and y coordinates and shared point S.

  To obtain a fixed-length hexadecimal string, use SHA-256:

  K = SHA-256 $(x_S||y_S)$

  This acts as the PRNG output generation's seed or key.

- **Pseudo-Random Point Generation:**

  For every $i = 1$ to N (for example, $N = 500$):

  Using SHA-256 Hash $K||i$. Three hash portions should be collected and reduced modulo maximum values such as:

  $u = H_i^1 mod\ m_u$, $v = H_i^2 mod\ m_v$, $w = H_i^3 mod\ m_w$.

  Store each $3D$ point (u, v, w).

- **Return the final PRNG output back:**

  A list of N $3D$ pseudo-random points, securely formed from hash functions and ECC procedures, is the algorithm's output.

Algorithm 3 ECC-Based Secure PRNG

Input: Set curve: $y^2 = x^3 + ax + b \mod p$
$G \leftarrow (x_G, y_G),\ p \in \mathbb{P},\ a, b \in [1, p-1]$
$G' \leftarrow [r]G$ with random r
Precompute $G'_i = 2^{i-1}G'$ for $i = 1$ to 32
Generate private keys: $n_A, n_B \in_R [2, 2^{32} - 1]$
$P_A \leftarrow [n_A]G'$, $P_B \leftarrow [n_B]G'$
Derived secret: $S \leftarrow [n_A]P_B = [n_B]P_A$
$K \leftarrow$ SHA-256 of $x_S \| y_S$
for $i = 1$ to N **do**
    $H_i \leftarrow$ SHA-256$(K \| i)$
    $(u, v, w) \leftarrow (\mathrm{mod}(H_i^{(1)}, m_u), \mathrm{mod}(H_i^{(2)}, m_v), \mathrm{mod}(H_i^{(3)}, m_w))$
    Store (u, v, w)
end for
return PRNG point list

4.4.1 Output of ECC-Based Secure PRNG

Using a custom curve parameter and a secret generator point, this Algorithm performs elliptic curve Diffie-Hellman key exchange, computes a common ECC point between two parties, hashes the common secret using SHA-256, and then uses the hash as a seed to create 500 pseudo-random 3D points, with each value being calculated by hashing the seed with an integer counter and projecting portions of the hash to U, V, and W coordinates in specified ranges. The output is a 500×3 matrix of cryptographically secure pseudo-random coordinates, which is also visualized in **Figure 4.2**.

ECC PRNG with Image Encryption (XOR + Permutation)

This algorithm uses a pseudo-random number generator based on ECC to encrypt images. Secure pixel permutation and bitwise XOR are features of the encryption. This method is a hybrid cryptosystem that combines safe key exchange with Elliptic Curve Cryptography (ECC) and symmetric encryption with a pseudo-random number generator (PRNG). It uses permutation-based confusion and XOR-based diffusion to encrypt images, guaranteeing confidentiality and defense against statistical assaults. A thorough explanation of each element is provided below.

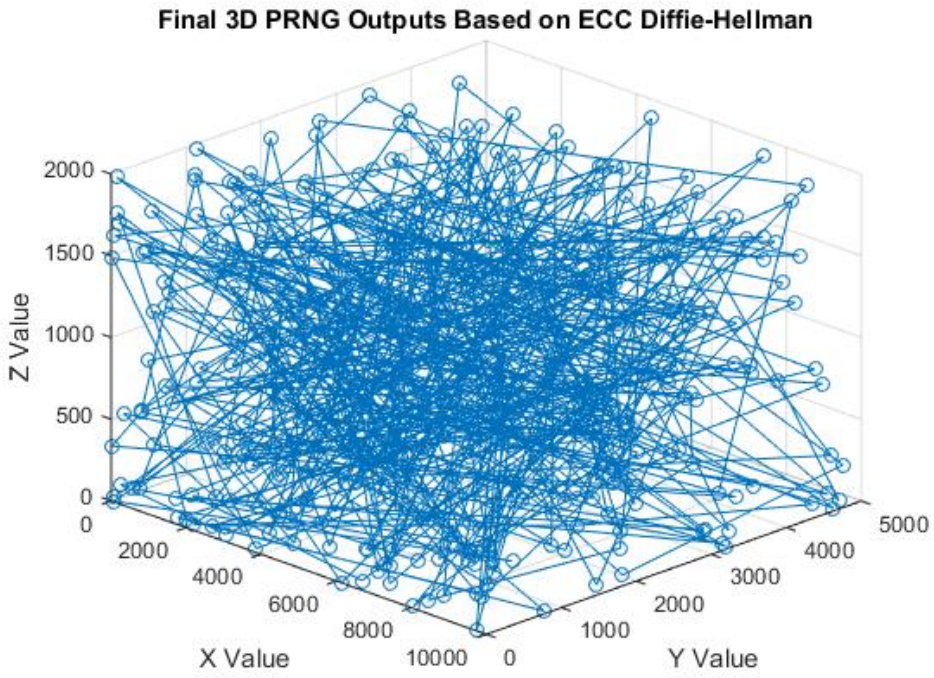


Figure 4.2: Visualization of ECC-Based Secure PRNG

Step by Step Explaination

- **Parameter initialization:**

  First, we define an elliptic curve, $y^2 = x^3 + ax + b \; mod \; p$, over a finite field $\mathbb{F}_p$. G is the base point on the curve. For security or privacy purposes, the generator point G may be hidden using the Hide-Generator custom function.

- **Compute the G Table in advance:**

  In order to faster ECC point multiplication, this phase precomputes multiples of the (hidden) generator, usually through windowing techniques. For scalar multiplication $k \cdot G$, this is helpful when k is a 32-bit value. The generating point G is concealed through a transformation (may be randomization or point compression) to improve security.

- **Generation of ECC Diffie-Hellman Keys:**

  Bob chooses private key n_b, while Alice chooses private key n_a. Thus, Alice computes the public key $P_a = n_a.G'$, and Bob uses scalar multiplication to compute his public key $P_b = n_b.G'$. Then Alice calculates $ABG' = n_a.P_b$ and Bob calculate $BAG' = n_b.P_a$, where ABG' represents that A is the sender (Alice) and B represents that B is the reciever (Bob) and G is the generator point or base point of Elliptic curve But we use $G' = s(secret \; scalar) \times G$

which is a modified generator for security and randomness.

- **Derivation of a Shared Secret:**

  A shared ECC secret is calculated by each party:

  $ABG' = n_a.P_B \quad and \quad BAG' = n_b.P_A$

  Both get to the same conclusion because of ECC properties:

$$ABG' = n_a.(n_B.G') = P_b.(P_a.G') = BAG' \tag{4.1}$$

- **Agree the derived secret:**

  Checking the derived secret of both parties. The commutative nature of scalar multiplication on elliptic curves ensures this that Shared secret match. if shared secret fails then some possible reason are MATLAB precision problems or symbolic mismatches, errors in scalar multiplication or ECC addition function or error in computing by using different G' between the two key calculations. To securely agree on a shared key, asymmetric ECC is utilized.

- **Generation of Symmetric key for Decryption and Encryption:**

  In this algorithm we use asymmetric key cryptography, which involves generating symmetric keys using a hash. The symmetric key is then used to encrypt and decrypt images. The shared secret, ABG', is obtained by multiplying Party A's private key and Party B's public key, resulting in a point on the elliptic curve. The coordinates are then hashed into a string, and a safe, fixed-length key is generated using SHA-256. This key, which appears random, can be computed independently by Alice and Bob, who share the same ECC point.

- **Set the PRNG Key:**

  The process involves creating a stream of pseudo-random numbers using a symmetric key as a seed, which is then used to XOR image pixels. The randomness is generated using the hashed ECC shared secret. The XOR technique makes each pixel unreadable by scrambling its data and can be reversed with the same random number. The shared key, created by Alice and Bob using ECC as a seed, generates the random numbers. This results in an encrypted image with chaotic values, making it impossible to guess the original.

- **preparation of image:**

  Firstly we load Gray scale image . Make sure the image is 256×256 pixels in size before

loading it. Create a $1D$ vector by flattening the $2D$ image:

$N = 256 \times 256 = 65536$

- **Generation of PRNG Streams:**

  The PRNG is seeded by the key, which is derived by ECC. $[N/3]$ triplets are produced by the method (probably for RGB, but the image is grayscale). To match the picture vector length (N), the PRNG output is trimmed or reshaped. Create enough PRNG output (maybe dependent on ECC output chunks) to fill the image. Reshape or truncate to fit the image vector's dimensions. For the security of PRNG, PRNG needs to be secure using cryptography (e.g., ChaCha20, AES-CTR). Decryption is made possible by reusing the same key, which will replicate the same keystream.

- **Encryption Process:**

  The two main steps of encryption are XOR encryption, a symmetric method comparing each pixel to its associated PRNG byte, and encryption via permutation, a permutation index array created using the key-derived seed and a deterministic permutation. The XOR-encrypted image vector is shuffled to increase confusion.

- **Resize the Encrypted Image:**

  Return the encrypted vector to a two-dimensional image.

- **Decryption Process:**

  The decryption process reverses encryption processes to restore the original image. It has two primary functions: permutation in reverse and XOR decryption. Perm is a permutation vector used to mix pixels during encryption, using the same RNG seed. The precise permutation order is retrieved using perm (N) and the inverse permutation $perm^{-1}$. The encrypted image is rearranged using $perm^{-1}$, and the rearranged pixel values are stored in the working variable depermuted. XOR decryption involves generating a PRNG stream using the same key as encryption and XORing each element in depermuted. The original grayscale intensities of the pixel values are restored.

- **Confirmation:**

  The algorithm determines whether the original image and the decrypted image are identical: The comparison is done pixel by pixel. If every pixel is the same result in the form of success . If there are discrepancies result shows Fail indicating a key/PRNG/permutation mistake.

Algorithm 4 ECC PRNG Image Encryption System

Input: Prime p, curve params a, b, base point G, image I

Output: Encrypted image C, decrypted image I'

Step 1: System Initialization

$E \leftarrow \text{EllipticCurve}(y^2 = x^3 + ax + b \mod p)$

$G \leftarrow (x_G, y_G) \in E$

Step 2: Generator Obfuscation

$k \leftarrow \text{RandomInteger}(2, p-1)$

$G' \leftarrow \text{EC\_ScalarMult}(G, k, a, p)$

Step 3: Key Exchange (ECDH)

$a_{priv} \leftarrow \text{RandomInteger}(2, p-1)$

$b_{priv} \leftarrow \text{RandomInteger}(2, p-1)$

$A \leftarrow \text{EC\_ScalarMult}(G', a_{priv}, a, p)$

$B \leftarrow \text{EC\_ScalarMult}(G', b_{priv}, a, p)$

$S \leftarrow \text{EC\_ScalarMult}(B, a_{priv}, a, p)$

Verify $S = \text{EC\_ScalarMult}(A, b_{priv}, a, p)$

Step 4: Key Derivation

$K \leftarrow \text{SHA-256}(x_S \parallel y_S)$

Step 5: PRNG Generation

$\text{PRNG\_stream} \leftarrow \text{Generate\_PRNG}(K, \text{len}(I))$

for $i \leftarrow 1$ to $\text{len}(I)$ **do**

    $\text{seed}_i \leftarrow \text{SHA-256}(K \parallel i)$

    $\text{PRNG}_i \leftarrow \text{seed}_i[0:24] \mod 256$

end for

Step 6: Image Encryption/Decryption

$C \leftarrow I \oplus \text{PRNG\_stream}$

$I' \leftarrow C \oplus \text{PRNG\_stream}$

return C, I'

4.4.2 Results of this Algorithm

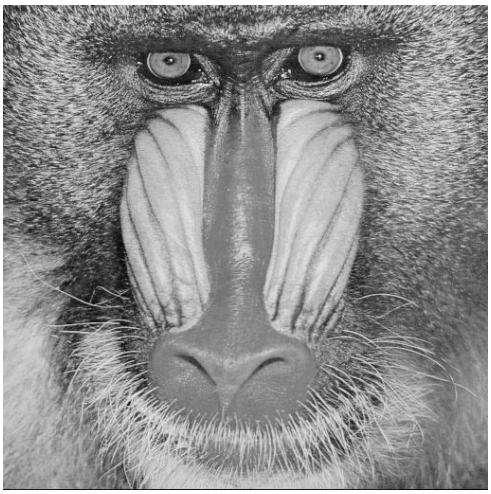
The resulting images proves the security of the ECC-based PRNG encryption system. In the figure **Figure 4.3 (a)-(c)** which are grayscale images of lena , cameramen and mandrill with the size of 256×256 . These are plain images which are used in encryption while the **Figure 4.3 (d)-(f)** are cipher uses a result of the combined effect of encryption and pixel permutation, it looks like pure ciphered and then we apply decryption on figures and **Figure 4.3 (g)-(i)** are the image that has been decoded, which is visually the same as the plain image, proving that the decryption stage correctly inverted both operations. This confirms the correctness of the system and its high potential for safe image encryption based on elliptic curve cryptography.

4.5 Mathematical example of ECC PRNG Encryption Decryption to verify Algorithm

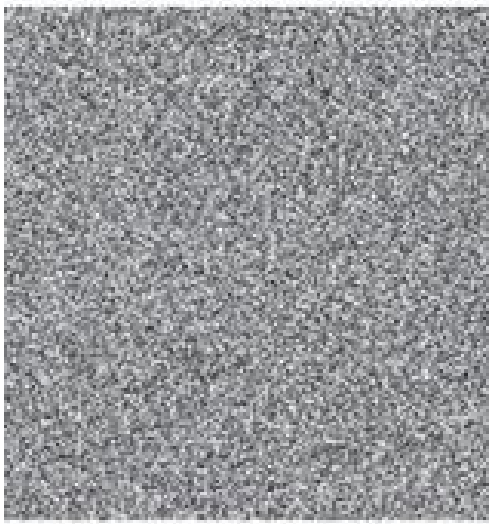
We take a ECC example over small field

- we select Prime field where $p = 17$
- Choose Elliptic curve: $y^2 \equiv x^3 + 2x + 2 \mod 17$
- Generator point: $G = (5, 1)$
- G(ord)=19
- The private key of Alice: $n_A = 3$
- The private key of Bob: $n_B = 7$

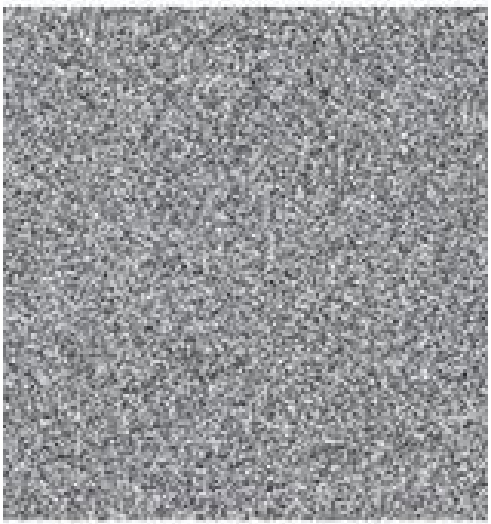
(a)



(b)

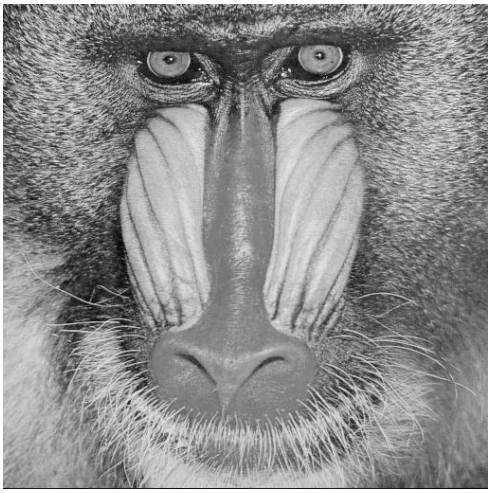


(c)



(d)

(e)



(f)

Figure 4.3: Plain images are shown in (a) and (b); ciphered images are shown in (c) and (d); and decrypted images are shown in (e), (f)

4.5.1 Step 1: Calculate the Public Key for Alice: $P_A = n_A \cdot G = 3 \cdot G$

Calculate $2G = G + G$ (point doubling):

$$\begin{aligned}
\lambda &= \frac{3x_1^2 + a}{2y_1} \mod p \\
&= \frac{3 \cdot 5^2 + 2}{2 \cdot 1} \\
&= \frac{75+2}{2} = \frac{77}{2} \mod 17 \\
77 &\equiv 9 \mod 17 \\
\frac{9}{2} &\equiv 9 \cdot 9 = 81 \equiv 13 \mod 17 \\
x_3 &= \lambda^2 - 2x_1 = 13^2 - 10 = 169 - 10 = 159 \equiv 6 \mod 17 \\
y_3 &= \lambda(x_1 - x_3) - y_1 = 13(5-6) - 1 = -13 - 1 = -14 \equiv 3 \mod 17 \\
2G &= (6,3)
\end{aligned}$$

Calculate $3G = 2G + G = (6,3) + (5,1)$ (point addition):

$$\begin{aligned}
\lambda &= \frac{y_2 - y_1}{x_2 - x_1} = \frac{1-3}{5-6} = \frac{-2}{-1} = 2 \mod 17 \\
x_3 &= \lambda^2 - x_1 - x_2 = 4 - 6 - 5 = -7 \equiv 10 \mod 17 \\
y_3 &= \lambda(x_1 - x_3) - y_1 = 2(6-10) - 3 = -8 - 3 = -11 \equiv 6 \mod 17 \\
3G &= (10,6) = P_A
\end{aligned}$$

4.5.2 Step 2: Calculate the Public Key for Alice: $P_B = n_B \cdot G = 7 \cdot G$

Using double-and-add method ($7 = 4 + 2 + 1$): Calculate $2G = (6,3)$ (from above). Calculate $4G = 2G + 2G = (6,3) + (6,3)$:

$$\begin{aligned}
\lambda &= \frac{3 \cdot 6^2 + 2}{2 \cdot 3} = \frac{110}{6} \mod 17 \\
110 &\equiv 8 \mod 17 \\
\frac{8}{6} &\equiv 8 \cdot 3 = 24 \equiv 7 \mod 17 \\
x_3 &= 49 - 12 = 37 \equiv 3 \mod 17 \\
y_3 &= 7(6-3) - 3 = 21 - 3 = 18 \equiv 1 \mod 17 \\
4G &= (3,1)
\end{aligned}$$

Calculate $6G = 4G + 2G = (3,1) + (6,3)$:

$$\lambda = \frac{3-1}{6-3} = \frac{2}{3} \equiv 12 \mod 17$$
$$x_3 = 144 - 3 - 6 = 135 \equiv 16 \mod 17$$
$$y_3 = 12(3-16) - 1 = -156 - 1 = -157 \equiv 13 \mod 17$$
$$6G = (16,13)$$

Calculate $7G = 6G + G = (16,13) + (5,1)$:

$$\lambda = \frac{1-13}{5-16} = \frac{-12}{-11} \equiv \frac{12}{11} \equiv 15 \mod 17$$
$$x_3 = 225 - 16 - 5 = 204 \equiv 0 \mod 17$$
$$y_3 = 15(16-0) - 13 = 240 - 13 = 227 \equiv 6 \mod 17$$
$$7G = (0,6) = P_B$$

4.5.3 Step 3: Calculate Shared Secret $S_B = n_B \cdot P_A = 7 \cdot (10,6)$

Following comparable computations (neglected for conciseness), we discover:

$$S_B = (6,3)$$

4.5.4 Conclusion of ECDH

The two sides come to the same mutually held secret:

$$S = (6,3)$$

So Alice Private key is $n_a = 3$

Alice Public key is $P_A = (10,6)$

Shared Secret is $S_A = (6,3)$

Bob Private key is $n_b = 7$

Bob Public key is $P_B = (0,6)$

Shared secret is $S_B = (6,3)$

4.5.5 Compute Shared Secret

$S_A = n_A \cdot P_B = 3 \cdot (0,6)$ Calculate $2 \cdot (0,6)$:

$$\lambda = \frac{3 \cdot 0^2 + 2}{2 \cdot 6} = \frac{2}{12} \equiv 3 \mod 17$$

$$x_3 = 9 - 0 = 9 \mod 17$$

$$y_3 = 3(0-9) - 6 = -27 - 6 = -33 \equiv 1 \mod 17$$

$$2 \cdot (0,6) = (9,1)$$

Compute $3 \cdot (0,6) = (0,6) + (9,1)$:

$$\lambda = \frac{1-6}{9-0} = \frac{-5}{9} \equiv 7 \mod 17$$

$$x_3 = 49 - 0 - 9 = 40 \equiv 6 \mod 17$$

$$y_3 = 7(0-6) - 6 = -42 - 6 = -48 \equiv 3 \mod 17$$

$$S_A = (6,3)$$

Bob's Shared Secret Calculation $S_B = 7 \cdot (10,6)$

Given: The private key of Bob: $n_b = 7$

The public key of Alice: $P_A = (10,6) = 3G$

Elliptic Curve: $y^2 = x^3 + 2x + 2 \mod 17$

Step 1: Point Doubling - Calculate $2 \cdot (10,6)$ By Using the point doubling formula:

$$\lambda = \frac{3x_1^2 + a}{2y_1} \mod 17$$

$$= \frac{3 \cdot 10^2 + 2}{2 \cdot 6}$$

$$= \frac{300+2}{12} = \frac{302}{12} \mod 17$$

Make the denominator and numerator simpler:

$$302 \mod 17 = 302 - 17 \times 17 = 13$$

$$12^{-1} \mod 17 = 10 \quad (\text{since } 12 \times 10 = 120 \equiv 1 \mod 17)$$

$$\lambda = 13 \times 10 = 130 \mod 17$$

$$= 130 - 7 \times 17 = 11 \mod 17$$

Calculate new point:

$$x_3 = \lambda^2 - 2x_1 = 11^2 - 20 = 121 - 20 = 101 \mod 17$$
$$= 101 - 5 \times 17 = 16$$
$$y_3 = \lambda(x_1 - x_3) - y_1 = 11(10-16) - 6 = -66 - 6 = -72 \mod 17$$
$$= -72 + 5 \times 17 = 13$$
$$2 \cdot (10,6) = (16,13)$$

Step 2: Point Doubling - Calculate $4 \cdot (10,6) = 2 \cdot (16,13)$

$$\lambda = \frac{3 \cdot 16^2 + 2}{2 \cdot 13} = \frac{768+2}{26} = \frac{770}{26} \mod 17$$
$$770 \mod 17 = 5$$
$$26 \mod 17 = 9$$
$$9^{-1} \mod 17 = 2$$
$$\lambda = 5 \times 2 = 10 \mod 17$$

Calculate new point:

$$x_3 = 10^2 - 2 \times 16 = 100 - 32 = 68 \mod 17 = 0$$
$$y_3 = 10(16-0) - 13 = 160 - 13 = 147 \mod 17$$
$$= 147 - 8 \times 17 = 11$$
$$4 \cdot (10,6) = (0,11)$$

Step 3: Point Addition - Compute $(0,11) + (16,13)$

$$\lambda = \frac{13-11}{16-0} = \frac{2}{16} = \frac{1}{8} \mod 17$$
$$8^{-1} \mod 17 = 15$$
$$\lambda = 1 \times 15 = 15 \mod 17$$

Calculate new point:

$$x_3 = 15^2 - 0 - 16 = 225 - 16 = 209 \mod 17$$
$$= 209 - 12 \times 17 = 5$$
$$y_3 = 15(0-5) - 11 = -75 - 11 = -86 \mod 17$$
$$= -86 + 6 \times 17 = 16$$
$$(0,11) + (16,13) = (5,16)$$

Step 4: Final Addition - Calculate $(5,16)+(10,6)$

$$\lambda = \frac{6-16}{10-5} = \frac{-10}{5} = -2 \equiv 15 \mod 17$$

Calculate new point:

$$\begin{aligned} x_3 &= 15^2 - 5 - 10 = 225 - 15 = 210 \mod 17 \\ &= 210 - 12 \times 17 = 6 \\ y_3 &= 15(5-6) - 16 = -15 - 16 = -31 \mod 17 \\ &= -31 + 2 \times 17 = 3 \end{aligned}$$

$$7 \cdot (10,6) = (6,3)$$

Conclusion of Bob's shared secret The Shared secret of Bob:

$$S_B = (6,3)$$

Hence Shared secret Matched.

4.5.6 Hash the Shared Secret

Now we hash the ECC point by converting shared Secret point "0623"

Use ASCII SUM

'0'=50

'6'=48

'0'=52

'3'=51

So Sum = 201. Now this is a PRNG seed

4.5.7 Generating PRNG Stream

- Now we generate PRNG stream by using this logic

$$PRNG[i] = mod\left((Hash - Seed \times i \times 17), 256\right) \tag{4.2}$$

Compute i=1 to 16

i	PRNG-stream	PRNG-Value
i=1	mod(201+17,256)	218
i=2	mod(402+17,256)	163
i=3	mod(603+17,256)	108
i=4	mod(804+17,256)	53
i=5	mod(1005+17,256)	254
i=6	mod(1206+17,256)	199
i=7	mod(1407+17,256)	144
i=8	mod(1608+17,256)	89
i=9	mod(1809+17,256)	34
i=10	mod(2010+17,256)	235
i=11	mod(2211+17,256)	180
i=12	mod(2412+17,256)	125
i=13	mod(2613+17,256)	70
i=14	mod(2814+17,256)	15
i=15	mod(3015+17,256)	216
i=16	mod(3216+17,256)	161

4.5.8 Encryption

$$Encrypted\ matrix_i = mod\ (Original\ matrix_i + PRNG, 256) \tag{4.3}$$

The above equation is used to encrypt our matrix

Let Original matrix = $\begin{bmatrix} 12 & 34 & 56 & 78 \\ 30 & 60 & 90 & 120 \\ 90 & 123 & 200 & 15 \\ 150 & 180 & 210 & 240 \end{bmatrix}$ We apply encryption on original matrix below

from $i = 1\ to\ 16$

i	Original-Value	PRNG	Encrypted-Value
i=1	12	218	230
i=2	30	163	193
i=3	90	108	198
i=4	150	53	203
i=5	34	254	32
i=6	60	199	3
i=7	123	144	11
i=8	180	89	13
i=9	56	34	90
i=10	90	235	69
i=11	200	180	124
i=12	210	125	79
i=13	78	70	148
i=14	120	15	135
i=15	15	216	231
i=16	240	161	145

So the encrypted Matrix = $\begin{bmatrix} 230 & 32 & 90 & 148 \\ 193 & 3 & 69 & 135 \\ 198 & 11 & 124 & 231 \\ 203 & 13 & 79 & 145 \end{bmatrix}$

- **Shuffling the Encrypted Matrix** Our shuffle version is given below:

  enc-shuffled = $\begin{bmatrix} e_3 & e_{12} & e_7 & e_1 \\ e_{15} & e_9 & e_6 & e_{14} \\ e_4 & e_2 & e_8 & e_{16} \\ e_5 & e_{13} & e_{11} & e_{10} \end{bmatrix}$

  Shuffled - Matrix= $\begin{bmatrix} 90 & 231 & 69 & 230 \\ 79 & 198 & 3 & 13 \\ 148 & 32 & 135 & 145 \\ 193 & 203 & 124 & 11 \end{bmatrix}$

- **Decryption Process of 4-by-4 Matrix:**

  For decryption firstly we reverse the permutation

$$\text{Shuffled[reverse-perm]}= \begin{bmatrix} 230 & 32 & 90 & 148 \\ 193 & 3 & 69 & 135 \\ 198 & 11 & 124 & 231 \\ 203 & 13 & 79 & 145 \end{bmatrix}$$

We done Decryption by using this

$$Decrypted = mod(Encrypted - PRNG, 256) \tag{4.4}$$

i	Encrypted-Value	PRNG	Decryption	Decrypted-Value
i=1	230	218	mod(12,256)	12
i=2	193	163	mod(30,256)	30
i=3	198	108	mod(90,256)	90
i=4	203	53	mod(150,256)	150
i=5	32	254	mod(-222,256)	34
i=6	3	199	mod(-196,256)	60
i=7	11	144	mod(-133,256)	123
i=8	13	89	mod(-76,256)	180
i=9	90	34	mod(56,256)	56
i=10	69	235	mod(-166,256)	90
i=11	124	180	mod(-56,256)	200
i=12	79	125	mod(-46,256)	210
i=13	148	70	mod(78,256)	78
i=14	135	15	mod(120,256)	120
i=15	231	216	mod(15,256)	15
i=16	145	161	mod(-16,256)	240

$$\text{Decrypted - Value = Original-Value} = \begin{bmatrix} 12 & 34 & 56 & 78 \\ 30 & 60 & 90 & 120 \\ 90 & 123 & 200 & 15 \\ 150 & 180 & 210 & 240 \end{bmatrix}$$

4.6 Differential Cryptanalysis

The UACI and NPCR techniques evaluate the sensitivity of encryption algorithms to differential attacks. If a minor alteration in plain images results in different cipher images, the algorithm

is resistant to differential attacks. The NPCR and UACI are calculated using equations (4.5) and (4.6).

$$NPCR = \frac{1}{N \times M} \sum_{i,j} D(i,j) \times 100 \tag{4.5}$$

$$UACI = \sum_{i,j} \frac{|I_C(i,j) - I'_C(i,j)|}{N \times M \times 255} \times 100 \tag{4.6}$$

whereever $D(i,j) = 1$ unless $C_I(i,j) - C'_I(i,j) eq 0$ and $D(i,j) = 0$. The average NPCR and UCAI value is shown in the following table.

4.6.1 The Image Encryption Calculation Process Using NPCR and UACI

NPCR and UACI are two crucial security metrics that are used to evaluate the strength of an image encryption system. These metrics evaluate the degree to which an encryption algorithm modifies the plain image in order protect against statistical attacks. Below is a detailed explanation of how these values were calculated and analyzed:

Number of Pixel Change Rate, or NPCR

Basic goal: NPCR measures the proportion of pixels that are different between the ciphered and plain images. A high NPCR (near 100%) means that the image is substantially changed by the encryption process, making it secure to differential attacks.
Now we describe the calculation steps below:

1. To make computations easier, the *RGB* input image is first converted to grayscale.

2. Determine whether the Plain (I) and ciphered (I') images differ for each pixel at position (i,j):

$$D(i,j) = \begin{cases} 1, & \text{if } I(i,j) \neq I'(i,j) \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

3. Determine how many different pixels there are in total:

$$\text{Total\_Diff} = \sum_{i=1}^{H} \sum_{j=1}^{W} D(i,j)$$

where height and width of an images are denoted by H and W.

4. The percentage of NPCR is now determined by dividing the total number of pixels $(H \times W)$ by the total number of differing pixels, and then multiplying the result by 100.

$$\text{NPCR} = \left(\frac{\text{Total\_Diff}}{H \times W} \right) \times 100$$

5. NPCR should be approximately 99.6094% for encryption to be effective, meaning that practically all pixels are changed.

UACI(Unified Average Changing Intensity)

Finding the average intensity difference between the original and encrypted images is the main objective of UACI. Uniform pixel changes are indicated by a value near 33.4635%, which guarantees resistance to statistical analysis.
Now we calculate the steps:

1. **Image Conversion to Double Precision:** For precise arithmetic, both the original (I) and encrypted (I') images are doubled.

2. **Determine Absolute Differences:** Determine the absolute intensity difference for each pixel:

$$\Delta(i, j) = \left| I(i, j) - I'(i, j) \right|$$

3. **Add up the differences:** Add up all of the absolute differences:

$$\text{Total\_Abs\_Diff} = \sum_{i=1}^{H} \sum_{j=1}^{W} \Delta(i, j)$$

4. **Improve for Maximum Intensity:** The sum of the difference is divided by the maximum pixel value (255 in 8-bit grayscale) and the total number of pixels:

$$\text{UACI} = \left(\frac{\text{Total\_Abs\_Diff}}{255 \times H \times W} \right) \times 100$$

5. A uniform distribution of pixel changes is indicated by a secure encryption produce of approximately 33.4635%.

Both the NPCR and UACI tests are important because the former ensures that the encryption is extremely vulnerable to even small changes, preventing predictability, while the latter ensures that pixel changes are statistically uniform by preventing frequency analysis.
Now, **Table 4.1** and **Figure 4.4** displays the average NPCR/UCAI value.

Table 4.1: NPCR and UACI values for each image

Metric	Image	Value (%)
NPCR	Mandrill	99.5865
NPCR	Lena	99.6582
NPCR	Cameraman	99.6201
UACI	Mandrill	27.9499
UACI	Lena	28.6563
UACI	Cameraman	31.1600

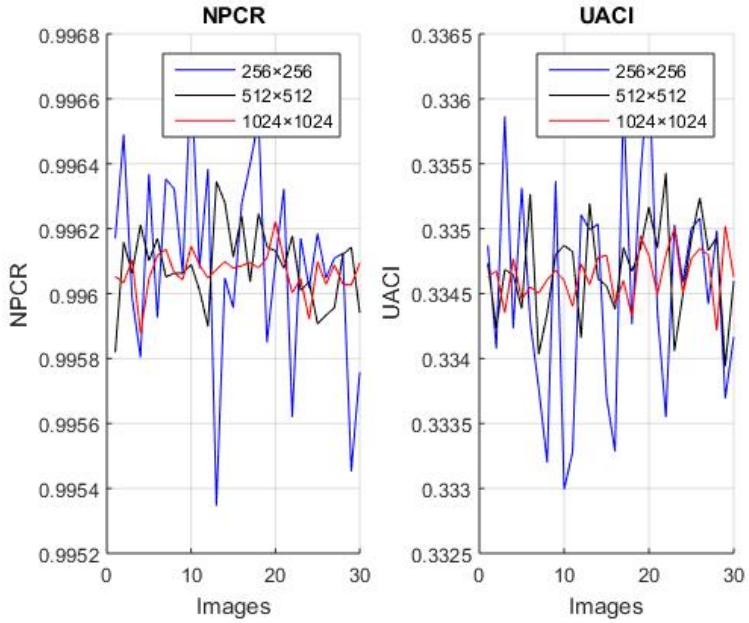


Figure 4.4: The distribution of NPCR and UACI of different size images

4.7 Statistical Cryptanalysis

As we discussed in chapter 2 the tests use in cryptography now in this section will apply test on images. If an encryption system passes well-known tests like correlation, histogram, and entropy, It can be applied to encryption in real time. Each test and its associated results are covered in full below.

4.7.1 Entropy Test

The most crucial metric for measuring the degree of unpredictability in a dataset of images is entropy. It illustrates the unpredictability of pixel intensity in an image dataset. If I were a

grayscale image, the entropy S equation would be

$$S = -\sum_{i=0}^{255} P(x_i)\log_2(Px_i) \tag{4.8}$$

High entropy indicates high randomness in image data. Thus, the suggested technique generates a lot of randomness. So, the entropy of following images shown below. With $p(i)$ representing

Table 4.2: Image Entropy Analysis

Plain-image	**Lena**	**Cameraman**	**Mandrill**
Plain image Entropy	7.7662	7.1048	7.3233
Cipher image Entropy	7.9976	7.9976	7.9969

the probability of each pixel intensity, $H = -\sum p(i)\cdot\log_2(p(i))$. One entropy value in bits is obtained from this result. Strong confusion and diffusion properties are desired in encryption, and a more randomized, information-rich image is indicated by a higher entropy (nearer to 8).

4.7.2 Histogram Test

Now we will apply histogram test in ciphered images. The cipher scheme will be considered secure if the encrypted images' histograms are uniform. The histograms of the plain and encrypted images are displayed in figures 4.5 (a)–(c) and 4.5 (d)–(f), respectively. our proposed encryption scheme's security is demonstrated by the uniform distribution of encrypted image histograms.

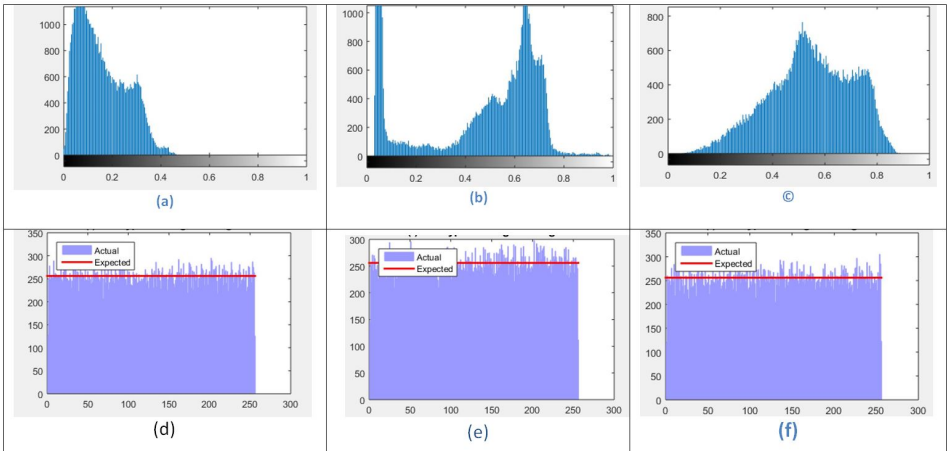


Figure 4.5: In (a)-(c) figure shows the Histogram of the Plain images and (d)-(f) shows the Histograms of the cipher texts, respectively.

Through a calculated procedure, the histogram test is used to confirm that the distribution of

pixels in the encrypted image is uniform. The encrypted image is first multiplied by 255 and rounded to integers to create 256 distinct intensity levels (pixel values normalized to [0,1)). The histogram data is then produced by using MATLAB's 'imhist()' function to count the instances of each intensity value (0–255). Each bin should have roughly $N/256$ pixels. The chi-square test quantitatively compares these observed bin counts against the expected uniform distribution by calculating the X^2 statistic, which is the sum of squared differences between observed and expected counts divided by the expected counts. A Cumulative Distribution Function (or CDF) expresses the likelihood that a random variable X will take a value less than or equal to a given value x. The empirical distribution function of the pixel values is compared to the theoretical uniform CDF. It is a basic idea in statistics and probability that is used to examine data distribution.), the Kolmogorov-Smirnov test simultaneously assesses whether the values follow a continuous uniform distribution. A bar graph is produced visually, with the y-axis displaying the frequency counts and the x-axis representing the 256 intensity levels. A horizontal red line denotes the expected uniform frequency level. In order to confirm that the encryption successfully eliminated all traces of the original image's statistical properties, the histogram is considered uniform if:

1. The X^2 statistic falls below the critical threshold (293.25 for 255 degrees of freedom at 5
2. The KS-test produces a p-value > 0.05; and (3) the bar heights fluctuate randomly around the red reference line without discernible patterns.
3. It is confirmed that the encryption effectively eliminated all traces of the original image's statistical properties because the bar heights varies randomly and uniformly around the red reference line.

4.7.3 Correlation Test

We will apply Correlation test on plain images and ciphered images. we will discuss further that what correlation values are secure for encrypted image.

The purpose of correlation test is to see if the encrypted image's neighboring pixels are still correlated, as this would be detrimental to security. Values appear random when this correlation is broken by a good encryption (ideally, correlation ≈ 0). **Table 4.2** displays the association between various images. we will also compared the results in **Table 4.3** . **Figure 4.6** shows the correlation of each plain image in the USC-SIPI database. The suggested method passes the correlation test, according to the findings in **Figures 4.6, 4.7, 4.8**.

Table 4.3: Correlation Analysis for Original and Encrypted Images

Image	**Original Image**			**Encrypted Image**		
	Horizontal	**Vertical**	**Diagonal**	**Horizontal**	**Vertical**	**Diagonal**
Cameraman	0.9469	0.9626	0.9062	-0.0185	0.0334	-0.0564
Lena	0.9128	0.9555	0.8947	-0.0361	0.0178	0.0067
Mandrill	0.8732	0.8297	0.7854	0.0040	-0.0040	-0.0021

One essential statistical technique for assessing an image encryption algorithm's competency is the correlation test. It calculates how similar the values of nearby pixels in an image are to one another. Because of the slow changes in intensity throughout the image, neighboring pixels in plain images, whether they are oriented diagonally, vertically, or horizontally, tend to show strong correlation. There are three ways to test correlation:

Horizontal: (x, y) and $(x+1, y)$

Vertical: (x, y) and $(x, y+1)$

Diagonal: (x, y) and $(x+1, y+1)$

This statistical redundancy should be removed by a robust encryption algorithm, though, creating an encrypted image in which neighboring pixels seem statistically independent and uncorrelated. In this work, 2000 pairs of adjacent pixels in each of the three location directions (horizontal, vertical, and diagonal) are randomly selected from both the plain and ciphered images in order to perform the correlation analysis. The intensity values of the pixel and its neighbor are noted for every sampled pair, and the Pearson correlation coefficient is calculated. Strong correlation is indicated by values close to $+1$ or -1, whereas no correlation is indicated by values close to 0. This coefficient measures the linear relationship between pixel pairs. Additionally, scatter plots are produced to show the pixel relationships graphically. The encryption process effectively breaks the plain image's predictable structure, improving security, as demonstrated by the nearly zero correlation of the encrypted image and the high correlation of the plain image. The ECC-PRNG-based encryption technique successfully generates a cipher image with high statistical randomness, as evidenced by the significant decrease in correlation coefficients in all directions.

4.8 Discussion

The experimental findings used to assess the effectiveness of the suggested ECC-based image encryption scheme are shown in this section. The algorithm was validated under various conditions using a number of standard test images, such as Lena, Cameraman, and Mandrill. Several

Table 4.4: Lena Grayscale Image comparison

Algorithm	**Entropy**	**UACI**	**Corr-H**	**Corr-D**	**Corr-V**	**NPCR**
Our	7.9976	28.65630	-0.0361	0.0178	0.0067	99.6582
[31]	7.9975	33.670	0.0030	0.0096	0.0026	99.610
[59]	7.9975	33.472	-0.0018	-0.0009	0.0011	99.614
[49]	7.9965	33.392	0.0029	-0.0003	0.0080	99.617
[50]	7.9972	33.423	0.0069	0.0075	0.0479	99.625
[60]	7.9977	33.413	0.0003	-0.0003	-0.0000	99.621
[48]	7.9974	33.463	0.0004	0.0051	0.0051	99.606
[61]	7.9976	33.451	-0.0018	0.0040	-0.0006	99.609
[51]	7.9967	34.080	-0.0003	-0.0066	-0.0013	99.580
[52]	7.9970	33.505	0.0119	0.0011	0.0092	99.594
[53]	7.9971	33.456	-0.0029	0.0004	-0.0017	99.599
[54]	7.9970	33.419	0.0086	0.0009	0.0024	99.605
[55]	7.9962	33.384	0.0015	0.0057	0.0041	99.633
[56]	7.9970	33.546	-0.0016	-0.0026	0.0043	99.614
[57]	7.9973	33.458	-0.0023	-0.0029	0.0016	99.626
[58]	7.9975	33.457	-0.0034	-0.0063	0.0013	99.621
[62]	7.9966	33.452	-0.0008	-0.0101	0.0014	99.617
[63]	7.9974	33.356	0.0102	0.0052	0.0067	99.580

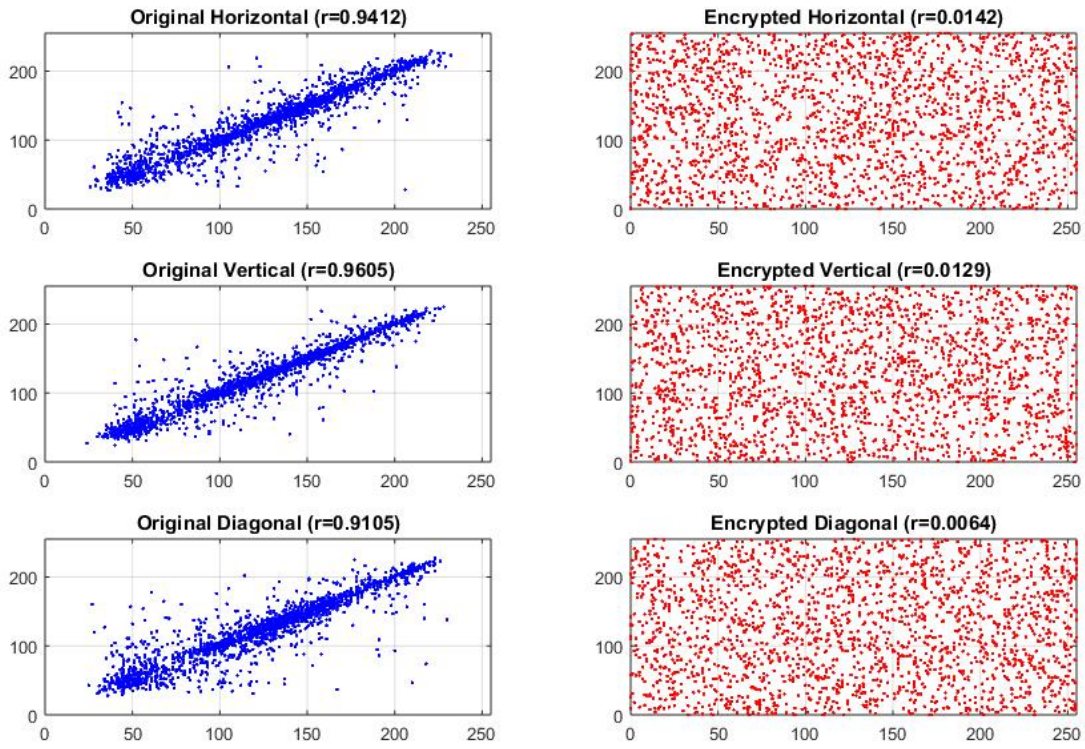


Figure 4.6: Adjacent pixel wise distribution analysis for $Lena_{256\times256}$ of original Horizontal, vertical and diagonal image; Adjacent pixel wise distribution analysis for $Lena_{256\times256}$ of Encrypted(cipher image) Horizontal, vertical and diagonal image;

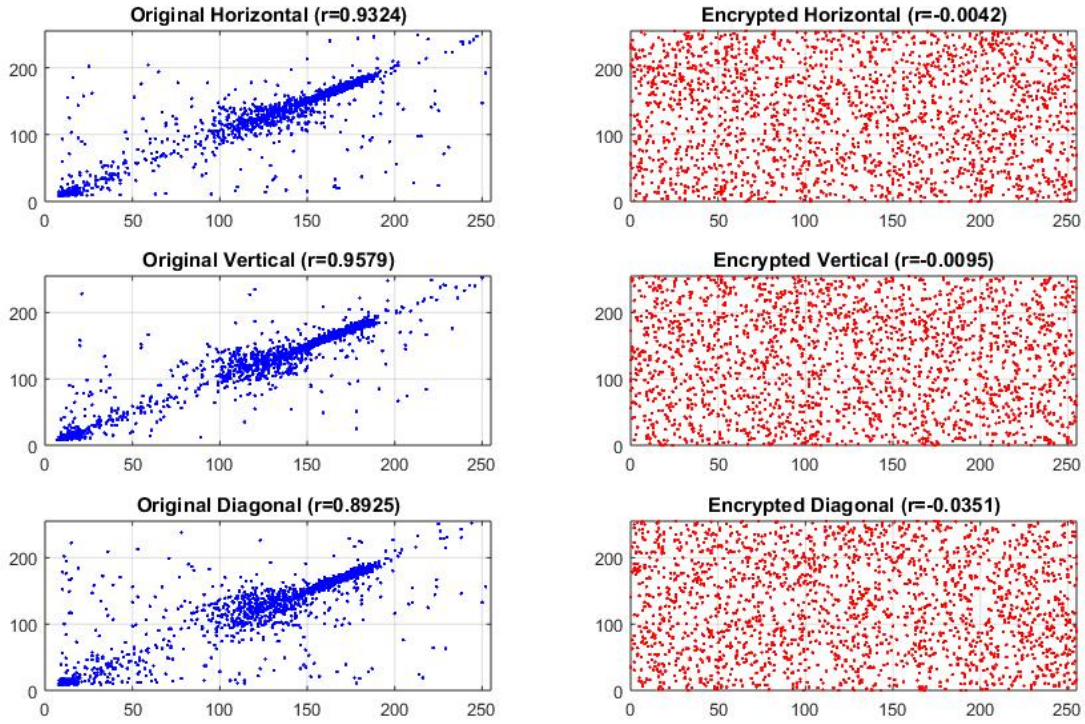


Figure 4.7: Adjacent pixel wise distribution analysis for $cameraman_{256\times256}$ of original Horizontal, vertical and diagonal image; Adjacent pixel wise distribution analysis for $cameraman_{256\times256}$ of Encrypted image Diagonal, Horizontal and vertical image;

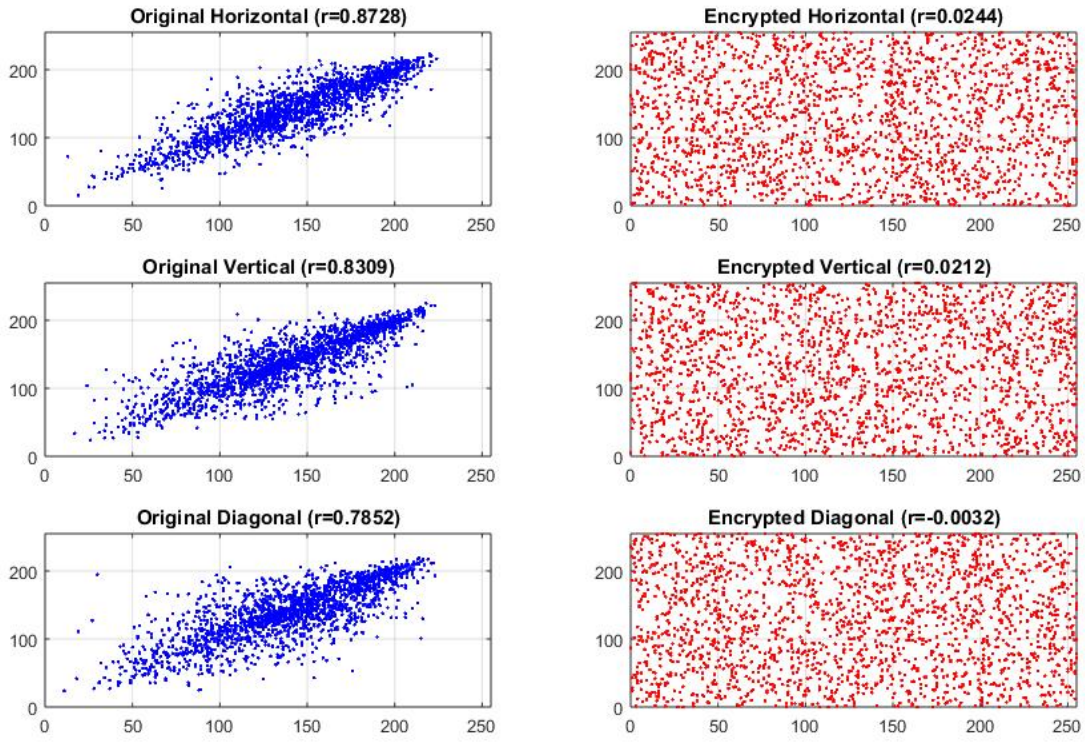


Figure 4.8: Adjacent pixel wise distribution analysis for $Mandrill_{256\times256}$ of original Horizontal, vertical and diagonal image; Adjacent pixel wise distribution analysis for $Mandrill_{256\times256}$ of Encrypted(cipher image) Horizontal, vertical and diagonal image;

statistical and differential analyses, such as NPCR, UACI, histogram uniformity, information entropy, and pixel correlation tests, were carried out to evaluate security and efficiency. The following tables and figures present the findings, which are given below:

- The typical Lena grayscale image's NPCR and UACI values in **Table 4.1**. it can be observed that the NPCR values indicating strong resistance against differential attacks
- **Table 4.2** presents the entropy value of the plain and cipher test images. The entropy of the plain images is significantly below the ideal value of 8, indicating non-random distribution of pixel intensity. After applying encryption, the entropy values are very close to 8, confirming that the cipher images approximate random noise and reveal no statistical patterns. Because of its high entropy, the encryption scheme defends against entropy-based attacks and successfully conceals all valuable information from possible attackers.
- **Figure 4.5** illustrates the histogram of plain and cipher images of cameramen, Mandrill, and Lena. The histogram of the plain image shows significant peaks and valleys, reflecting non-uniform pixel intensity distribution. In contrast, the cipher image is rarely uniform, with no visible patterns, which indicates the pixel values are well randomized. This consistent distribution demonstrates how well the suggested encryption technique hides the plain image's statistical characteristics, preventing histogram-based attacks.
- **Table 4.3** shows the correlation coefficients of adjacent pixels in plain and cipher images. As illustrated in **Figure 4.7, 4.8, 4.6**, the scatter plots of the plain images show dense clustering along the diagonal line, indicating strong correlation. In contrast, the cipher images display a uniform noise-like distribution, further confirming the effectiveness of the proposed method.
- **Table 4.4** presents a performance comparison between the proposed ECC-based image encryption scheme and several existing methods reported in [31, 59, 49, 50, 60, 48, 61, 51, 52, 53, 54, 55, 56, 57, 58, 62, 63].

CHAPTER 5

CONCLUSION

5.1 Overview

In this chapter we describe our proposed algorithm and its efficiency. In this section, we discuss how we improve security in our algorithms.

5.2 Summary and Conclusion

In our ECC-Based PRNG Generator we use the suggested framework and reducing the EC and non-EC operations, the presented PRNG's design is straightforward and effective. As a result, the proposed encryption system is an excellent fit for real-time applications because it uses minimal computational resources. To sum up, EC's make excellent candidates for PRNG design. In secure curves with big prime numbers, the amount of bits in each point coordinate is appropriate for bit extraction. By fusing the algebraic complexity of elliptic curves with random perturbations and nonlinear arithmetic, this ECC-based PRNG offers a promising method for generating random numbers. Additional improvements, such changing curve parameters, secure seeding, and formal testing, can increase its durability and applicability for cryptographic applications, even though the functional and visual properties show great unpredictability.

In the second algorithm we make the ECDH protocol, which we use in image encryption. In this algorithm, we used modular arithmetic over a finite field to develop the Elliptic Curve Diffie-Hellman (ECDH) key exchange mechanism. The code effectively illustrated how Alice and Bob, two parties, might exchange elliptic curve public keys obtained from private scalars to safely build a common shared secret across an insecure channel. Both sides independently

calculated the same shared secret using elliptic curve point addition and scalar multiplication, confirming the protocol's accuracy and security. This shared secret, represented as a point on the elliptic curve, may be subjected to further processing (e.g., hashing) in order to be used in symmetric encryption systems. The answer exemplifies the fundamental strength of elliptic curve cryptography (ECC), which offers strong security with relatively small key sizes, making it suitable for resource-constrained scenarios like embedded systems and wireless communication protocols.

In our 3rd algorithm, we aim to improve our PRNG so that it can generate points with higher randomness. High-entropy, unpredictable random numbers are produced by the implemented ECC-based PRNG system using secure hash functions and elliptic curve encryption. The system guarantees cryptographic strength and effective key agreement by combining precomputation, hidden generation techniques, and Diffie-Hellman key exchange. Strong randomness features are displayed by the final 3D PRNG outputs, making them appropriate for safe applications including cryptographic protocols, secure communications, and image encryption.

The XOR-permutation procedure in our ECC based PRNG image cipher takes advantage of ECC's asymmetric security to convert it into an effective symmetric encryption method. After exchanging elliptic-curve public keys, Alice and Bob reach the same curve point without disclosing their secrets. By hashing the curve's (x, y) coordinates, they are able to obtain an identical 256-bit key that never crosses the channel. In addition to driving a key-dependent permutation that confuses all pixel positions (diffusion), that key also seeds a PRNG whose bytes are XOR-mixed with every pixel (confusion). While anyone without the key sees only noise, Bob can undo the shuffle and XOR to restore the image flawlessly because both procedures are fully reversible with the same PRNG output and permutation order.

5.3 Future Work

While ECC is widely regarded as a robust and efficient cryptographic scheme, the following challenges remain:

Scalar Multiplication Optimization: Although those techniques are available, scalar multiplication is still the performance hindrance of ECC operations, especially in the limited-resource scenario.

Side-Channel Attacks: Timing and power analysis attack on ECC implementations are possible if security measures are not adopted.

Quantum Computing Threat: The Elliptic Curve Discrete Logarithm Problem (ECDLP) could

theoretically be solved by new quantum algorithms like Shor's algorithm, lowering the security of ECC once sufficiently powerful quantum computers become available.

This presents the direction for further research work to improve ECC performance and to strengthen its security, especially in the post-quantum cryptography environment.

REFERENCES

[1] S. Bhattacharya, "Cryptology and information security—past, present, and future role in society," *International Journal on Cryptography and Information Security (IJCIS)*, vol. 9, no. 1/2, pp. 13–36, 2019.

[2] F. Maqsood, M. Ahmed, M. M. Ali, and M. A. Shah, "Cryptography: A comparative analysis for modern techniques," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 6, pp. 1–6, 2017.

[3] S. Chandra, S. Paira, S. S. Alam, and G. Sanyal, "A comparative survey of symmetric and asymmetric key cryptography," in *Proceedings of the 2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE)*, pp. 83–93, IEEE, 2014.

[4] E. Oswald, "Introduction to elliptic curve cryptography," *Institute for Applied Information Processing and Communication, Graz University Technology*, 2002.

[5] V. Gupta, S. Gupta, S. Chang, and D. Stebila, "Performance analysis of elliptic curve cryptography for ssl," in *Proceedings of the 1st ACM workshop on Wireless security*, pp. 87–94, 2002.

[6] M. Bafandehkar, S. M. Yasin, R. Mahmod, and Z. M. Hanapi, "Comparison of ecc and rsa algorithm in resource constrained devices," in *2013 international conference on IT convergence and security (ICITCS)*, pp. 1–3, IEEE, 2013.

[7] D. Hankerson, S. Vanstone, and A. Menezes, *Guide to elliptic curve cryptography*. Springer, 2004.

[8] H. Loriya, A. Kulshreshta, and D. Keraliya, "Security analysis of various public key cryptosystems for authentication and key agreement in wireless communication network,"

International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), vol. 6, no. 2, pp. 267–274, 2017.

[9] V. S. Miller, "Use of elliptic curves in cryptography," in *Advances in Cryptology – CRYPTO '85, Santa Barbara, CA, USA*, vol. 218, pp. 417–426, 1985.

[10] N. Koblitz, A. Menezes, and S. Vanstone, "The state of elliptic curve cryptography," *Designs, codes and cryptography*, vol. 19, pp. 173–193, 2000.

[11] C. E. Shannon, "Communication theory of secrecy systems," *The Bell system technical journal*, vol. 28, no. 4, pp. 656–715, 1949.

[12] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.

[13] M. Amara and A. Siad, "Elliptic curve cryptography and its applications," in *Proceedings of the International Workshop on Systems, Signal Processing and Their Applications (WOSSPA)*, pp. 247–250, IEEE, 2011.

[14] U. Hayat and N. A. Azam, "A novel image encryption scheme based on an elliptic curve," *Signal Processing*, vol. 155, pp. 391–402, 2019.

[15] W. Diffie and M. E. Hellman, "New directions in cryptography," in *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*, pp. 365–390, 2022.

[16] R.-I. Chang, C.-W. Chiang, and Y.-H. Hung, "Grouping sensors for the key distribution of implicit certificates in wireless sensor networks," *Electronics*, vol. 12, no. 13, p. 2815, 2023.

[17] M. Scott, "Implementing cryptographic pairings," *Lecture Notes in Computer Science*, vol. 4575, p. 177, 2007.

[18] K. Gupta, S. Silakari, R. Gupta, and S. A. Khan, "An ethical way of image encryption using ecc," in *2009 First International Conference on Computational Intelligence, Communication Systems and Networks*, pp. 342–345, IEEE, 2009.

[19] X. Li, J. Chen, D. Qin, and W. Wan, "Research and realization based on hybrid encryption algorithm of improved aes and ecc," in *2010 International Conference on Audio, Language and Image Processing*, pp. 396–400, IEEE, 2010.

[20] Z. E. Dawahdeh, S. N. Yaakob, and A. M. Sagheer, "Modified elgamal elliptic curve cryptosystem using hexadecimal representation," *Indian Journal of Science and Technology*, vol. 8, no. 15, pp. 1–8, 2015.

[21] L. C. Washington, *Elliptic curves: number theory and cryptography*. Chapman and Hall/CRC, 2008.

[22] A. Menezes, "Evaluation of security level of cryptography: The elliptic curve discrete logarithm problem (ecdlp)," *University of Waterloo*, vol. 14, pp. 1–24, 2001.

[23] R. Haakegaard and J. Lang, "The elliptic curve diffie-hellman (ecdh)," *Online at https://koclab. cs. ucsb. edu/teaching/ecc/project/2015Projects/Haakegaard+ Lang. pdf*, 2015.

[24] S. H. Islam and G. Biswas, "Design of two-party authenticated key agreement protocol based on ecc and self-certified public keys," *Wireless Personal Communications*, vol. 82, no. 4, pp. 2727–2750, 2015.

[25] A. R. Omondi, "Elliptic-curve basics," in *Cryptography Arithmetic: Algorithms and Hardware Architectures*, pp. 225–241, Springer, 2020.

[26] U. M. Maurer and S. Wolf, "The diffie–hellman protocol," *Designs, Codes and Cryptography*, vol. 19, no. 2, pp. 147–171, 2000.

[27] S. M. C. Vigila and K. Muneeswaran, "Elliptic curve based key generation for symmetric encryption," in *2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies*, pp. 824–829, IEEE, 2011.

[28] S. Baghbanijam, H. Sanaei, and M. Farajzadeh, "An improved authentication & key exchange protocol based on ecdh for wsns," in *2022 30th International Conference on Electrical Engineering (ICEE)*, pp. 563–569, IEEE, 2022.

[29] E. Avaroğlu, I. Koyuncu, A. B. Özer, and M. Türk, "Hybrid pseudo-random number generator for cryptographic systems," *Nonlinear Dynamics*, vol. 82, no. 1, pp. 239–248, 2015.

[30] A. Zaru and M. Khan, "General summary of cryptography," *Journal of Engineering Research and Application*, vol. 8, no. 02, pp. 68–71, 2018.

[31] N. A. Azam, G. Murtaza, and U. Hayat, "A novel image encryption scheme based on elliptic curves and coupled map lattices," *Optik*, vol. 274, p. 170517, 2023.

[32] M. F. Khan, A. Ahmed, and K. Saleem, "A novel cryptographic substitution box design using gaussian distribution," *IEEE Access*, vol. 7, pp. 15999–16007, 2019.

[33] D. Lambić, "A novel method of s-box design based on discrete chaotic map," *Nonlinear dynamics*, vol. 87, pp. 2407–2413, 2017.

[34] A. H. Zahid and M. J. Arshad, "An innovative design of substitution-boxes using cubic polynomial mapping," *Symmetry*, vol. 11, no. 3, p. 437, 2019.

[35] G. Murtaza, N. A. Azam, and U. Hayat, "Designing an efficient and highly dynamic substitution-box generator for block ciphers based on finite elliptic curves," *Security and Communication Networks*, vol. 2021, no. 1, p. 3367521, 2021.

[36] S. Ibrahim and A. M. Abbas, "Efficient key-dependent dynamic s-boxes based on permutated elliptic curves," *Information Sciences*, vol. 558, pp. 246–264, 2021.

[37] F. Özkaynak and A. B. Özer, "A method for designing strong s-boxes based on chaotic lorenz system," *Physics Letters A*, vol. 374, no. 36, pp. 3733–3738, 2010.

[38] B. B. Cassal-Quiroga and E. Campos-Cantón, "Generation of dynamical s-boxes for block ciphers via extended logistic map," *Mathematical Problems in Engineering*, vol. 2020, no. 1, p. 2702653, 2020.

[39] A. A. Abdellatif and F. Holzapfel, "Model based safety analysis (mbsa) tool for avionics systems evaluation," in *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, pp. 1–5, IEEE, 2020.

[40] A. A. Abd EL-Latif, B. Abd-El-Atty, and S. E. Venegas-Andraca, "A novel image steganography technique based on quantum substitution boxes," *Optics & Laser Technology*, vol. 116, pp. 92–102, 2019.

[41] G. Liu, J. Zhang, G. Xi, R. Zuo, and S. Liu, "Designing mg alloys with high ductility: Reducing the strength discrepancies between soft deformation modes and hard deformation modes," *Acta Materialia*, vol. 141, pp. 1–9, 2017.

[42] U. Hayat, N. A. Azam, and M. Asif, "A method of generating 8× 8 substitution boxes based on elliptic curves," *Wireless Personal Communications*, vol. 101, pp. 439–451, 2018.

[43] M. B. Farah, A. Farah, and T. Farah, "An image encryption scheme based on a new hybrid chaotic map and optimized substitution box," *Nonlinear Dynamics*, vol. 99, no. 4, pp. 3041–3064, 2020.

[44] T. Farah, R. Rhouma, and S. Belghith, "A novel method for designing s-box based on chaotic map and teaching–learning-based optimization," *Nonlinear dynamics*, vol. 88, no. 2, pp. 1059–1074, 2017.

[45] E. Biham and A. Shamir, "Differential cryptanalysis of des-like cryptosystems," *Journal of CRYPTOLOGY*, vol. 4, pp. 3–72, 1991.

[46] M. Matsui, "Linear cryptanalysis method for des cipher," in *Workshop on the Theory and Application of of Cryptographic Techniques*, pp. 386–397, Springer, 1993.

[47] Y. Wu, J. P. Noonan, S. Agaian, *et al.*, "Npcr and uaci randomness tests for image encryption," *Cyber journals: multidisciplinary journals in science and technology, Journal of Selected Areas in Telecommunications (JSAT)*, vol. 1, no. 2, pp. 31–38, 2011.

[48] X. Wang and M. Zhang, "An image encryption algorithm based on new chaos and diffusion values of a truth table," *Information Sciences*, vol. 579, pp. 128–149, 2021.

[49] C. Zou, X. Wang, and H. Li, "Image encryption algorithm with matrix semi-tensor product," *Nonlinear Dynamics*, vol. 105, no. 1, pp. 859–876, 2021.

[50] K. Hosny, S. Kamal, M. Darwish, and G. Papakostas, "New image encryption algorithm using hyperchaotic system and fibonacci q-matrix. electronics 2021 (10): 1066," 2021.

[51] M. Y. Abdelatty and M. A. Swillam, "Hybrid plasmonic electro-optical absorption modulator based on phase change characteristics of vanadium-dioxide," *Journal of Nanophotonics*, vol. 13, no. 4, pp. 046014–046014, 2019.

[52] S. Amina and F. K. Mohamed, "An efficient and secure chaotic cipher algorithm for image content preservation," *Communications in Nonlinear Science and Numerical Simulation*, vol. 60, pp. 12–32, 2018.

[53] X. Wang, X. Zhu, and Y. Zhang, "An image encryption algorithm based on josephus traversing and mixed chaotic map," *IEEE Access*, vol. 6, pp. 23733–23746, 2018.

[54] X. Wang, S. Wang, Y. Zhang, and K. Guo, "A novel image encryption algorithm based on chaotic shuffling method," *Information Security Journal: A Global Perspective*, vol. 26, no. 1, pp. 7–16, 2017.

[55] A. Pourjabbar Kari, A. Habibizad Navin, A. M. Bidgoli, and M. Mirnia, "A novel multi-image cryptosystem based on weighted plain images and using combined chaotic maps," *Multimedia Systems*, vol. 27, no. 5, pp. 907–925, 2021.

[56] X. Yan, X. Wang, and Y. Xian, "Chaotic image encryption algorithm based on arithmetic sequence scrambling model and dna encoding operation," *Multimedia Tools and applications*, vol. 80, pp. 10949–10983, 2021.

[57] Y. Niu and X. Zhang, "A novel plaintext-related image encryption scheme based on chaotic system and pixel permutation," *IEEE Access*, vol. 8, pp. 22082–22093, 2020.

[58] H.-M. Yuan, Y. Liu, T. Lin, T. Hu, and L.-H. Gong, "A new parallel image cryptosystem based on 5d hyper-chaotic system," *Signal Processing: Image Communication*, vol. 52, pp. 87–96, 2017.

[59] M. Li, M. Wang, H. Fan, K. An, and G. Liu, "A novel plaintext-related chaotic image encryption scheme with no additional plaintext information," *Chaos, Solitons & Fractals*, vol. 158, p. 111989, 2022.

[60] X. Wang and Y. Li, "Chaotic image encryption algorithm based on hybrid multi-objective particle swarm optimization and dna sequence," *Optics and Lasers in Engineering*, vol. 137, p. 106393, 2021.

[61] Y. Luo, S. Tang, J. Liu, L. Cao, and S. Qiu, "Image encryption scheme by combining the hyper-chaotic system with quantum coding," *Optics and Lasers in Engineering*, vol. 124, p. 105836, 2020.

[62] Y. Chen, Y. Hu, K. Li, C. K. Yeo, and K. Li, "Approximate personalized propagation for unsupervised embedding in heterogeneous graphs," *Information Sciences*, vol. 600, pp. 287–300, 2022.

[63] X. Wang, Y. Wang, S. Wang, Y. Zhang, and X. Wu, "A novel pseudo-random coupled lp spatiotemporal chaos and its application in image encryption," *Chinese Physics B*, vol. 27, no. 11, p. 110502, 2018.

[64] N. A. Azam, I. Ullah, and U. Hayat, "A fast and secure public-key image encryption scheme based on mordell elliptic curves," *Optics and Lasers in Engineering*, vol. 137, p. 106371, 2021.

[65] E. Z. Zefreh, "An image encryption scheme based on a hybrid model of dna computing, chaotic systems and hash functions," *Multimedia Tools and Applications*, vol. 79, no. 33, pp. 24993–25022, 2020.

[66] L. Xu, Z. Li, J. Li, and W. Hua, "A novel bit-level image encryption algorithm based on chaotic maps," *Optics and Lasers in Engineering*, vol. 78, pp. 17–25, 2016.

[67] G. Ye, "Image scrambling encryption algorithm of pixel bit based on chaos map," *Pattern Recognition Letters*, vol. 31, no. 5, pp. 347–354, 2010.

[68] T. J. Satish, M. N. S. Theja, G. G. Kumar, and V. Thanikaiselvan, "Image encryption using integer wavelet transform, logistic map and xor encryption," in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 704–709, IEEE, 2018.

[69] P. Singh, A. Yadav, and K. Singh, "Phase image encryption in the fractional hartley domain using arnold transform and singular value decomposition," *Optics and Lasers in Engineering*, vol. 91, pp. 187–195, 2017.

[70] K. Xuejing and G. Zihui, "A new color image encryption scheme based on dna encoding and spatiotemporal chaotic system," *Signal Processing: Image Communication*, vol. 80, p. 115670, 2020.

[71] A. A. Karawia and Y. A. Elmasry, "New encryption algorithm using bit-level permutation and non-invertible chaotic map," *IEEE Access*, vol. 9, pp. 101357–101368, 2021.

[72] X. Wang and H. Sun, "A chaotic image encryption algorithm based on improved joseph traversal and cyclic shift function," *Optics & Laser Technology*, vol. 122, p. 105854, 2020.

[73] U. Hayat, I. Ullah, N. A. Azam, and S. Azhar, “A novel image encryption scheme based on elliptic curves over finite rings,” *Entropy*, vol. 24, no. 5, p. 571, 2022.