Mesopotamian journal of Cybersecurity
Vol.5,No.2, **pp**. 395–423
DOI: https://doi.org/10.58496/MJCS/2025/025; ISSN: 2958-6542
https://mesopotamian.press/journals/index.php/cybersecurity

Research Article

A Novel Multi-Layered Secure Image Encryption Scheme Utilizing Protein Sequences, Dynamic Mealy Machines, 3D-AS Scrambling, and Chaotic Systems

Radhwan Jawad Kadhim[1,*], Hussein K. Khafaji[2]

[1] *Informatics Institute for Postgraduate Studies, University of Information Technology and Communications, Baghdad, Iraq.*
[2] *Computer Engineering Department-Al-Rafidain University College, Baghdad, Iraq.*

ARTICLE INFO

Article history

Received	16 Jan 2025
Revised	28 Feb 2025
Accepted	07 Mar 2025
Published	30 May 2025

Keywords

Image Encryption

Protein sequence

Mealy Machine

Chaotic System

3D-AS Scrambling

ABSTRACT

The swift rise in multimedia transmission through insecure channels has made the study of information security critically important. Image encryption holds significant importance in this context, hence necessitating the improvement of the encryption algorithms. This research introduces a Protein-Driven Mealy Machine Image Encryption with Multi-Layer Protection (PMIE-MLP) algorithm, an innovative cryptographic system to improve image security that uses a dynamic protein-based Mealy machine, a novel 3D-AS scrambling, and a chaotic system. An encryption framework comprises key generation and six protection layers: substitution, four layers of diffusion, and confusion. These layers attempt to address the confusion and diffusion principles of Shannon's cryptographic system and meet the high security standards of encrypted images. The authors conduct rigorous experiments to measure the efficiency of the PMIE-MLP along with its security quotient. These experimental results provide evidence that the PMIE-MLP is capable of achieving resistance to attacks such as brute force, statistical, differential, occlusion, and noise attacks. Moreover, the metrics obtained by the PMIE-MLP demonstrated equal or better security results than those of prior studies. A better performance analysis was achieved through the key space, chi-square value, correlation coefficient values, and image resistance to differential attacks; thus, it is evident that the PMIE-MLP is capable of transmitting colored image information in a secure manner.

1. INTRODUCTION

With the rapid growth of digital technology and communications, data transfer over the internet is important to our daily lives. In this context, images are among the most important types of data circulated in various fields, whether medical, educational, commercial, or even entertainment. As the significance of digital images increases, so does the need to safeguard these important data from unauthorized access or illicit manipulation[1], [2]. Consequently, image security has emerged as a critical concern in recent years, prompting researchers to suggest diverse techniques to safeguard images. Image encryption stands out as one of the most popular and efficient techniques [3]–[6]. Encrypting the digital image ensures the reliable protection of sensitive information through transmission and storage. Because images exhibit significant pixel correlation at neighboring locations, a large data volume, and high pixel redundancy, traditional text encryption algorithms such as DES, 3DES, AES, IDEA, and RSA are no longer suitable for use in image encryption [5], [7]–[9]. Therefore, several image encryption systems based on different areas have been presented in recent years. Despite the existence of these systems, the encryption system can still be improved more securely and efficiently. Cun et al. [10] introduced a new chaotic image encryption technique utilizing dynamic DNA coding and RNA computing. The conversion of pixels into amino acids involves several computational stages, including DNA coding, mRNA transcription, and amino acid replacement. This method incurs computational complexity, which could delay the encryption time, especially for large images. Another drawback is the limited key space in the process of DNA and mRNA coding, as it utilizes a finite set of nucleotides (A, T, G, and C for DNA and A, U, G, and C for mRNA), potentially resulting in a lower key space for this segment of the encryption. This may render the system more susceptible to brute-force attacks. Gao et al. [11] suggested a color image encryption method that combines cross-plane permutation, hyperchaotic mapping, and DNA mutation. Analysis of some ciphertext images yields average chi-square test results, particularly for image 4.2.03, which has a value of 352.3131, which is significantly greater than the critical value. A nonperfectly uniform pixel distribution in a ciphered image enhances

*Corresponding author. Email: phd202110686@iips.edu.iq

statistical attacks, as they detect nonuniform patterns. Meng and Gu [12] developed an image encryption system that combines extended DNA coding with Zig-Zag transformation along with a fractional-order laser system. The analysis of encrypted images reveals high average chi-square and high average correlation coefficient measurements with lower average NPCR and UACI values for selecting cipher images than PMIE-MLP does. Lu et al. [13] suggested an image encryption method that employs a new chaotic system with enhanced zigzag disambiguation. The 1D chaotic systems tend to have a small state space, resulting in a limited chaotic sequence range. This may restrict the total randomness and efficacy of encryption. The paper fails to provide numerical evidence of pixel correlation alongside an investigation into how the algorithm withstands common attacks, specifically brute-force attacks, for security assessment purposes. Abdul-Hameed et al. [14] established an image encryption methodology based on third-order differential equations with a 3D logistic map. The high chi square values indicate that the pixel distribution in encrypted images does not reach an adequate state of uniformity. The calculated average values of the UACI and NPCR fall below the ideal criteria for specific encrypted images. The method proves ineffective in significantly lowering the pixel correlation present between adjacent image regions. Alexan et al. [15] designed a technique to encrypt images on the basis of Rule 30 Cellular Automata with S-box components and the Lorenz system. The primary disadvantage of this strategy lies in its vulnerability to specific types of differential attacks, as the UACI values for some test images, such as the Lenna image, fall below the 33.4635 % threshold. The reduced UACI values indicate that the scheme may lack sufficient diffusion. Another drawback is that the chi-square test values are not sufficiently low, suggesting that encrypted images may still show statistical patterns from the original images, which impacts the scheme's effectiveness against statistical attacks. Zhang and Wang [16] developed an image encryption method that combines controlled zigzag transformations with bit-level encryption alongside their implementation of quantum walk concepts. The correlation between adjacent pixels, although reduced, did not decrease to adequately low levels compared with those of the PMIE-MLP, rendering this approach less secure against statistical attacks. Alexan et al. [17] designed a technique to encrypt color images that integrates the KAA map with several chaotic maps. The average UACI value for the Lenna image is 30.5681, which falls below the 33.4635 % threshold. This indicates that the system may be ineffective in countering differential attacks. The entropy values for some images are low, which indicates that the approach may lack sufficient diffusion. Wang et al. [18] proposed a technique for color image encryption that built a chaotic system from a tri-valued memristor design. Only the Lena image with dimensions of 256 × 256 undergoes experimental testing as part of this study. The 7.9895 entropy value demonstrates low randomness in the cipher image, which exposes this method to statistical attacks. The encryption algorithm suffers from weak resistance against noise interference and attacks that introduce occlusions. Wang et al. [6] presented an image encryption system using a new three-dimensional chaotic system together with bidirectional spiral transform algorithms and DNA sequence methods. The main drawback of this approach is its inability to produce a substantial decrease in pixel adjacency correlations. Some of the test images exhibit UACI and NPCR metrics that drop below the recommended values; therefore, the system appears susceptible to differential attacks because its performance measurements suggest potential security weaknesses[37]. Li and Chen [8] established an encryption method to encrypt color images that uses a 6D hyperchaotic system combined with DNA encoding technology. The encrypted images present low entropy measurements along with higher average correlation coefficients relative to those of PMIE-MLP because its confusion and diffusion properties remain ineffective.

To address the identified security weaknesses in the aforementioned studies, this study proposes a novel image cryptographic algorithm that improves image security more efficiently on the basis of a protein sequence-driven dynamic Mealy machine, a novel 3D-AS scrambling, and a chaotic system. An encryption framework comprises key generation coupled with six protective layers, which include substitution, four diffusion layers, and confusion for implementing Shannon's cryptographic principles to fulfil strict encrypted image requirements. The main contributions and novelties of this study are as follows:

- A new weighted sum method is proposed to find the initial parameters of the 3D logistic map, which demonstrates that this algorithm is exceedingly sensitive to plaintext images and is capable of effectively defending against differential attacks.
- An amino acid encoding and decoding rule is proposed to transform the image pixels.
- The principles of the protein-driven dynamic Mealy machine are combined with a 3D logistic map to substitute the pixel values with other values to enhance the degree of confusion. This is for the following reasons:
    - Protein sequence implementation adds another random component to pixel replacement operations. The encoding of amino acids through 4-bit binary sequences renders the transformation process enormously complex, as attackers struggle to identify original pixel values.
    - Unlike traditional static substitution boxes (S-boxes), which are found in conventional cryptographic techniques, the Mealy Machine operates with dynamic state transitions through chaotic sequences. Dynamic state transitions, which avoid fixed input–output relationships, result in greater unpredictability and confusion.

- ○ The 3D logistic map produces unpredictable sequences through input-dependent initial parameters. The slightest modification of an input image produces dramatically unique chaotic sequences that increase confusion through an effective avalanche effect.
- ○ The encryption system consists of six security layers that combine protein sequences and the Mealy Machine with substitution methods alongside the 3D-AS scrambling technique and chaotic sequences to destroy pixel correlation patterns and enhance confusion.
- ○ The experimental results confirm that the proposed method succeeds in decreasing the pixel correlation between adjacent pixels, which establishes robust secure encryption schemes.

- A novel scrambling method known as three-dimensional alternating scanning (3D-AS), which is used to scramble the pixels of the three channels with each other in the colored image to diminish the correlation between adjacent pixels, is proposed.
- Evaluations from experiments and security tests demonstrate that the PMIE-MLP exhibits robust resistance against differential, brute force, occlusion, statistical, and noise attacks.
- A comparison of the PMIE-MLP algorithm's metrics revealed comparable or superior security performance compared with earlier studies. Superior performance is shown by the key space, chi-square values, correlation coefficient values, NPCR values, and UACI values, proving that the PMIE-MLP can reliably protect the security of image information.

This research follows this organizational pattern: Section 2 discusses initial considerations important for developing the PMIE-MLP. Section 3 delineates the architecture of the PMIE-MLP system. Section 4 reveals the experimental simulation findings alongside a security evaluation. Finally, Section 5 presents the conclusions.

2. PRELIMINARIES

The preliminary details on the amino acid encoding rule, the chaotic system, and the mealy machine are provided in this section, as these concepts are fundamental to the PMIE-MLP.

2.1 Amino Acid Encoding Rules

In the field of bioinformatics, a protein molecule is represented as a set of twenty amino acids (AAs); therefore, these 20 amino acids can be represented digitally in 5 bits, as explained in Table 1 [19]. In the suggested image cryptographic system, only 16 amino acids are used, so we have the amino acid encoding rule, as shown in Table I. Given that one can randomly assign the binary coding of the amino acids in Table I, there are a total of 16! = 20922789888000 possible encoding rules. Hence, accurately deducing the appropriate binary encoding scheme for amino acids is exceedingly challenging.

TABLE I. AMINO ACID ENCODING RULE

Amino Acid Symbol	Binary Value	Amino Acid Symbol	Binary Value
A (1)	0000	M (9)	1000
C (2)	0001	P (10)	1001
D (3)	0010	Q (11)	1010
F (4)	0011	R (12)	1011
G (5)	0100	S (13)	1100
H (6)	0101	V (14)	1101
K (7)	0110	W (15)	1110
L (8)	0111	Y (16)	1111

2.2 Chaotic System

Currently, the field of cryptography uses chaos systems since they have the advantage and capacity to enhance the security of cryptography [20]. They are dynamical systems, have high unpredictability, are very sensitive to the initial parameters, and are mathematically described by equations that create random numbers. In recent years, researchers have proposed numerous types of chaotic systems that exhibit high levels of randomness. In this research, two chaotic systems are adopted: a 3D logistic map [21] and a chaotic circuit [22].

2.2.1 3D logistic map

The 3D logistic map exhibits a significant degree of unpredictability, resulting in enhanced security, and it can be expressed mathematically via Equations (1) to (3) [21].

$$x_{i+1} = \mu x_i(1 - x_i) + \delta y_i^2 x_i + \beta z_i^3 \quad (1)$$

$$y_{i+1} = \mu y_i(1 - y_i) + \delta z_i^2 y_i + \beta x_i^3 \quad (2)$$

$$z_{i+1} = \mu z_i(1 - z_i) + \delta x_i^2 z_i + \beta y_i^3 \quad (3)$$

Chaotic behavior manifests when the μ, δ, and β values are between $3.53 < \mu < 3.81$, $0 < \delta < 0.022$, and $0 < \beta < 0.015$, respectively. In addition, the variables x_0, y_0, and z_0 are restricted to the range of values between 0 and 1.

2.2.2 Chaotic Circuit

Recently, researchers introduced a chaotic circuit with two memristors. The dynamic behavior of this circuit is versatile over a wide range, allowing it to be used securely for communication. It is defined in Equations (4) to (7) [22].

$$A = \left(\frac{1}{M} \times b\right) - \left(\frac{1}{M \times s1} \times \frac{a}{c}\right) \tag{4}$$

$$B = -\left(\frac{1}{K} \times a\right) - \left(\frac{s2}{K} \times (d \times b)\right) \tag{5}$$

$$C = \left(\frac{a}{s1 \times c}\right)^2 - f1 \tag{6}$$

$$D = b^2 - f2 \tag{7}$$

where a, b, c, and d represent the state variables and where M, K, s1, s2, f1, and f2 are the system parameters. If we set M = 0.025, K = 0.025, s1 = 5.8, s2 = 0.825, f1 = 1.85, and f2 = 10 and the initial variables are (a = 1, b = 3, c = 1, and d = – 0.7), then the chaotic circuit system will exhibit unpredictable behavior. Thus, four sequences with a high level of randomness are generated.

2.3 Mealy Machine

The Mealy machine (MM) is a concept in computation theory that is characterized as a finite-state machine [23]. The MM may be utilized in the field of cryptography to enhance system security [24]–[27], where its output is defined with the present input and the present state of the MM [27]. MM is specified by a set of six tuples, which are denoted as MM = (₷, I, O, Ғ, Ω, p_0). In such a way,

- ₷ is a set of finite states that are not empty.
- The letters I and O represent distinct sets of finite alphabets used as inputs and outputs.
- p_0 indicates the start state, where p_0 belongs to the set ₷.
- Ғ indicates the input transition function, which is expressed as Ғ: ₷ × I ⟶ ₷.
- Ω indicates the output transition function, which is expressed as Ω: ₷ × I ⟶ O.

The following example illustrates the Mealy machine:

– ₷ = {1, 2}
– I = {h, b}
– O = {h, b}
– p_0 = 1

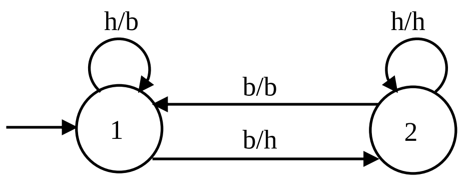


Fig. 1. Design of MM with Two States

Figure 1 shows a transition diagram for the Mealy machine. The machine has two states, labelled '1' and '2', with '1' being the start state. The Mealy machine uses 'h' and 'b' as its input and outputs alphanumeric components. The machine function here is to transform every string into a new string that exclusively utilizes the 'h' and 'b' elements from the alphabet.

TABLE II. MM STATE TABLE

Ғ: ₷ × I ⟶ ₷ (Input transition function)		
Present State	**Input = h**	**Input = b**
	Next State	Next State
1	1	2
2	2	1

Tables II and III show the state and output tables that serve as the foundation for the Mealy machine architecture. Let us consider the string 'bhbbbhb' as the input for the Mealy machine. The machine starts with its start state '1' and processes the first input symbol 'b'. As a result, the output string is 'h', which transitions to state '2'. The symbol that is subsequently encountered is 'h', which results in the output machine producing 'h' while remaining in state '2'. This process is repeated until all the input symbols have been utilized. Ultimately, the input string 'bhbbbhb' of the Mealy machine transforms into 'hhbhbbh'.

TABLE III. MM OUTPUT TABLE

Ω: Ƀ × I → O (Output transition function)		
Present State	**Input = h**	**Input = b**
	Output	Output
1	*b*	*h*
2	*h*	*b*

MM is inherently reversible. To design a trustworthy cryptographic system, it is necessary to ensure that the computations used in the encryption scheme are reversible. To retrieve the original string 'bhbbbhb', the mealy machine's inverse function is used as follows: the mealy machine begins with the start state '1' and takes the output symbol 'h' from the string 'hhbhbbh', so the original input symbol is 'b' (see Table III). In Table II, for state 1 and input 'b', the next state is '2'. After that, the machine moves to the second symbol of the string ('hhbhbbh'), which is the character 'h', so the original input symbol will be 'h', and the next state is '2'. The machine repeats this process until it uses all of the symbols ('hhbhbbh') and retrieves the original string, 'bhbbbhb'. In PMIE-MLP, we dynamically generate the MM on the basis of the amino acid symbols and the chaotic system, where the job of the MM is to confuse the image pixels.

3. THE PMIE-MLP DESIGN

This section provides a comprehensive explanation of the PMIE-MLP algorithm. Figure 2 shows the data flow diagram of the PMIE-MLP algorithm, which consists of six layers of protection in addition to the key generation module, which is detailed in Subsections 3.1 and 3.2. In the first layer, the substitution process is executed via an amino acid encoding/decoding rule (Section 3.3) combined with a dynamic Mealy machine for pixel substitution (Section 3.4). The confusion process is performed in layer 4 via the generated key, whereas the diffusion process occurs in layers 2, 3, 5, and 6 via the proposed three-dimensional alternating scanning (3D-AS) technique (Section 3.5) along with the generated key. Finally, Subsections 3.6 and 3.7 offer a detailed and systematic explanation of the image encryption and decryption processes, outlining the key steps and methodologies involved in each phase.

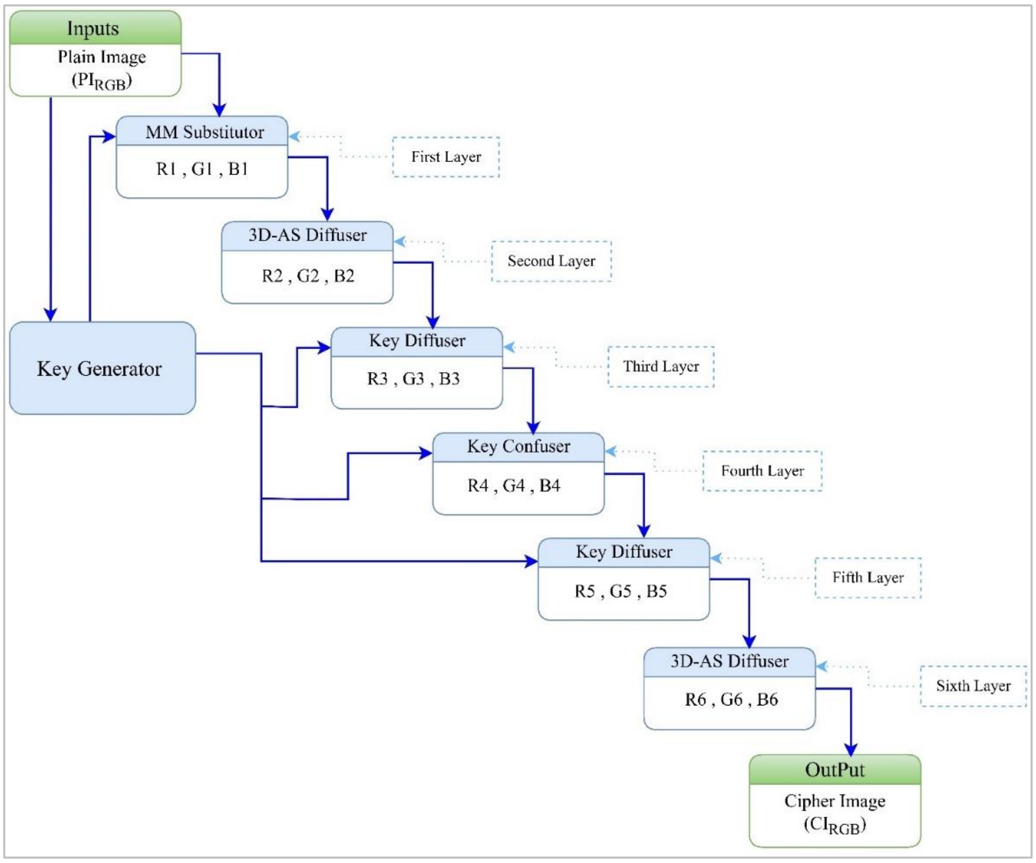


Fig. 2. Data flow diagram of the PMIE-MLP algorithm

3.1 Chaotic Sequence Generation

3.1.1 3D Logistic Map Initial Value Generation

In this subsection, the weighted sum method is suggested to find the initial parameters of the 3D logistic map. Thus, it is guaranteed that the PMIE-MLP is very sensitive to the plaintext image. The plaintext image PI is of size $u \times n$, and its components are PI_R, PI_G, and PI_B. Therefore, the initial values of x_0, y_0, z_0, μ, δ, and β in Equations (1--3) can be calculated via the proposed Equations (8--13):

$$x_0 = (\sum_{i=1}^{u}\sum_{j=1}^{n}(NR_{ij} \times Wt_{ij} + NG_{ij} \times Wt_{ij}^2 + NB_{ij} \times Wt_{ij}^3)) \bmod 1 \tag{8}$$

$$y_0 = (\sum_{i=1}^{u}\sum_{j=1}^{n}(NG_{ij} \times Wt_{ij} + NB_{ij} \times Wt_{ij}^2 + NR_{ij} \times Wt_{ij}^3)) \bmod 1 \tag{9}$$

$$z_0 = (\sum_{i=1}^{u}\sum_{j=1}^{n}(NB_{ij} \times Wt_{ij} + NR_{ij} \times Wt_{ij}^2 + NG_{ij} \times Wt_{ij}^3)) \bmod 1 \tag{10}$$

$$\mu = (3.53 + \frac{e^{x_0} - 1}{e - 1} \times (0.28) \tag{11}$$

$$\delta = \left(0.022 \times \left(1 - e^{(-y_0 \times 5)}\right)\right) \tag{12}$$

$$\beta = \left(0.015 \times \sin(\frac{\pi \times z_0}{2})\right) \tag{13}$$

where NR_{ij}, NG_{ij}, and NB_{ij} are the normalized pixel values of the three channels, respectively, at position (i, j), $0 \le (NR_{ij}, NG_{ij},$ and $NB_{ij}) \le 1$.

Wt_{ij} is the summation of coordinates i and j (i + j), and e is the exponent.

To demonstrate the effectiveness of the suggested weighted sum method, it is applied to a 512*512-pixel color image of peppers. Applying Equations (8-13) to that image yields the following initial parameters for the 3D logistic map: x_0 = 0.507812500000000, y_0 = 0.125000000000000, z_0 = 0.546875000000000, μ = 3.637818551917099, δ = 0.010224248572582, and β = 0.011358132697597. Next, we select a pixel value at random from one of the three channels and adjust it by one bit. For example, we select the green channel and alter the pixel value at location (25, 167) from 206 to 207. When equations (8-13) are applied to the modified image, the 3D logistic map initial parameters are x_0 = 0.011718750000000, y_0 = 0.976562500000000, z_0 = 0.671875000000000, μ = 3.531920844032654, δ = 0.021833335096226, and β = 0.013051304866631. Therefore, Equations (8)–(13) demonstrate that the weighted sum values of the plain image have a direct effect on the initial parameters of the 3D logistic map. Despite minimal variations in the values of the pixels between two plain images, the resulting values from the weighted sum will exhibit substantial disparities. Consequently, the chaotic sequences employed for encryption also differ significantly, resulting in completely distinct encrypted cipher images. Thus, the proposed weighted sum method is extremely sensitive to plain images and is capable of effectively defending against differential attacks.

3.1.2 Chaotic Circuit Initial Value Generation

To further improve security, we use the intermediate key TK, created by calculating the plain image's SHA-512 hash value. Hence, each image provides different hash values, resulting in different initial values and subsequently generating different chaotic circuit sequences. First, the intermediate key TK is divided into 64 8-bit blocks (TK_1 TK_2 … TK_{64}). Then, we XOR these key blocks together to create a new set of 32 8-bit blocks, as shown below:

$$HB_j = TK_j \oplus TK_{64-j+1} \tag{14}$$

where $\oplus$ indicates the exclusive OR (XOR) operator, $j \in (1,32)$. Finally, the initial parameters a, b, c, and d of the chaotic circuit can be identified via Equations (15-18):

$$a = a_0 + \sum_{i=0}^{7} HB_{2i+1} / 10^5 \tag{15}$$

$$b = b_0 + \sum_{i=1}^{8} HB_{2i} / 10^4 \qquad (16)$$

$$c = c_0 + \sum_{i=8}^{15} HB_{2i+1} / 10^5 \qquad (17)$$

$$d = d_0 + \sum_{i=9}^{16} HB_{2i} / 10^4 \qquad (18)$$

where a, b, c and d represent modified initial state variables of the chaotic circuit and where a_0, b_0, c_0, and d_0 represent given values without any changes.

3.1.3 Chaotic Sequence Generation

For any plain image PI of size $u \times n$ ($PI_{u \times n}$):

(1) Generating 3D logistic map sequences

Step 1: Iterate Equations (1--3) $(1000 + u \times n)$ times, using the initial values (x_0, y_0, z_0, μ, δ, and β) obtained from Equations (8--13). To avoid transient effects, the initial 1000 outcomes were excluded. During each iteration, we can obtain three decimal sequences $x = (x_1, x_2, \dots, x_{u \times n})$, $y = (y_1, y_2, \dots, y_{u\times n})$, and $z = (z_1, z_2, \dots, z_{u \times n})$, which are inside the range of 0--1.
Step 2: Convert the decimal sequences in Step 1 (x, y, and z) into sequences ($\bar{x}$, $\bar{y}$, and $\bar{z}$) in the range of 1--16 to utilize them in constructing the state table and output table and in choosing an appropriate amino acid encoding rule, as shown in Equations (19--21):

$$\begin{cases} x_{tmp} = \left(\lceil\left((x + 100) \times 10^{10}\right)\rceil \bmod 16\right) + 1 \\ [\sim, x_{idx}] = \text{unique}\left(x_{tmp}\right) \\ \bar{x}\ (1{:}16) = x_{tmp}\left(\text{sort}(x_{idx})\right) \end{cases} \qquad (19)$$

$$\begin{cases} y_{tmp} = \left(\lceil\left((y + 100) \times 10^{10}\right)\rceil \bmod 16\right) + 1 \\ [\sim, y_{idx}] = \text{unique}\left(y_{tmp}\right) \\ \bar{y}\ (1{:}16) = y_{tmp}\left(\text{sort}(y_{idx})\right) \end{cases} \qquad (20)$$

$$\begin{cases} z_{tmp} = \left(\lceil\left((z + 100) \times 10^{10}\right)\rceil \bmod 16\right) + 1 \\ [\sim, z_{idx}] = \text{unique}\left(z_{tmp}\right) \\ \bar{z}\ (1{:}16) = z_{tmp}\left(\text{sort}(z_{idx})\right) \end{cases} \qquad (21)$$

where $\lceil p \rceil = \text{maximum } \{q \in Z;\ p \geq q\}$.

(2) Generating chaotic circuit sequences

Step 3: Utilizing the modified initial state variables obtained from Equations (15)–(18) and the given values (M, K, s1, s2, f1, and f2), the chaotic circuit system (equations (4)–(7)) is iterated $(1000 + u \times n)$ times via the Runge–Kutta 4th-order program (ODE45). We eliminate the first 1000 results to achieve the desired level of unpredictability. Four decimal chaotic sequences are created here: $A = (A_1, A_2, \dots, A_{u \times n})$, $B = (B_1, B_2, \dots, B_{u \times n})$, $C = (C_1, C_2, \dots, C_{u \times n})$, and $D = (D_1, D_2, \dots, D_{u \times n})$.
Step 4: Merge the three decimal chaotic sequences in step 1 (x, y, and z) with the first three decimal chaotic sequences in step 3 (A, B, and C) to prepare the final integer chaotic sequences $\bar{A} = (\bar{A}_1, \bar{A}_2, \dots, \bar{A}_{u\times n})$, $\bar{B} = (\bar{B}_1, \bar{B}_2, \dots, \bar{B}_{u\times n})$, and $\bar{C} = (\bar{C}_1, \bar{C}_2, \dots, \bar{C}_{u\times n})$, within the interval of 1--$u \times n$ to utilize them in the diffusion process as follows:

$$[\sim, \bar{A}] = \text{sort}\ (A + x) \qquad (22)$$

$$[\sim, \overline{\overline{B}}] = \mathrm{sort}\,(\,B + y) \tag{23}$$

$$[\sim, \overline{\overline{C}}] = \mathrm{sort}\,(\,C + z) \tag{24}$$

where the expression $[H, T] = \mathrm{sort}\,(G)$ signifies the process of arranging the elements in G in an ascending fashion, which produces the sorted array H and the index array T. The symbol ($\sim$) denotes that the variable is unnecessary.

Step 5: Convert the last three sequences of the chaotic circuit in step 3 (B, C, and D) into integer sequences ($\overline{\overline{\overline{B}}}$, $\overline{\overline{\overline{C}}}$, and $\overline{D}$) in the range of 0--255 to utilize them in the confusion process, as shown in Equations (25--27):

$$\overline{\overline{\overline{B}}} = (\mathrm{round}(\mathrm{abs}(B) \times 2^{26}) \bmod 256 \tag{25}$$

$$\overline{\overline{\overline{C}}} = (\mathrm{round}(\mathrm{abs}(C) \times 2^{26}) \bmod 256 \tag{26}$$

$$\overline{D} = (\mathrm{round}(\mathrm{abs}(D) \times 10^{15}) \bmod 256 \tag{27}$$

3.2 Round Key Generation

In this research, we use the generated round keys to confuse and diffuse the image pixels, using a unique round key for each group of pixels. Equation (28) calculates the number of unique round keys (UK_s) required to confuse and diffuse the pixels, assuming that the key length is KL bits and that the image size is $u \times n$.

$$UK_s = \left\lceil \frac{u \times n \times 8}{KL} \right\rceil \tag{28}$$

To generate strong and highly random secret keys, we use the KE-DMM3DLMPS method proposed in the literature [28], which generates secret keys in the form of amino acids of various sizes. This method involves entering a master secret key (MSK) of a specific length into the Mealy machine to generate round keys. For a MSK length of 128 bits (32 amino acids), the KE-DMM3DLMPS generates 32 round keys in a single run, as illustrated in Equation 10 [28].

To generate a sufficient number of unique round keys (UK_s), the KE-DMM3DLMPS is run a number of times (N_{run}), as in equation 29:

$$N_{run} = \left\lceil \frac{UK_s}{KL_{aa}} \right\rceil \tag{29}$$

where KL_{aa} is the number of amino acids in a single key and where $\lceil x \rceil$ is the rounding-up function.

Given a master secret key length of KL = 256 bits (KL_{aa} = 64 amino acids) and an image size of $u = 256$ and $n = 256$, we can calculate UK_s = 2048 round keys and N_{run} = 32 runs.

In this case, the KE-DMM3DLMPS method necessitates the entry of a master secret key (MSK) for the generation of round keys in each iteration. This method generates the first 64 round keys ($rk_1^1, rk_2^1, \dots, rk_{64}^1$) using the entered MSK and considers the final round key from the first 64 round keys (rk_{64}^1) as a master secret key ($MSK = rk_{64}^1$) to generate the second set of round keys ($rk_1^2, rk_2^2, \dots, rk_{64}^2$). This process continues until a sufficient number of unique round keys ($RK = rk_1^1, rk_2^1, \dots, rk_{64}^{32}$) is generated. Finally, we merge those (RK) into a single round key, convert it into a binary key (BK) by converting every two amino acids to eight bits in accordance with Table 5 in [28], and then convert it into an integer key (IK) within the range of 0--255, as demonstrated in the following example:

Let us take a fragment of a round key with a length of 40 bits, RK = 'TKNQACHTGY', so according to Table 5 in [28], BK = 0011 0101 1100 1001 0011 0000 0110 0011 0100 1111 and IK = 53, 201, 48, 99, 79.

3.3 Image Pixel Transformation

In the PMIE-MLP, one of the 16! The rule is used to transform image pixels on the basis of the generated chaotic sequence ($\overline{z}$; Equation 21). If $\overline{z}$ = [15, 7, 12, 11, 14, 9, 6, 13, 5, 10, 4, 1, 8, 16, 3, 2] and amino_acids = $\{A, C, D, F, G, H, K, L, M, P, Q, R, S, V, W, Y\}$, then Table IV displays the amino acid encoding and decoding rule.

TABLE IV. AMINO ACID ENCODING AND DECODING RULE

Amino Acid Symbol	Binary Value	Amino Acid Symbol	Binary Value
W (15)	0000	G (5)	1000
K (7)	0001	P (10)	1001
R (12)	0010	F (4)	1010
Q (11)	0011	A (1)	1011
V (14)	0100	L (8)	1100
M (9)	0101	Y (16)	1101
H (6)	0110	D (3)	1110
S (13)	0111	C (2)	1111

The process of transforming the image pixels into protein sequences is performed before and after the confusion process by the Mealy machine. First, for each channel, the pixels are transformed to 8 bits and then to a specific amino acid before the confusion process (during encryption/decryption). After the confusion process (during encryption/decryption) is completed, the pixel values are transformed back to 8 bits and then to integers. A specific rule represents each 8 bits as two amino acids, and vice versa. For example, if the pixel value is 167, converting it into a binary value results in '10100111'. Consequently, the protein sequence associated with this pixel is 'FS', as illustrated in Table IV. Additionally, using the rule in Table IV to transform the protein sequence 'FS' into a pixel value, we can obtain a binary value of '10100111', followed by a pixel value of 167.

3.4 Design of a Dynamic Mealy Machine (DMM)

The DMM is designed with sixteen states, namely, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, and 16. The ability of a cryptographer to reconstruct a Mealy Machine (MM) when necessary defines what is meant by "dynamic" meaning, incorporating new features to eliminate any possibility of predicting its structure. To minimize complexity, every state produces a particular amino acid. Therefore, the number of amino acids used is only sixteen, namely, $D, A, C, F, G, H, L, Y, M, K, Q, R, S, P, W$ and V. The DMM is formally defined as MM = (Ѣ, I, O, Ϝ, Ω, p_0). where:

- Ѣ = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16}
- I = $\{D, A, C, F, G, H, L, Y, M, K, Q, R, S, P, W, V\}$
- O = $\{D, A, C, F, G, H, L, Y, M, K, Q, R, S, P, W, V\}$
- Ϝ and Ω = Derived with the help of the sequences, which are received at the result from the chaotic system
- p_0 = chosen through the user (p_0 = q | q ∈ Ѣ)

Algorithms I and II explain the structure of the DMM and the inverse of the DMM, which are mainly based on two tables, named SST for the secret state and SOT for the secret output. Both SST and SOT store transition functions; the former stores the input function Ϝ: Ѣ × I → Ѣ, whereas the latter stores the output function Ω: Ѣ × I → O. The entries of the 'next states' in the SST are allocated from the given set {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16} on the basis of the chaotic integer sequence $\bar{x}$ (Equation 19) and the circular shift operation to the left, as explained in Algorithm 1 [28]. Additionally, the entries of the 'Output' in the SOT are assigned from the set $\{D, A, C, F, G, H, L, Y, M, K, Q, R, S, P, W, V\}$ on the basis of the chaotic integer sequence $\bar{y}$ (Equation 20) and the circular shift operation to the left, as explained in Algorithm 2 [28].

Algorithm I. Algorithm of the Dynamic Mealy Machine

```
Input: SST, SOT, current state ($C_s$), plain text (PT)
Output: CT//CT is a cipher text
Step 1. S ← $C_s$
Step 2. CT ← blanks(length(PT))//Initialize an empty output string
Step 3. Iterate over the plain text:
  for i = 1 to length(PT) do
    inpt ← PT (i)
    $inpt_{index}$ ← find([D, A, C, F, G, H, L, Y, M, K, Q, R, S, P, W, V ] == inpt)
    $nxt_{state}$ ← SST ( S, $inpt_{index}$)
    CT (i) ← SOT ( S, $inpt_{index}$)
    S ← $nxt_{state}$
  end for
```

Algorithm II. Algorithm of the Inverse Dynamic Mealy Machine

```
Input: SST, SOT, current state ($C_s$), cipher text (CT)
Output: PT//PT is a plain text
Step 1. S ← $C_s$
Step 2. PT ← blanks(length(CT))//Initialize an empty string
Step 3. $inpt_{alphabet}$ ← [D, A, C, F, G, H, L, Y, M, K, Q, R, S, P, W, V]
Step 4. Iterate over the cipher text:
```

```
for q = 1 to length(CT) do
  outpt ← CT (q)
  $inpt_{index}$ ← find(SOT(S, :) == outpt)
  $nxt_{state}$ ← SST ( S, $inpt_{index}$)
  PT (q) ← $inpt_{alphabet}$ ($inpt_{index}$)
  S ← $nxt_{state}$
end for
```

3.5 Proposed Three-Dimensional Alternating Scanning (3D-AS) Scrambling

In image encryption, traditional zigzag or scan methods are widely used for pixel position scrambling. The pixels of the image are scanned in a sequential fashion in accordance with the "Z" form, starting from the left upper corner and then taking all subsequent pixels one by one via a fixed zigzag path, and the pixels that have been scanned are then placed in a vector following the same path. The vector is subsequently transformed into a 2D matrix [29]. Therefore, the pixels of the image can be diffused. This zigzag scrambling method is limited to traversing numbers in an N × N matrix via a fixed scanning path. Figure 3 provides an illustration of a 4 × 4 matrix as an example. Through this established pathway, the various channels of the color image cannot be scrambled.

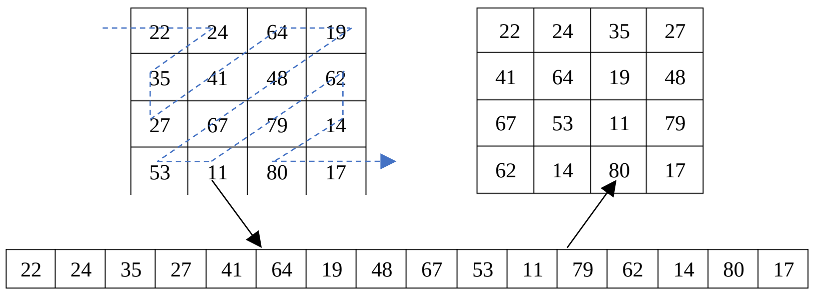


Fig. 3. 4 x 4 Standard Zigzag Scrambling

In light of its limitations, this study introduces an innovative three-dimensional alternating scanning (3D-AS) scrambling that is used to scramble the pixels of the three channels with each other in the colored image. 3D-AS starts by dividing each channel of the color image into upper and lower triangles along the main diameter and then flipping the elements of the upper triangle, as illustrated by an example of a 4 × 4 × 3 color image, as shown in Figure 4.

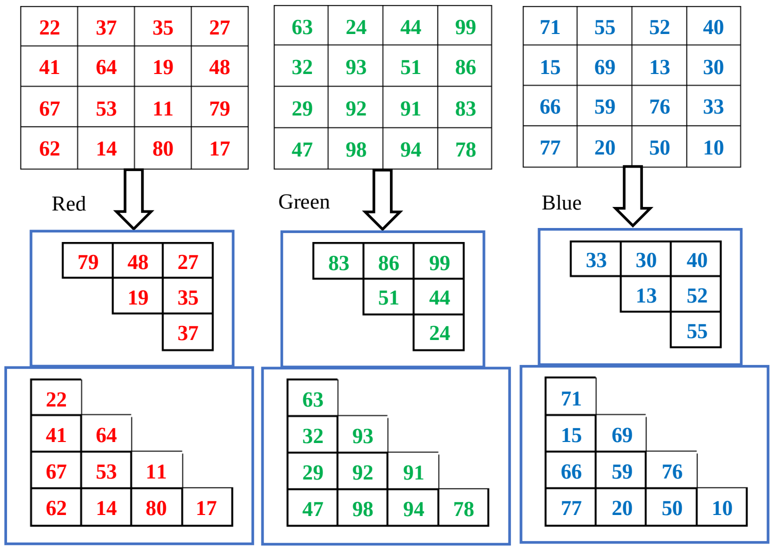


Fig. 4. Colored Image to Six Triangles

These six triangles are then arranged in a circular form, as shown in Figure 5. The diffusion process begins by alternatively scanning the entire circle from the center, where the odd circles (pixel values are surrounded by purple arrows) are first scanned and then the even circles (pixel values are surrounded by black arrows) are scanned. All the pixels that have been scanned from these circles are subsequently placed in a vector following the order of the scan. Finally, the vector is transformed into three two-dimensional matrices, as shown in Figure 6. In this method, the pixels of all channels of the image are ensured to be scrambled to eliminate pixel adjacency correlations. The well-known correlation coefficient measure (CC) provides an analytical method for testing the effectiveness of the 3D-AS scrambling method in reducing the degree of pixel correlation between adjacent elements. Table V presents the calculated CC values for connecting elements between different plaintext and scrambled images taken from three orientations across all color channels. Test outcomes indicate that the CC metrics of plain images' three channels in three directional parameters display values very close to one, whereas encrypted image CC values stand close to zero. The results show that 3D-AS scrambling provides effective pixel decorrelation between adjacent components in an image.

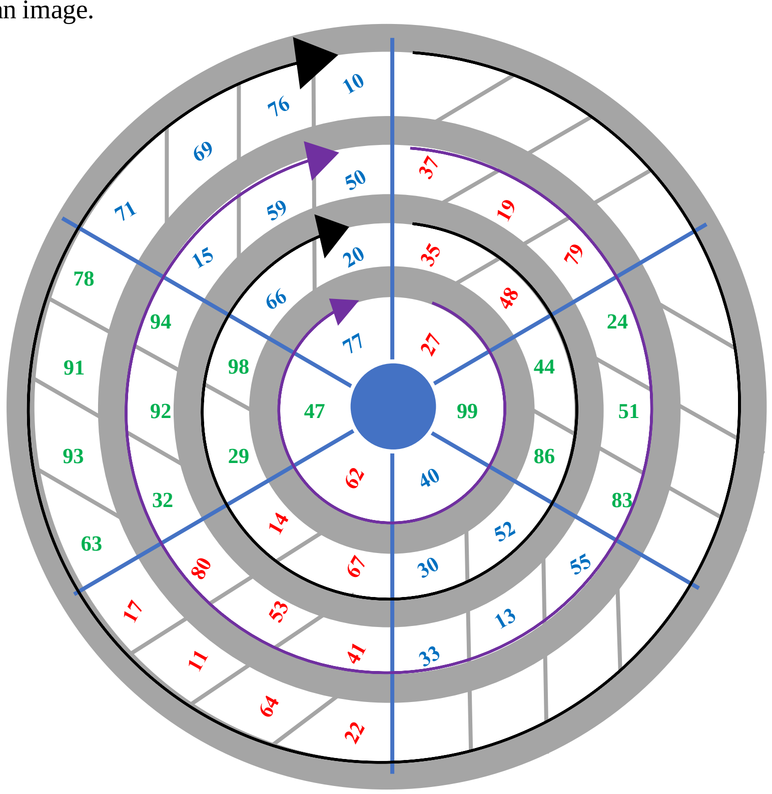


Fig. 5. Paths of the Proposed 3D-AS Scrambling

TABLE V. CC VALUES FOR VARIOUS PLAINTEXT AND SCRAMBLED IMAGES

Image	Color	Plaintext image			Scrambled image		
		Horizontal	**Vertical**	**Diagonal**	**Horizontal**	**Vertical**	**Diagonal**
Tree 256	R	0.958993	0.936077	0.915908	0.007702	-0.000303	0.000797
	G	0.968698	0.945745	0.931768	0.030226	0.005607	-0.000705
	B	0.961233	0.940562	0.926500	0.031368	-0.004373	-0.002069
Lenna 256	R	0.957216	0.978889	0.933884	-0.151646	0.007257	-0.004004
	G	0.943203	0.971369	0.919305	-0.158238	-0.003748	-0.007094
	B	0.928443	0.955931	0.900678	-0.151805	-0.002868	0.014904
Peppers 512	R	0.963525	0.966337	0.956377	-0.099269	0.004055	0.005154
	G	0.981118	0.981774	0.968658	-0.107545	-0.005323	-0.003305
	B	0.966517	0.966425	0.947794	-0.101123	0.001952	0.006058
Baboon 512	R	0.923066	0.865959	0.854341	-0.025337	0.001878	0.004855
	G	0.865479	0.765007	0.734795	-0.027026	-0.001893	0.001888
	B	0.907344	0.880892	0.839855	-0.021989	-0.000816	0.004051

Moreover, Table VI compares the CC of different scrambled color images of the 3D-AS scrambling method to that of the traditional zigzag method, which scrambles only one channel at a time [29]. Table VI shows that the 3D-AS scrambling method is much better than the traditional method in reducing the correlation between adjacent image elements.

TABLE VI. COMPARISON OF THE CC VALUES FOR VARIOUS SCRAMBED IMAGES

Image	Color	Scrambled image using 3D-AS			Scrambled image using Method in Ref. [29]		
		Horizontal	**Vertical**	**Diagonal**	**Horizontal**	**Vertical**	**Diagonal**
Tree 256	R	0.007702	-0.000303	0.000797	0.855223	0.025778	0.031473
	G	0.030226	0.005607	-0.000705	0.869506	0.079911	0.081378
	B	0.031368	-0.004373	-0.002069	0.884428	0.065674	0.067898
Lenna 256	R	-0.151646	0.007257	-0.004004	0.917400	0.123350	0.119954
	G	-0.158238	-0.003748	-0.007094	0.892121	0.090034	0.089479
	B	-0.151805	-0.002868	0.014904	0.865531	0.104593	0.102893
Peppers 512	R	-0.099269	0.004055	0.005154	0.958658	0.089313	0.088077
	G	-0.107545	-0.005323	-0.003305	0.971034	0.168233	0.167274
	B	-0.101123	0.001952	0.006058	0.948298	0.058653	0.056872
Baboon 512	R	-0.025337	0.001878	0.004855	0.852408	0.211434	0.212461
	G	-0.027026	-0.001893	0.001888	0.726344	0.049491	0.048986
	B	-0.021989	-0.000816	0.004051	0.842724	0.099946	0.098069

Red

27	99	40	62
47	77	37	19
79	24	51	83
55	13	33	41

Green

53	80	32	92
94	15	59	50
35	48	44	86
52	30	67	14

Blue

29	98	66	20
22	64	11	17
63	93	91	78
71	69	76	10

Fig. 6. Scrambled Channels

Figure 7 shows the effectiveness of the 3D-AS algorithm for recovering encrypted images with missing information. Despite the destruction of the image, the information of the original image can still be retrieved effectively.

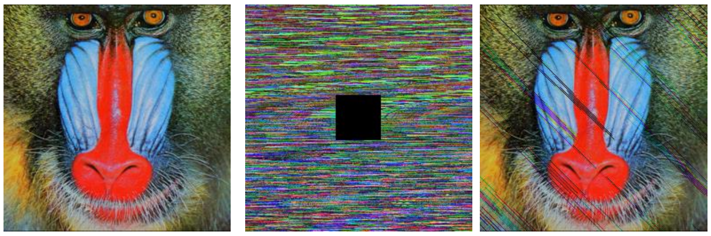


Fig. 7. Scrambling Performance of 3D-AS Scrambling

3.6 PMIE-MLP/Encryption Sub algorithm

After all the algorithms, methodologies, and techniques required to design the PMIE-MLP are elucidated, this section explains in detail its design and the cooperation among its modules. The PMIE-MLP utilizes a dynamic Mealy machine-based protein sequence, 3D-AS scrambling, and a chaotic system. The dynamic Mealy machine, which functions as a key generator (KE-DMM3DLMPS), is employed to both confuse and diffuse the image pixels. Additionally, it substitutes the pixel values with alternate values, thereby strengthening the confusion property and introducing an additional layer of security to the encryption process. An encryption structure consists of key generation and six layers of protection: substitution, four layers of diffusion, and confusion, as shown in Figure 8. The key generation process is described in Step 1, the substitution process is described in Step 3, the diffusion process is elucidated in Steps 4, 5, 7, and 8, and the confusion process is given in Step 6, as explained in Algorithm III. A detailed explanation of the encryption steps is as follows:

Algorithm III. The PMIE-MLP Algorithm

Input: Plaintext image PI_{RGB} of size $u \times n$, given values $(a_0, b_0, c_0, d_0$, M, K, s1, s2, f1, f2), and Master Secret Key
Output: Cipher image CI_{RGB} of sizeu $\times$ n
Step 1. Key Generation:
Step 1.1. Generate the sequences: ($\bar{x}, \bar{y}$, and $\bar{z}$) in the range of 1--16 (Equations 19--21), ($\bar{A}, \bar{B}$, and $\bar{C}$) in the range of 1--$u \times n$ (Equations 22--24), and ($\bar{\bar{B}}, \bar{\bar{C}}$, and $\bar{D}$) in the range of 0--255 (Equations 25--27), as explained in Section 3.1.
Step 1.2. The round keys (RK) are generated via the KE-DMM3DLMPS method and converted into integer keys (IK) in the range of 0--255, as described in Section 3.2.
Step 1.3. Create a new key by performing the exclusive-OR between one of the keys from Step 1.1 and the key from Step 1.2, as follows:

$$NK = \bar{D} \oplus IK \tag{30}$$

where $\oplus$ indicates the exclusive OR (XOR) operator.
Step 1.4. The new random key is prepared from IK in the range of 1--$u \times n$, as in Equation (31).

$$[\sim, IK_{idx}] = \mathrm{sort}\,(IK(1{:}u \times n)) \tag{31}$$

Step 2. Separate the plain image PI_{RGB} of size $u \times n$ into red (PI_R), green (PI_G), and blue (PI_B) components, with $u \times n$ being the size of each. Then, convert the pixels of each channel from a 2D matrix (PI_R,PI_G, and PI_B) to a vector R_{vec} , G_{vec} , and B_{vec} of size 1 to $u \times n$.
Step 3. First Layer (Substitution):
Step 3.1. Amino Acid (AA) Encoding: Transform the pixels of each vector from integer values (R_{vec} , G_{vec} , and B_{vec}) to amino acid values (R_{aa} , G_{aa}, and B_{aa}) based on the key ($\bar{Z}$), following the discussion in Section 3.3.
Step 3.2. Substitute the pixels of each vector (R_{aa} , G_{aa}, and B_{aa}) using the DMM based protein sequence and two processed chaotic sequences ($\bar{X}$ and $\bar{Y}$), as discussed in Section 3.4, to obtain ($R1_{aa}$, $G1_{aa}$, and $B1_{aa}$).
Step 3.3. Amino Acid Decoding: Transform the Substituted pixels of each vector from amino acid values ($R1_{aa}$, $G1_{aa}$, and $B1_{aa}$) to integer values (R1, G1, and B1) based on the key ($\bar{Z}$), following the discussion in Section 3.3.
Step 4. Second Layer (Diffusion): Convert each vector (R1, G1, and B1) to a 2D matrix ($R1_{2d}$, $G1_{2d}$, and $B1_{2d}$) with the same dimension of a plain image ($u \times n$), as described below:

$$\begin{cases} R1_{2d} = \mathrm{reshape}\,(R1, [u, n])' \\ G1_{2d} = \mathrm{reshape}\,(G1, [u, n])' \\ B1_{2d} = \mathrm{reshape}\,(B1, [u, n])' \end{cases} \tag{32}$$

Then, diffuse the pixels of the three channels ($R1_{2d}$, $G1_{2d}$, and $B1_{2d}$) using the proposed 3D-AS scrambling described in Section 3.5 to obtain R2, G2 and B2.
Step 5. Third Layer (Diffusion): Convert the diffused pixels of each channel from a 2D matrix (R2, G2 and B2) to a vector $R2_{vec}$, $G2_{vec}$, and $B2_{vec}$ of size 1 to $u \times n$.
Then, diffuse the pixels of each vector ($R2_{vec}$, $G2_{vec}$, and $B2_{vec}$) using the generated sequences (IK_{idx}) from Step 1.4 to get (R3, G3, and B3), as follows:

$$\begin{cases} R3 = R2_{vec}\,(IK_{idx}) \\ G3 = G2_{vec}\,(IK_{idx}) \\ B3 = G2_{vec}\,(IK_{idx}) \end{cases} \tag{33}$$

Step 6. Fourth Layer (Confusion): Change the values for each vector's diffused pixels via the two keys ($\bar{\bar{B}}$ and $\bar{\bar{C}}$) from step 1.1 and the key (NK) from Step 1.3 to obtain (R4, G4, and B4), as explained below:

$$\begin{cases} R4 = R3 \oplus NK \\ G4 = G3 \oplus \bar{\bar{B}} \\ B4 = B3 \oplus \bar{\bar{C}} \end{cases} \tag{34}$$

Step 7. Fifth Layer (Diffusion): Diffuse the confused pixels (R4, G4, and B4) of each vector via the generated sequences ($\bar{A}, \bar{B}, \bar{C}$) from Step 1.1 to obtain (R5, G5, and B5) as follows:

$$\begin{cases} R5 = R4\,(\bar{A}) \\ G5 = G4\,(\bar{B}) \\ B5 = B4\,(\bar{C}) \end{cases} \tag{35}$$

Step 8. Sixth Layer (Diffusion): Convert each vector (R5, G5, and B5) to a 2D matrix ($R5_{2d}$, $G5_{2d}$, and $B5_{2d}$) with the same dimension as a plain image ($u \times n$), as described below:

$$\begin{cases} R5_{2d} = \mathrm{reshape}\,(R5, [u, n])' \\ G5_{2d} = \mathrm{reshape}\,(G5, [u, n])' \\ B5_{2d} = \mathrm{reshape}\,(B5, [u, n])' \end{cases} \tag{36}$$

After that, the pixels of the three channels ($R5_{2d}, G5_{2d}$, and $B5_{2d}$) are diffused via the proposed 3D-AS to obtain R6, G6 and B6.

Step 9. The three ciphered channels (R6,G6, and B6) are combined to construct the cipher image CI_RGB.

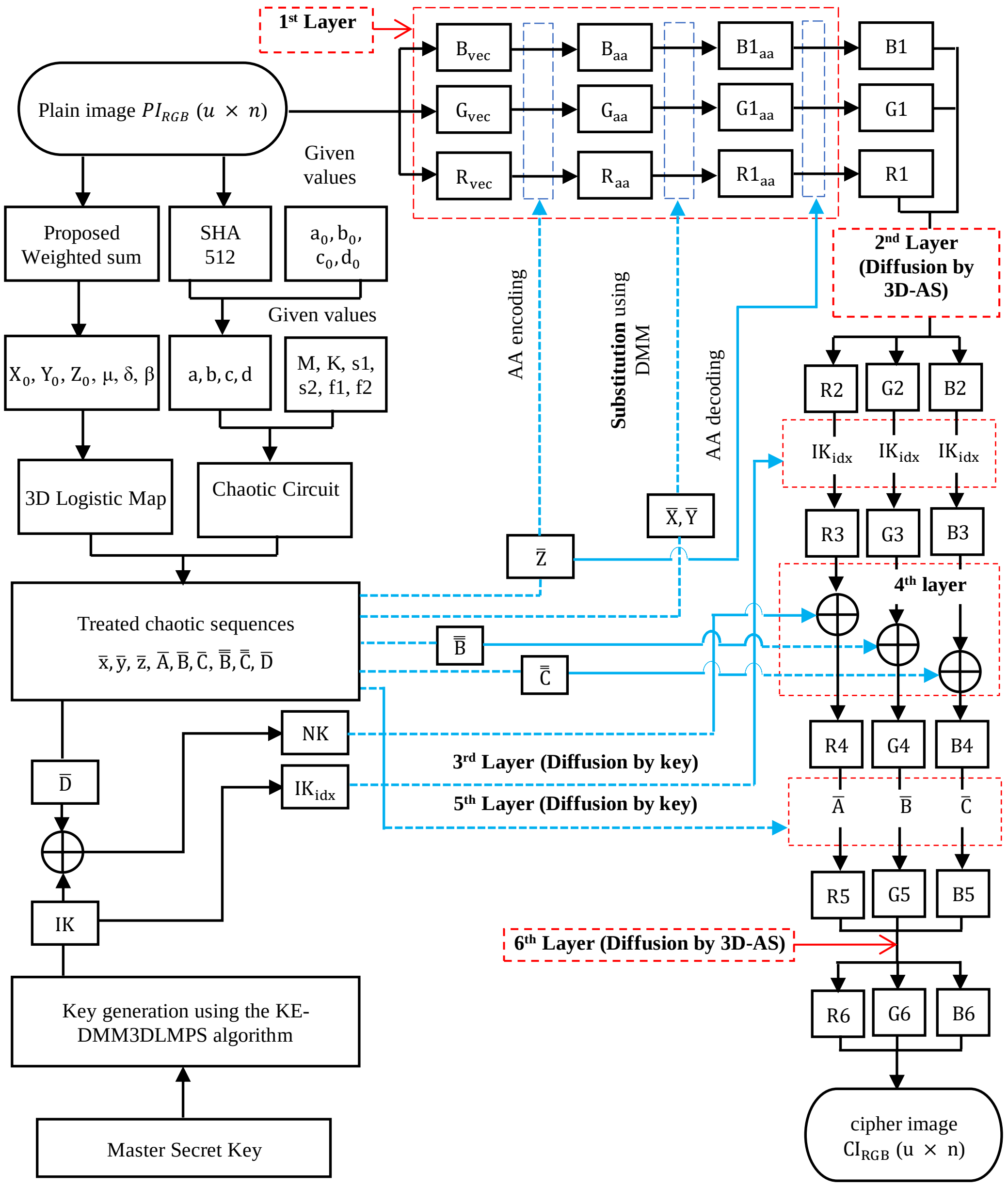


Fig. 8. PMIE-MLP Algorithm Architecture/Encryption Part

3.7 PMIE-MLP/Decryption Sub algorithm

The decryption steps of the PMIE-MLP are the inverse operation of its encryption steps since the PMIE-MLP is symmetric encryption. Before the decryption process, the initial parameters (x_0, y_0, z_0, μ, δ, and β) of the 3D logistic map, the 512-bit hash values TK (see 3.1.2), the given values (a_0, b_0, c_0, d_0, M, K, s1, s2, f1, and f2), and the MSK must be sent to the recipient side. The receiver side first uses Equations (14–27) to produce the chaotic sequences and then uses MSK to generate round keys via the KE-DMM3DLMPS method described in Section 3.2. After generating the keys needed to decrypt the image, the recipient reverses all the encryption steps to obtain the plain image, as explained in Figure 9.

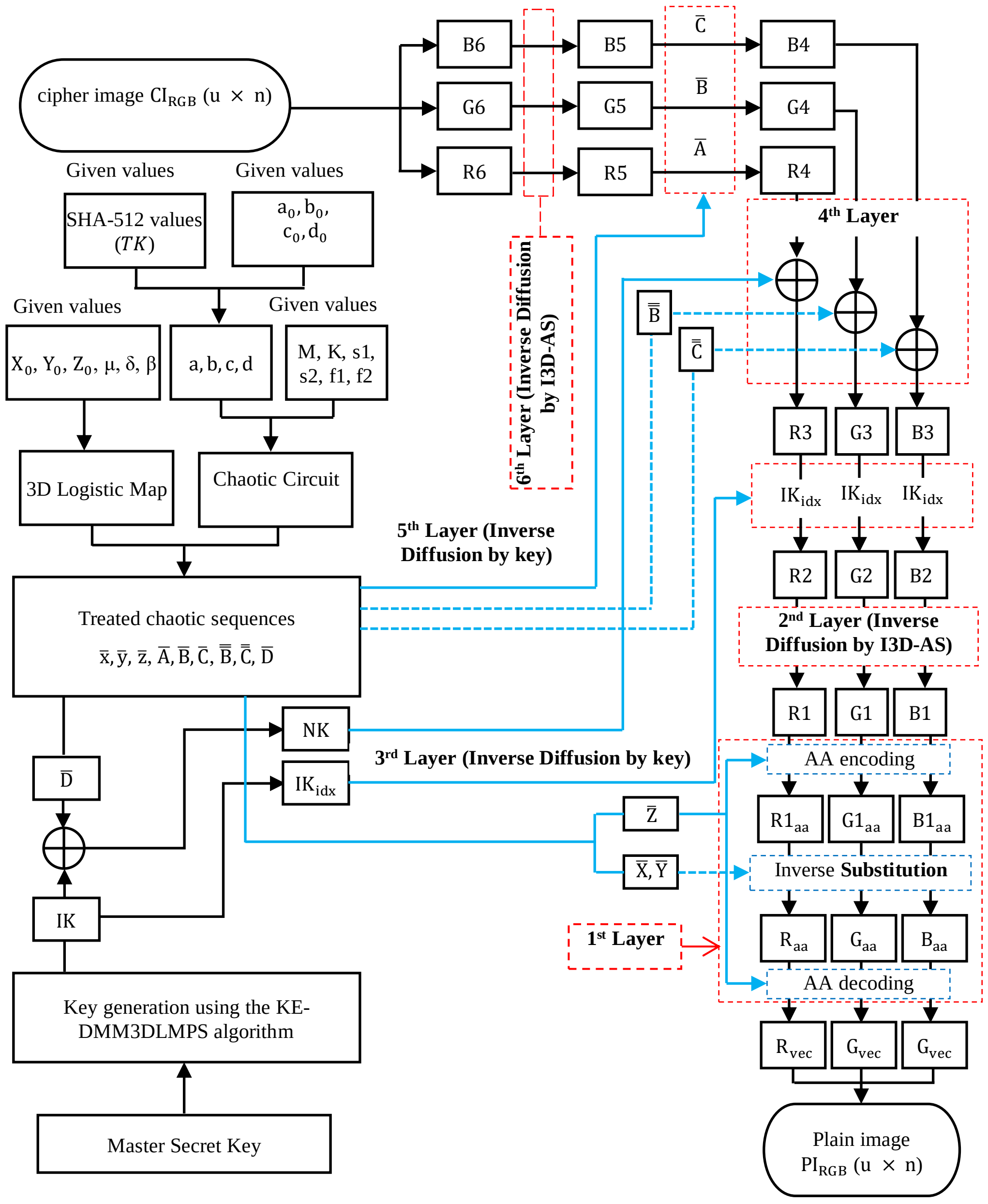


Fig. 9. PMIE-MLP Algorithm Architecture/Decryption Part

4. RESULTS OF EXPERIMENTATION AND SECURITY EVALUATION

An extensive analysis is carried out to assess the effectiveness and security of the PMIE-MLP, and the results are compared to those of some of the latest and outstanding techniques. The experimental results are obtained via an HP PC with an Intel (R) Core (TM) i5-6200U processor @ 2.30 GHz, 4 GB of RAM, an operating system of 64-bit MS Windows 10, and MATLAB (R2021a). The simulation results are performed on four standard color images. Using internet sources and the trustworthy USC-SIPI image database [30], four test images are selected: Tree (256 × 256), Lenna (256 × 256), Peppers (512 × 512), and Baboon (512 × 512). Figure 10 shows the outcome of the encryption and decryption. The performance test reveals that the encrypted images are too obscure and therefore inconsequential to contain identifiable information in relation to the plain images. In addition, the decrypted images are matched to the plain images that were decrypted via the correct secret keys. Thus, the PMIE-MLP is expected to have higher encryption and decryption capabilities to ensure data security.

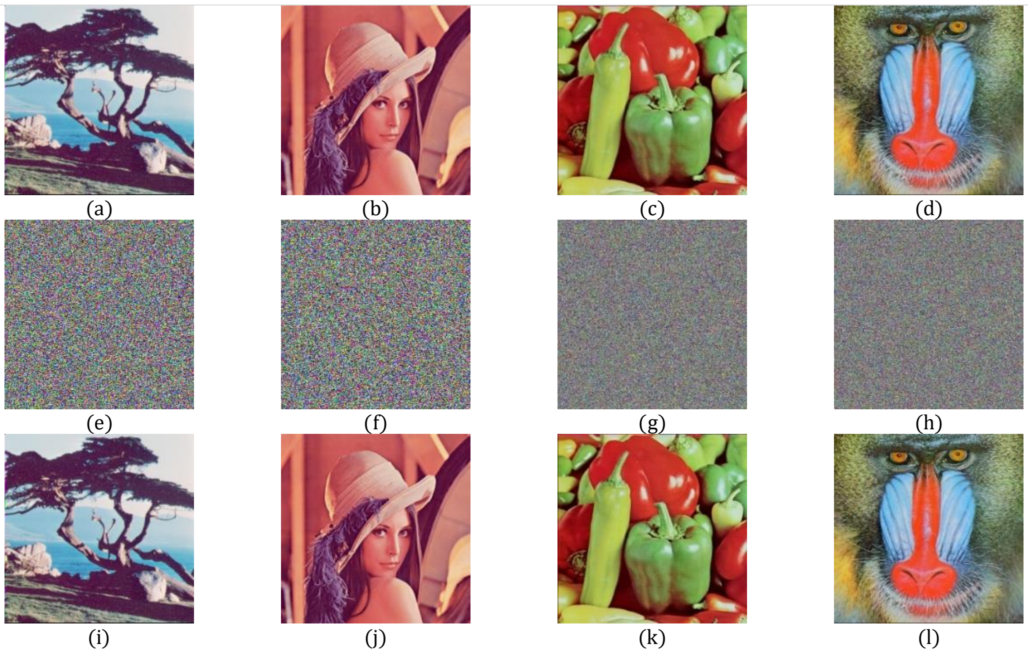


Fig. 10. Test results. (a)-(d) plaintext images. (e)-(h) encrypted images. (i)-(l) decrypted images.

4.1 Analysis of the Key Space

Creating an effective key space for highly secure applications is crucial in the development of modern cryptographic algorithms. The key space must exceed 2^{100}, with the goal of making brute-force attacks ineffective [31].

TABLE VII. COMPARISON OF THE KEY SPACES

Algorithm	Key space
Ref. [10]	2^{212}
Ref. [11]	2^{398}
Ref. [12]	2^{239}
Ref. [14]	10^{168}
Ref. [15]8	2^{425}
Ref. [16]9	$10^{32} \times 2^{256}$
Ref. [17]11	2^{478}
Ref. [18]13	4×10^{95}
Ref. [6]4	10^{266}
Ref. [8]6	10^{195}
PMIE-MLP	$\mathbf{2^{1488}}$

The key space of the PMIE-MLP includes (1) the initial values of the system parameters (μ, δ, β, M, K, s1, s2, f1, and f2) and the state variables (x_0, y_0, z_0, a, b, c, and d) of the 3D logistic map and the chaotic circuit system; therefore, if the precision of the computer is 10^{-15}, then the key space for these initial values is $(10^{15})^{16} = 10^{240} \approx 2^{797}$. (2) the amino acid encoding rules consist of 16! ≈ 2^{44} distinct rules that transform the 4 bits to amino acids, and the amino acid decoding rules consist of 16! ≈ 2^{44} distinct rules that transform the amino acids to 4 bits. (3) The key space of the KE-DMM3DLMPS is $20 \times 2^{599} \approx 2^{603}$ when the master secret key is 256 [28]. Thus, the PMIE-MLP has a key space of 2^{1488}. Table VII presents the results for the key space between the PMIE-MLP and a comparison with several recent encryption algorithms. Table VII clearly shows that the key space of the PMIE-MLP is far larger than that of certain current encryption systems. This method thus clearly offers a high degree of defense against brute force assaults.

4.2 Analysis of the Histogram

The histogram is one of the significant measures employed in the assessment of the encryption's vulnerability to statistical analysis of encrypted images. It is used to compare the distribution of the pixel density in an image [32]. Statistical analysis attacks are capable of collecting information from images containing an uneven distribution of pixels by examining the statistical properties of the encrypted images. If the image histogram is uniform, meaning that the frequency distribution of pixels in an image is even, extracting information through statistical attacks becomes more challenging. Therefore, the resulting histogram of the ciphered images produced by an effective encryption scheme must be uniform. Figures 11(a)–(c) show the histograms for each channel of the plain color Baboon image in Figure 10(d), which lacks uniformity. Additionally, Figures 11(d)-(f) show the histogram for each channel of the ciphered color image of Baboon in Figure 10(h), which is uniform. Therefore, this research's encryption algorithm produces an encrypted image that can withstand statistical analysis attacks.

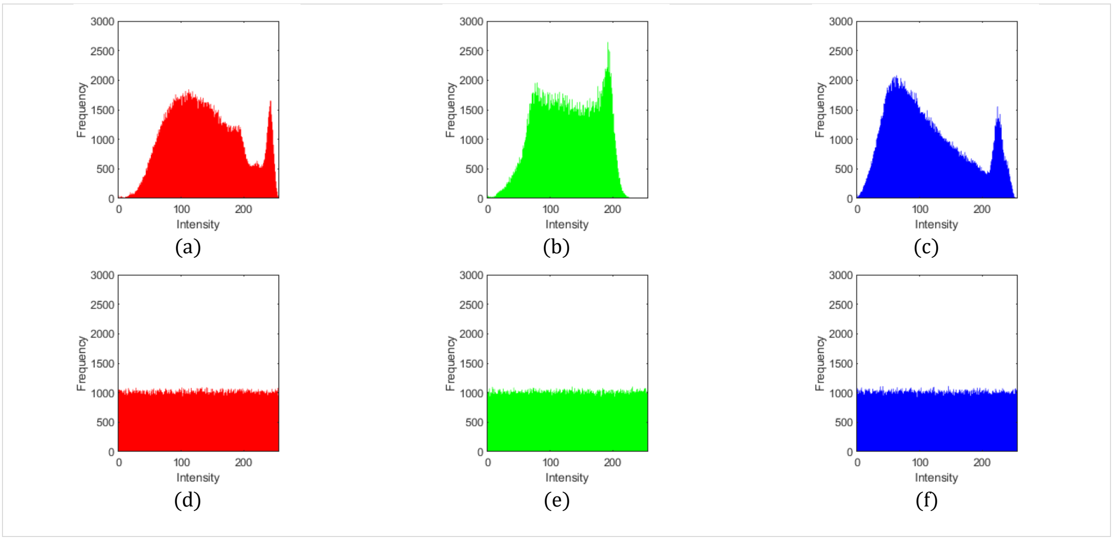


Fig. 11. Histogram analysis. (a)-(c) histogram plots of the red, green, and blue channels for the plain image of Baboon. (d)-(f) histogram plots of the red, green, and blue channels for the cipher image of Baboon (512 × 512).

Furthermore, the chi-square (x^2) test is utilized to statistically evaluate the uniformity of the pixel value distribution in the cryptographic images and can be described mathematically by the equation below [20]:

$$x^2 = \sum_{j=0}^{255} \frac{(O_j - \varepsilon_j)^2}{\varepsilon_j} \tag{37}$$

where j signifies the pixel value, O_j is the observed frequency, and ε_j is the expected frequency and can be formulated as follows:

$$\varepsilon_j = \frac{u \times n}{256} \tag{38}$$

where u and n are the color image dimensions. The threshold value of x^2 for significance levels of 1 % and 5 % is $x^2{}_{0.01}$ = 310.4574 and $x^2{}_{0.05}$ = 293.2478 to determine whether to pass the x^2 test for uniformity [20]. A reduced x^2 value reflects a more even distribution of pixel values in the encrypted images [7]. Table VIII displays the results of the x^2 test on the various encrypted images, with all the x^2 values less than the thresholds of 293.2478 and 310.4574, indicating that the PMIE-MLP investigates the uniformity distribution of pixel values and thus withstands statistical attacks.

TABLE VIII. CHI-SQUARE VALUES FOR VARIOUS CIPHER IMAGES

Cipher image	**Red**	**Green**	**Blue**	**Average**	**Results**
Tree 256	239.3203	247.9609	255.7266	247.6693	Pass
Lenna 256	228.9297	231.8750	221.9297	227.5781	Pass
Peppers 512	247.4414	192.2930	222.1543	220.6296	Pass
Baboon 512	242.4180	229.3496	248.0996	239.9557	Pass

Moreover, Table IX displays a comparison of the x^2 values with those of the most recent studies that used the x^2 test to assess uniformity. Obviously, the x^2 values in this study are markedly lower than those in other studies. This reveals that the PMIE-MLP algorithm yields a more uniform pixel distribution and exhibits an excellent ability to randomize pixel values relative to prior methodologies. Consequently, our strategy diminishes the likelihood of detecting patterns in the ciphered images, indicating better security performance.

TABLE IX. COMPARISON OF THE AVERAGE CHI-SQUARE VALUES

Cipher image	**PMIE-MLP**	**Ref.** [11]	**Ref.** [12]	**Ref.** [14]	**Ref.**[6]	Ref. [15]
Tree 256	247.6693	NaN	283.8233	NaN	**231.3359**	NaN
Lenna 256	**227.5781**	NaN	248.7867	NaN	NaN	289
Lenna 512	**244.0605**	NaN	NaN	283.2109	NaN	NaN
Peppers 512	**220.6296**	257.3021	227.4600	256.1543	NaN	NaN
Baboon 512	**239.9557**	352.3131	NaN	279.7891	NaN	NaN
Peppers 256	**250.7318**	NaN	NaN	NaN	257.8906	NaN
Baboon 256	**246.8281**	NaN	NaN	NaN	247.3646	NaN
Fruits 512×480	**249.3215**	NaN	271.7467	NaN	NaN	NaN
San Diego 1024×1024	**240.1935**	270.7210	NaN	NaN	NaN	NaN

4.3 Analysis of the correlation coefficients

The correlation coefficient (CC) test is a common metric that is used to compute the relationships among neighboring pixels across horizontal, vertical, and diagonal orientations. The plaintext images have significant correlations among nearby pixels, rendering them susceptible to statistical attacks. A successful encryption technique must disrupt the correlation between adjacent pixels; a higher correlation diminishes the algorithm's efficacy. The CC between pixels for both plaintext and cipher images can be computed via Equations (39–42) [3].

$$\mathrm{M}\ (\mathrm{x}) = \frac{1}{\mathrm{NP}} \sum_{\mathrm{i}=1}^{\mathrm{NP}} \mathrm{x_i} \tag{39}$$

$$\mathrm{V}\ (\mathrm{x}) = \frac{1}{\mathrm{NP}} \sum_{\mathrm{i}=1}^{\mathrm{NP}} (\mathrm{x_i} - \mathrm{M}\ (\mathrm{x}))^2 \tag{40}$$

$$\mathrm{C}(\mathrm{x},\mathrm{y}) = \frac{1}{\mathrm{NP}} \sum_{\mathrm{i}=1}^{\mathrm{NP}} ((\mathrm{x_i} - \mathrm{M}\ (\mathrm{x})) \times (\mathrm{y_i} - \mathrm{M}\ (\mathrm{y}))) \tag{41}$$

$$\mathrm{CC_{xy}} = \frac{\mathrm{C}\ (\mathrm{x},\mathrm{y})}{\sqrt{\mathrm{V}(\mathrm{x})} \times \sqrt{\mathrm{V}(\mathrm{y})}} \tag{42}$$

where x_i and y_i are the values of two neighboring pixels; M (x) is the mean; V (x) is the variance; $C(x,y)$ is the covariance; and NP is the total number of pixels in the image. When the CC value is close to 1 or -1, the correlation between neighboring pixels is strong. Conversely, if the CC value is close to zero, the relationship between neighboring pixels is weak.

Figure 12 shows the CC between neighboring pixels of the plaintext and cipher images for Lenna256×256 in all directions. The distributions of neighboring pixels in the plaintext image clearly have a concentrated shape, whereas the distributions in the encrypted image are more random and uniform, demonstrating the efficiency of the scrambling methods used in this paper, such as 3D-AS.

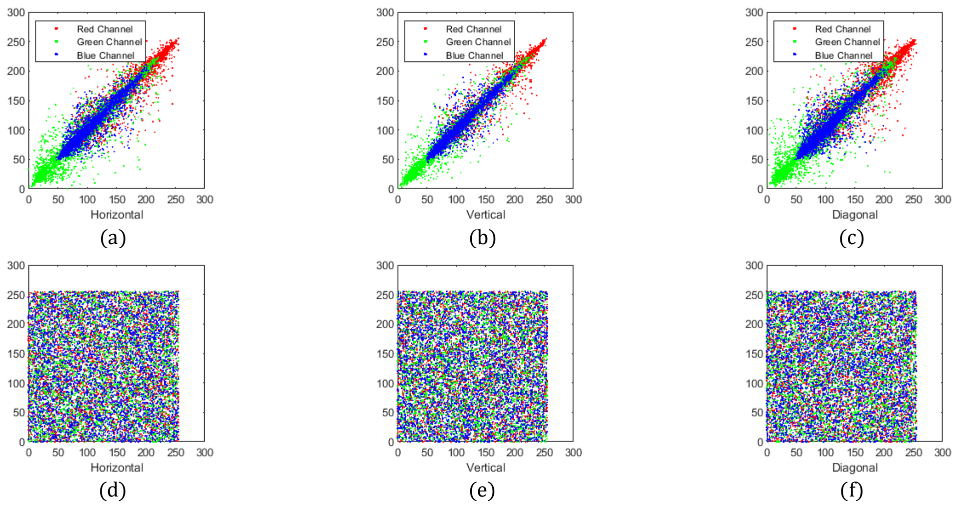


Fig. 12. Correlation coefficient plots. (a)-(c) Horizontal, vertical, and diagonal correlation, respectively, of the three channels for the plain image of Lenna. (d)-(f) Horizontal, vertical, and diagonal correlation, respectively, of the three channels for the cipher image of Lenna.

TABLE X. CC VALUES FOR VARIOUS PLAINTEXT AND CIPHER IMAGES

Image	**Color**	**Plaintext image**			**Cipher image**		
		H	**V**	**D**	**H**	**V**	**D**
Tree 256	R	0.958993	0.936077	0.915908	0.000017	0.006223	0.002231
	G	0.968698	0.945745	0.931768	0.002055	– 0.001861	0.000226
	B	0.961233	0.940562	0.926500	– 0.002935	0.004891	0.000372
Lenna 256	R	0.957216	0.978889	0.933884	0.000114	– 0.000176	– 0.006901
	G	0.943203	0.971369	0.919305	0.002168	– 0.001067	– 0.001003
	B	0.928443	0.955931	0.900678	0.003415	– 0.001035	0.001366
Peppers 512	R	0.963525	0.966337	0.956377	0.000860	0.002048	– 0.001777
	G	0.981118	0.981774	0.968658	– 0.000177	– 0.000822	– 0.002614
	B	0.966517	0.966425	0.947794	0.003410	0.000456	0.000509
Baboon 512	R	0.923066	0.865959	0.854341	– 0.002348	0.000225	– 0.002606
	G	0.865479	0.765007	0.734795	– 0.001678	– 0.000735	– 0.001566
	B	0.907344	0.880892	0.839855	0.000772	0.001359	– 0.001031

More precisely, Table X shows the values of the CC between adjacent pixels of different plaintexts and ciphered images in three directions and for each colored channel. According to the test results, the CC values for the three channels in three directions for plaintext images are all very close to one, whereas the CC values for encrypted images are very near zero. This shows that the PMIE-MLP works well at eliminating the strong correlation between pixels that are next to each other, which effectively stops statistical attacks.

Table XI compares the CC of different color images of the PMIE-MLP to some recent and excellent studies. Table XI calculates the horizontal CC of all images by calculating the average absolute horizontal correlation coefficient of the three channels [10]; it also calculates the vertical and diagonal CC in the same manner.

Table XI clearly shows that the PMIE-MLP has a lower average CC between adjacent pixels across different image sizes and types. This is a basic criterion for secure encryption, indicating that the PMIE-MLP can effectively and significantly resist statistical analysis assaults since a lower statistical correlation resists security threats successfully.

TABLE XI. COMPARISON OF AVERAGE CC VALUES FOR VARIOUS CIPHER IMAGES

Cipher image	Algorithm	Average value			
		Horizontal	Vertical	Diagonal	Average
Tree 256	PMIE-MLP	**0.001669**	**0.004325**	**0.000943**	**0.002313**
	Ref. [12]	0.005952	0.007854	0.009093	0.007633
	Ref. [6]	0.012633	0.005567	0.010967	0.009722
Lenna 256	PMIE-MLP	0.001899	**0.000759**	0.003090	**0.001916**
	Ref. [12]	0.027389	0.008922	0.004550	0.013620
	Ref. [15]	0.001799	0.002613	0.002387	0.002266
	Ref. [17]	**0.000913**	0.004870	**0.001780**	0.002521
Lenna 512	PMIE-MLP	**0.000511**	**0.000939**	0.001751	**0.001067**
	Ref. [10]	0.0009	0.0019	**0.0009**	0.001233
	Ref. [14]	0.010333	0.015233	0.011833	0.012466
	Ref. [8]	0.002800	0.005700	0.003700	0.004067
Peppers 512	PMIE-MLP	**0.001482**	**0.001109**	**0.001633**	**0.001408**
	Ref. [11]	0.004100	0.003167	0.003000	0.003422
	Ref. [12]	0.010380	0.004716	0.013437	0.009511
	Ref. [14]	0.004833	0.019400	0.006367	0.010200
	Ref. [8]	0.006767	0.004533	0.004567	0.005289
Baboon 512	PMIE-MLP	**0.001599**	**0.000773**	0.001735	**0.001369**
	Ref. [11]	0.002967	0.004200	0.003267	0.003478
	Ref. [12]	0.006587	0.010245	0.006484	0.007772
	Ref. [14]	0.013633	0.020400	0.019067	0.017700
	Ref. [16]	0.0063	0.0156	0.0072	0.009700
	Ref. [8]	0.003233	0.005033	**0.001333**	0.003200
Peppers 256	PMIE-MLP	0.003864	**0.003070**	**0.002818**	**0.003250**
	Ref. [6]	**0.003100**	0.009767	0.003900	0.005589
Baboon 256	PMIE-MLP	0.003064	**0.003778**	**0.001359**	**0.002734**
	Ref. [6]	**0.001767**	0.008700	0.008667	0.006378
Fruits 512×480	PMIE-MLP	**0.000963**	**0.001076**	**0.001418**	**0.001152**
	Ref. [12]	0.015292	0.011114	0.012482	0.012963
San Diego 1024×1024	PMIE-MLP	**0.000548**	**0.001072**	**0.000716**	**0.000779**
	Ref. [11]	0.002000	0.003933	0.002567	0.002833

4.4 Analysis of the Entropy

It is a crucial metric that primarily assesses the randomness and unpredictability of data and frequently gauges the level of disorder in images. The information entropy value is determined by determining how spread out the pixels are in the three channels of the image. The higher the value is, the more evenly distributed the pixels are, the more random the data are, and the more robust the data are against statistical attacks. The information entropy value for both plaintext and cipher images can be formulated via Equation (43) [29].

$$E(x) = -\sum_{i=0}^{255} P(x_i) \log_2 P(x_i) \tag{43}$$

where $E(x)$ is the information entropy and where $P(x_i)$ signifies the probability that pixel x_i is present. To withstand the statistical attack, the cipher images' entropy value should approximate an ideal value of 8 [33].

TABLE XII. INFORMATION ENTROPY VALUES FOR VARIOUS PLAINTEXT AND CIPHER COLOR IMAGES

Image	Plaintext image			Cipher image		
	Red	Green	Blue	Red	Green	Blue
Tree 256	7.210437	7.413611	6.920742	7.9974	7.9973	7.9972
Lenna 256	7.241727	7.576708	6.917058	7.9975	7.9974	7.9976
Peppers 512	7.338827	7.496253	7.058306	7.9993	7.9995	7.9994
Baboon 512	7.706672	7.474432	7.752217	7.9993	7.9994	7.9993

Table XII explains the values of entropy for the three channels of the assorted plaintext and cipher images of varying dimensions. As shown in Table XII, the average entropy value for each encrypted image exceeds 7.9972 and 7.9992, respectively, and is very close to the ideal value of 8. This finding indicates that the PMIE-MLP effectively randomizes the distribution of the pixels of the original image, rendering it infeasible for an attacker to extract any information regarding the original image from its encrypted counterpart. Moreover, Table XIII compares the average entropy values for the various cipher images used in this research, showing that the PMIE-MLP produces average entropy values that are either somewhat higher or quite similar to those in some recent studies. proving that the PMIE-MLP exhibits competitive performance in comparison to other approaches, achieving almost ideal entropy values, suggesting that it can withstand statistical attacks.

TABLE XIII. COMPARISON OF AVERAGE ENTROPY VALUES FOR VARIOUS CIPHER IMAGES

Cipher image	Algorithm	Average
Lenna 256	PMIE-MLP	**7.9975**
	Ref. [12]	7.9973
	Ref. [15]	7.9970
	Ref. [17]	7.9966
	Ref. [18]	7.9895
Lenna 512	PMIE-MLP	**7.9993**
	Ref. [10]	7.9992
	Ref. [13]	**7.9993**
	Ref. [14]	**7.9993**
	Ref. [8]	7.9930
Peppers 512	PMIE-MLP	**7.9994**
	Ref. [11]	7.9993
	Ref. [12]	7.9992
	Ref. [14]	7.9993
	Ref. [8]	7.9990
Baboon 512	PMIE-MLP	**7.9993**
	Ref. [11]	**7.9993**
	Ref. [12]	**7.9993**
	Ref. [14]	7.9992
	Ref. [16]	7.9992
	Ref. [8]	7.9970
Peppers 256	PMIE-MLP	**7.9972**
	Ref. [15]	7.9967
	Ref. [17]	**7.9972**
	Ref. [6]	7.9971
Baboon 256	PMIE-MLP	**7.9973**
	Ref. [15]	7.9969
	Ref. [17]	7.9965
	Ref. [6]	**7.9973**
Fruits 512×480	PMIE-MLP	**7.9993**
	Ref. [12]	7.9992
San Diego 1024×1024	PMIE-MLP	**7.9998**
	Ref. [11]	**7.9998**

4.5 Analysis of Differential Attack

Differential attacks involve assessing the security and resistance of encryption algorithms to differential attacks. This type of analysis involves making few alterations to the original image and comparing the resulting ciphered images before and after the alterations. The aim is to determine whether an attacker can find the relationship between the plaintext image and the encrypted image through comparison and analysis [13][34]. For a secure encryption algorithm, a one-bit change in pixels between two plaintext images leads to a significant alteration between the two encrypted images, E_1 and E_2. The number of pixels change rate (NPCR) and the unified average changing intensity (UACI) indicate how well a system can withstand a differential attack. The NPCR and UACI formulas are defined by Equations (44–46) [33].

$$D(i,j) = \begin{cases} 0, & E_1(i,j) = E_2(i,j) \\ 1, & E_1(i,j) \neq E_2(i,j) \end{cases} \tag{44}$$

$$\mathrm{NPCR}(E_1, E_2) = \frac{1}{u \times n} \sum_{i=1}^{u} \sum_{j=1}^{n} D(i,j) \times 100\% \tag{45}$$

$$\mathrm{UACI}(E_1, E_2) = \frac{1}{u \times n} \left(\sum_{i=1}^{u} \sum_{j=1}^{n} \frac{|E_1(i,j) - E_2(i,j)|}{255} \times 100\% \right) \tag{46}$$

where u and n denote the rows and columns of the image, respectively; E_1 denotes the original cipher image; and E_2 denotes the cipher image after changing only one bit pixel value in the plain image. The best values for the NPCR and UACI are 99.6094 % and 33.4635 %, respectively [20].

TABLE XIV. NPCR (%) AND UACI (%) VALUES FOR VARIOUS CIPHERED IMAGES

	NPCR (%)				**UACI (%)**			
Image	Red	Green	Blue	Average	Red	Green	Blue	Average
Tree 256	99.6231	99.6140	99.6582	99.6318	33.5739	33.5131	33.3626	33.4832
Lenna 256	99.6170	99.6445	99.5758	99.6124	33.4217	33.5807	33.5432	33.5152
Peppers 512	99.6307	99.6189	99.6246	99.6248	33.4398	33.5703	33.5543	33.5214
Baboon 512	99.6349	99.6212	99.6243	99.6268	33.4759	33.4627	33.5580	33.4989
Lenna 512×512	99.6216	99.6201	99.6136	99.6184	33.5302	33.5490	33.4585	33.5126
Peppers 256×256	99.6155	99.6384	99.5941	99.6160	33.4030	33.6282	33.4158	33.4823
Baboon 256×256	99.6094	99.6017	99.6582	99.6231	33.6430	33.4183	33.4991	33.5201
House 512×512	99.5968	99.6155	99.6323	99.6148	33.4467	33.5011	33.5722	33.5067
Fruits 512×480	99.6285	99.6269	99.6175	99.6243	33.4814	33.4942	33.4312	33.4689
Boats 787×576	99.6120	99.6140	99.6228	99.6162	33.4607	33.5164	33.4656	33.4809
San Diego 1024	99.6077	99.6233	99.6065	99.6125	33.4748	33.4819	33.4519	33.4696
Average				99.6201				33.4963

To test against differential attacks, we selected a set of different color images, assigned a random location from one of the three channels, changed the pixel value of that location by one bit, and then encrypted this modified plaintext image via PMIE-MLP. We then calculate the results of the UACI and NPCR for two types of encrypted images, as presented in Table XIV. As we can see from Table XIV, all values of the UACI and NPCR are higher than the best values, and the average values of the UACI and NPCR for all cipher images are 99.6201 and 33.4963, respectively, indicating that the PMIE-MLP is highly sensitive to a single bit difference in the original image and is very resistant to differential attacks. Furthermore, Table XV lists the performance comparisons of the average values of the UACI and NPCR of the PMIE-MLP and some other studies on various cipher images. According to Table XV, the average NPCR values of the PMIE-MLP for all cipher images are consistently approximately 99.61 %, slightly greater than or matching those of the other studies and higher than the ideal value of 99.6094 %. Moreover, the average UACI values for the PMIE-MLP are predominantly near 33.5 %, marginally surpassing those of other studies and higher than the best value of 33.4635 %. This suggests a powerful diffusion impact in the ciphered images and demonstrates sufficient strength against differential attacks for diverse image sizes.

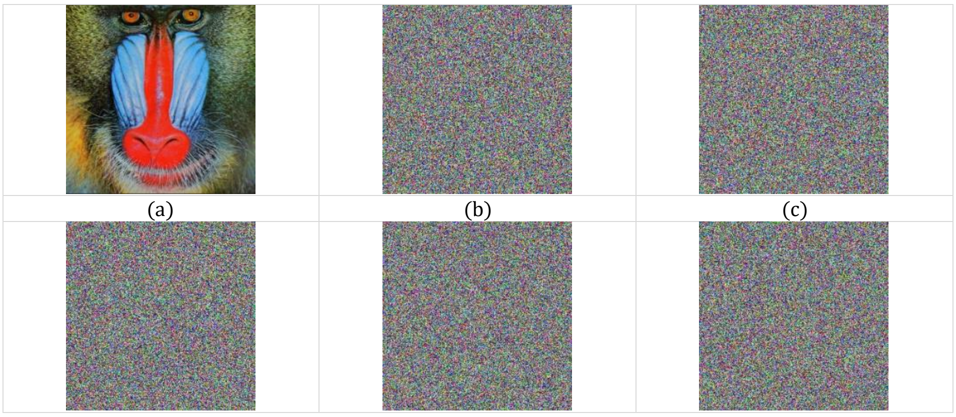
TABLE XV. COMPARISON OF THE AVERAGE NPCR AND UACI VALUES FOR VARIOUS CIPHER IMAGES

Cipher image	**Algorithm**	**Average NPCR (%)**	**Average UACI (%)**
Tree 256	PMIE-MLP	**99.6318**	**33.4832**
	Ref. [12]	99.6231	33.3059
	Ref. [6]	99.6068	33.4564
Lenna 512	PMIE-MLP	99.6184	**33.5126**
	Ref. [10]	99.6189	33.4501
	Ref. [14]	**99.6195**	33.4312
Peppers 512	PMIE-MLP	**99.6248**	**33.5214**
	Ref. [11]	99.6157	33.4632

	Ref. [12]	99.6036	33.4629
	Ref. [13]	99.6124	33.4655
	Ref. [14]	99.5984	33.3374
Baboon 512	PMIE-MLP	**99.6268**	33.4989
	Ref. [11]	99.6172	33.4663
	Ref. [12]	99.6078	33.4557
	Ref. [13]	99.5994	33.4649
	Ref. [14]	99.6181	**33.5120**
	Ref. [16]	99.6046	33.4936
Lenna 256	PMIE-MLP	99.6124	**33.5152**
	Ref. [15]	99.6198	32.3493
	Ref. [17]	**99.6254**	30.5681
Peppers 256	PMIE-MLP	**99.6160**	**33.4823**
	Ref. [15]	99.5271	33.4173
	Ref. [6]	99.5946	33.4464
Baboon 256	PMIE-MLP	99.6231	33.5201
	Ref. [15]	99.5880	**33.5620**
	Ref. [6]	**99.6277**	33.5293
Fruits 512×480	PMIE-MLP	**99.6243**	**33.4689**
	Ref. [12]	99.6132	33.4381
San Diego 1024×1024	PMIE-MLP	99.6125	**33.4696**
	Ref. [11]	**99.6137**	33.4656

4.6 Analysis of the Key's Sensitivity

A secure image encryption technique should demonstrate a high level of sensitivity to the secret key, meaning that even a slight alteration in the key would result in complete distortion in the ciphered image. To test the sensitivity of the baboon (256 × 256) image to the key, we use some of the initial parameters (x_0 = 0.780395507812500, y_0 = 0.384399414062500, z_0 = 0.753051757812500, μ = 3.722665637060787, δ = 0.018781081662647, β = 0.013885550641593, M = 0.025, K = 0.025, s1 = 5.8, s2 = 0.825, f1 = 1.85, f2 = 10, a_0 = 1, b_0 = 3, c_0 = 1, d_0 = – 0.7, and MSK = 743B5A203B1F8EDF6C0FB0D7497CB2E228689AD00F57F8953B5C6127E1C26053) as an example, leaving the rest of the keys unchanged. Figure 13 (a) illustrates the ciphered image with the correct initial values of the keys. Figure 13 (b)-(f) shows the results of decrypting the Baboon image with slightly different initial values of the keys. The slight difference in some of the initial values of the keys is as follows: (b) y_0 = 0.38439941406250**1**; (c) β = 0.01388555064159**4**; (d) M = 0.02**6**; (e) s1=5.**9**; and (f) MSK = 743B5A203B1F8EDF6C0FB0D7497CB2E228689AD**1**0F57F8953B5C6127E1C26053. After the test results are analysed, the plaintext image can be recovered only through the original key used in the encryption process. This demonstrates that the PMIE-MLP is extremely sensitive to the key.



(d)	(e)	(f)

Fig. 13. Decrypted Baboon images. (a) Decrypted image with the correct key. (b)-(f) Decrypted images with a tiny difference in the initial values of the keys.

Additionally, the NPCR can be utilized to assess the key's sensitivity by detecting the discrepancies between two decrypted images generated by distinct keys. Table XVI clearly demonstrates that all average differences surpass 99.6047 %, indicating that the PMIE-MLP is highly sensitive to the key. The high values of the NPCR indicate that a single-pixel modification in the original image causes widespread changes in the encrypted image, ensuring strong key sensitivity.

TABLE XVI. THE NPCR BETWEEN TWO DECRYPTED IMAGES WITH TWO DIFFERENT KEYS

Channel	(a)- (a)	(a)- (b)	(a)- (c)	(a)- (d)	(a)- (e)	(a)- (f)
Red	0 %	99.6063 %	99.6033 %	99.6002 %	99.5880 %	99.6246 %
Green	0 %	99.6338 %	99.5667 %	99.6033 %	99.6338 %	99.6490 %
Blue	0 %	99.6185 %	99.6475 %	99.6109 %	99.6399 %	99.6552 %
Average	0 %	99.6195 %	99.6058 %	99.6048 %	99.6206 %	99.6429 %

4.7 Analysis of the Quality Measure

From the concept of image encryption, numerous statistical attacks aim to compromise the integrity of the encrypted image's quality. Consequently, standard metrics are utilized to evaluate the quality comparison between the plaintext and encrypted images [20].

4.7.1 Mean Square Error (MSE)

The MSE assesses the performance of the encryption system and is computed between the plaintext and the ciphered image via Equation (47). Elevated MSE values signify an effective encryption system [35].

$$\mathrm{MSE} = \frac{1}{\mathrm{u} \times \mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{u}} \sum_{\mathrm{j}=1}^{\mathrm{n}} \left(\mathrm{P}(\mathrm{i},\mathrm{j}) - \mathrm{E}(\mathrm{i},\mathrm{j})\right)^2 \tag{47}$$

where u and n are the rows and columns of the image, respectively; $\mathrm{P}(\mathrm{i},\mathrm{j})$ is the plaintext image; and $\mathrm{E}(\mathrm{i},\mathrm{j})$ is the cipher image.

4.7.2 Peak signal-to-noise ratio (PSNR)

The PSNR is another metric that relies primarily on the MSE. It is also calculated between the plaintext and encrypted image. The quality of the encryption system is high when the PSNR value is less than 10 dB between the plaintext and encrypted images, and it can be calculated mathematically via equation (48) [35].

$$\mathrm{PSNR} = 10 \log_{10}\left(\frac{\mathrm{P}_{\mathrm{max}}^2}{\mathrm{MSE}}\right) \tag{48}$$

where $\mathrm{P}_{\mathrm{max}}$ is the highest pixel value, specifically 255.
Table XVII shows the MSE and PSNR values for various images, where the result of each quality measure between the plaintext and the encrypted image is the average of the three color channels.

TABLE XVII. THE AVERAGE MSE AND PSNR VALUES

Image	MSE	PSNR (dB)
Tree 256	9958.0310	8.1491
Lenna 256	8868.1628	8.6525
Peppers 512	10131.502	8.0741
Baboon 512	8615.7119	8.7779

The data presented in Table XVII indicate that the average MSE values are relatively elevated, whereas the average PSNR values are notably low, falling below 10 dB. The encrypted image exhibits considerable distortion, indicating the efficacy of the PMIE-MLP in safeguarding image data. Additionally, Table XVIII presents comparisons with some recent studies that report MSE and PSNR values. It is evident that the average values of MSE and PSNR calculated for titled images via the suggested approach are either better or on par with results from the references [15] and [17].

TABLE XVIII. COMPARISON OF THE AVERAGE MSE AND PSNR VALUES FOR VARIOUS CIPHER IMAGES

Image	Algorithm	Average MSE	Average PSNR (dB)
Lenna 256	PMIE-MLP	8868.1628	8.6525
	Ref. [15]	8888.8821	8.7051
	Ref. [17]	**8983.7886**	**8.6510**
Peppers 256	PMIE-MLP	10054.3810	**8.1073**
	Ref. [15]	**10092.3268**	8.1400
	Ref. [17]	10033.0116	8.1624
Baboon 256	PMIE-MLP	**9517.2557**	**8.3457**
	Ref. [15]	8295.2068	8.9575
	Ref. [17]	8349.5506	8.9283

4.8 Analysis of the robustness

Robustness is a critical characteristic denoting an encryption algorithm's capacity to withstand interference. Images are vulnerable to noise or loss during transmission. Consequently, an effective encryption approach must demonstrate a degree of resilience to withstand interference on cipher images, facilitating the recovery of plain images with little modification [36]. We assess the resilience of our method against noise and occlusion assaults.

4.8.1 Analysis of the noise attack

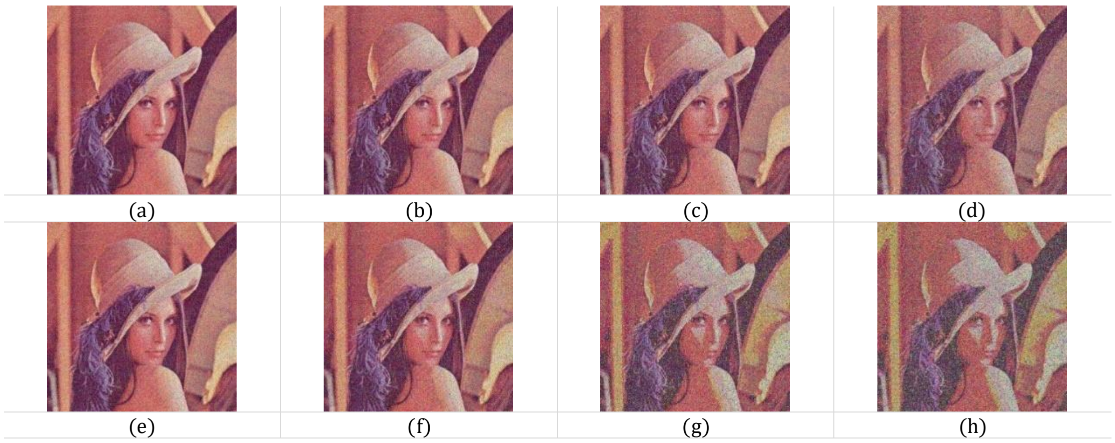


Fig. 14. Noise attack test. (a)-(d) Decrypted images subsequent to the introduction of Salt & Pepper noise at densities of 0.01, 0.05, 0.10, and 0.20, respectively, to the encrypted images. (e)-(h) Decrypted images subsequent to the introduction of Gaussian noise with variances of 0.0001, 0.001, 0.01, and 0.02, respectively, to the encrypted images.

The color "Lenna" image with a size of 512 × 512 is selected as the plaintext PMIE-MLP. Salt and pepper noises with densities of 0.01, 0.05, 0.10, and 0.20 and Gaussian noises with variances of 0.0001, 0.001, 0.01, and 0.02 are added to the encrypted Lenna images. The PMIE-MLP is subsequently utilized for deciphering these attacked images, and the test results are shown in Figure 14. Figure 14 clearly demonstrates that when the encrypted image encounters noise attacks, such as salt and pepper noise and Gaussian noise, a significant portion of the original image's information can be effectively recovered, indicating that the PMIE-MLP exhibits strong robustness.

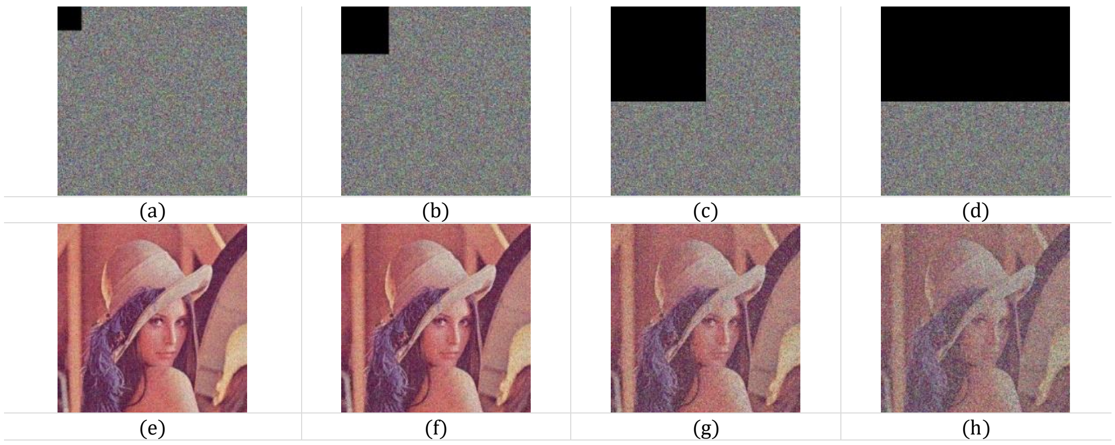


Fig. 15. Occlusion attack test. (a)-(d) Cropped encrypted images of Lenna by eliminating blocks of dimensions 64 × 64, 128 × 128, 256 × 256, and 512 × 256, respectively. (e)-(h) decrypted images of (a)–(d).

4.8.2 Analysis of the Occlusion Attack

Another type of test is to further assess the robustness of the given encryption approach. The cipher image of Lenna with a size of 512 × 512 is cropped by removing blocks of sizes 64 × 64, 128 × 128, 256 × 256, and 512 × 256, as explained in Figure 15 (a)-(d). As shown in Figure 15(e)-(h), all the decrypted images can still be recognized even at various levels of cropping attacks. Thus, our encryption algorithm exhibits robust resistance against cropping attacks.

4.9 Complexity and time analysis

Suppose that N = u × n is the total number of pixels. Most steps in the algorithm—such as generating sequences, channel separation, substitution, diffusion, XOR operations, and reshaping—are accomplished in linear time, i.e., O(N). However, the key generation step requires processing an array of size N, which has a complexity of O (N log N). Therefore, the overall computational complexity of the algorithm is dominated by this process, making it O (N log N).

Time analysis is used to assess the complexity of the PMIE-MLP and its appropriateness for real-time applications.

TABLE XIX. TIME ANALYSIS RESULTS

Image	Encryption Time (s)	Decryption Time (s)
Tree 128	0.8431	0.8456
Tree 256	3.4208	3.4147
Lenna 256	3.4137	3.4074
Peppers 256	3.4350	3.4130
Baboon 256	3.4188	3.4140
Lenna 512	14.8122	14.8189
Peppers 512	14.7718	14.7557
Baboon 512	14.8173	14.8322

Table XIX shows the encryption and decryption times for a set of images at different dimensions. The encryption time ranges from 0.8431 to 14.8173 s, whereas the decryption time ranges from 0.8456 to 14.8322 s, which is very efficient, especially for environments that require a high level of security.

Furthermore, Table XX presents a comparison of the encryption time between the PMIE-MLP and other related methods. Importantly, numerous factors influence the execution time, such as the algorithm's structure, the programming environment in which the algorithm is designed, and the specifications of the device executing the algorithm, such as memory and processing power. According to Table XX, the encryption time of the PMIE-MLP is slightly slower than that of the other devices because of the device's low specifications. Therefore, the encryption time of the PMIE-MLP algorithm can be optimized efficiently by developing device specifications.

TABLE XX. COMPARISON OF THE ENCRYPTION TIMES

Image	Algorithm	Encryption Time (s)	Machine Specifications (CPU and RAM)	Software Platform
Lenna 256 × 256	PMIE-MLP	3.4137	2.3 GHz Intel® Core™ i5, 4 GB	MATLAB (R2021a)
	Ref. [15]	2.582389	2.9 GHz Intel® Core™ i9, 32 GB	Wolfram Mathematica
	Ref. [17]	2.750966	3.4 GHz Intel® Core™ i7, 8 GB	NaN
	Ref. [12]	1.911	1.10 GHz Intel® Core™ i7, 32 GB	MATLAB (R2020b)
Peppers 256 × 256	PMIE-MLP	3.4350	2.3 GHz Intel® Core™ i5, 4 GB	MATLAB (R2021a)
	Ref. [6]	1.1677	3.6 GHz Intel® Core™ i9, 32 GB	MATLAB (R2022b)

5. CONCLUSION

This research presents a novel image cryptographic system for enhancing the security of images, which uses a dynamic Mealy machine-based protein sequence, a novel 3D-AS scrambling, and a chaotic system. An encryption structure consists of key generation and six stages of protection: substitution, four layers of diffusion, and confusion. To achieve the greatest possible unpredictability and increase the efficiency of the PMIE-MLP, the pixel positions in the image are shuffled more than once (four layers of diffusion) without changing its value, and the image becomes unknowable. To support the proposition of the theoretical framework, various types of experimental analyses and security assessments are conducted. The experimental results presented in this research demonstrate that the key space used in the PMIE-MLP is (2^{1488}); therefore, it is evident that this algorithm presents a very high level of security for brute force attacks. The visualization of cipher images is shown by the histogram of image pixels with equal distributions, and the values of the correlation coefficient are almost zero for the encrypted images, which indicates that the PMIE-MLP does not allow the attacker to launch a statistical attack. The average entropy for the cipher images is very close to the ideal entropy value, which ensures the effectiveness of the PMIE-MLP in randomizing the pixel distribution of the original image so that the attacker cannot obtain any information from the encrypted image. The PMIE-MLP has very high sensitivity to the plaintext image and to the key, which should have enough immunity to differential attacks. The encryption algorithm is vulnerable to most attack types, including occlusion attacks and noise attacks. In addition, when all the PMIE-MLP metrics are compared, the security performance of the PMIE-MLP is equivalent to or even greater than that reported in prior studies. The analysis of the test results demonstrates that the superior performance is illustrated by the parameters of the key space, chi-square values, correlation coefficient values, and image resistance to differential attacks; therefore, the offered PMIE-MLP encryption algorithm serves practical uses in real-world security through its protective functions for image communications across medical imaging systems, military networks and secure storage systems.
While the PMIE-MLP algorithm demonstrates strong security performance, several factors require further consideration. The compatibility of the encryption process with common image compression techniques needs to be explored. The study does not explicitly address potential limitations, such as its suitability for resource-constrained environments, energy consumption, or computational complexity. Additionally, future research could focus on optimizing the execution speed, improving resistance against new attack vectors, or adapting the method for real-time applications in constrained devices.

Conflicts of interest

The authors of this work report no conflicts of interest.

Funding

No funding.

References

[1] R. M. Al-Amri, D. N. Hamood, and A. K. Farhan, “Theoretical Background of Cryptography,” *Mesopotamian J. CyberSecurity*, vol. 2023, pp. 7–15, 2023, doi: 10.58496/MJCS/2023/002.

[2] R. R. N. Alogaili *et al.*, “AntDroidNet Cybersecurity Model: A Hybrid Integration of Ant Colony Optimization and Deep Neural Networks for Android Malware Detection,” *Mesopotamian J. CyberSecurity*, vol. 5, no. 1, pp. 104–120, 2025, doi: 10.58496/MJCS/2025/008.

[3] X. Wei, L. Guo, Q. Zhang, J. Zhang, and S. Lian, “A novel color image encryption algorithm based on DNA sequence operation and hyper-chaotic system,” *J. Syst. Softw.*, vol. 85, no. 2, pp. 290–299, 2012, doi: 10.1016/j.jss.2011.08.017.

[4] C. Zhu, Z. Gan, Y. Lu, and X. Chai, “An image encryption algorithm based on 3-D DNA level permutation and substitution scheme,” *Multimed. Tools Appl.*, vol. 79, no. 11, pp. 7227–7258, 2020.

[5] X. Liu, X. Tong, M. Zhang, and Z. Wang, “A highly secure image encryption algorithm based on conservative hyperchaotic system and dynamic biogenetic gene algorithms,” *Chaos, Solitons and Fractals*, vol. 171, no. February, p. 113450, 2023, doi: 10.1016/j.chaos.2023.113450.

[6] Q. Wang, X. Zhang, and X. Zhao, “Color image encryption algorithm based on bidirectional spiral transformation and DNA coding,” *Phys. Scr.*, vol. 98, no. 2, 2023, doi: 10.1088/1402-4896/acb322.

[7] L. Huang, S. Wang, J. Xiang, and Y. Sun, “Chaotic Color Image Encryption Scheme Using Deoxyribonucleic Acid (DNA) Coding Calculations and Arithmetic over the Galois Field,” *Math. Probl. Eng.*, vol. 2020, 2020, doi: 10.1155/2020/3965281.

[8] Q. Li and L. Chen, “An image encryption algorithm based on 6-dimensional hyper chaotic system and DNA encoding,” *Multimed. Tools Appl.*, vol. 83, no. 2, pp. 5351–5368, 2024, doi: 10.1007/s11042-023-15550-3.

[9] A. S. Almasoud, B. Alabduallah, H. Alqahtani, S. S. Aljameel, S. S. Alotaibi, and A. Mohamed, “Chaotic image encryption algorithm with improved bonobo optimizer and DNA coding for enhanced security,” *Heliyon*, vol. 10, no. 3, p. e25257, 2024, doi: 10.1016/j.heliyon.2024.e25257.

[10] Q. Cun, X. Tong, Z. Wang, and M. Zhang, “A new chaotic image encryption algorithm based on dynamic DNA coding and RNA computing,” *Vis. Comput.*, vol. 39, no. 12, pp. 6589–6608, 2023, doi: 10.1007/s00371-022-02750-5.

[11] X. Gao, B. Sun, Y. Cao, S. Banerjee, and J. Mou, “A color image encryption algorithm based on hyperchaotic map and DNA mutation,” *Chinese Phys. B*, vol. 32, no. 3, 2023, doi: 10.1088/1674-1056/ac8cdf.

[12] F. Meng and Z. Gu, “A Color Image-Encryption Algorithm Using Extended DNA Coding and Zig-Zag Transform Based on a Fractional-Order Laser System,” *Fractal Fract.*, vol. 7, no. 11, 2023, doi: 10.3390/fractalfract7110795.

[13] H. Lu, L. Teng, and L. Du, “Image encryption with 1D-MS chaotic systems and improved zigzag disambiguation,” *Eur. Phys. J. Plus*, vol. 139, no. 4, pp. 1–16, 2024, doi: 10.1140/epjp/s13360-024-05146-7.

[14] M. Abdul-Hameed, H. El-Metwally, S. Askar, A. M. Alshamrani, M. Abouhawwash, and A. A. Karawia, “Advanced color image encryption using third-order differential equations and three-dimensional logistic map,” *AIP Adv.*, vol. 14, no. 7, 2024, doi: 10.1063/5.0214794.

[15] W. Alexan, M. ElBeltagy, and A. Aboshousha, “Rgb image encryption through cellular automata, s-box and the lorenz system,” *Symmetry (Basel).*, vol. 14, no. 3, p. 443, 2022.

[16] T. Zhang and S. Wang, “Image encryption scheme based on a controlled zigzag transform and bit-level encryption under the quantum walk,” *Front. Phys.*, vol. 10, pp. 1–12, 2023, doi: 10.3389/fphy.2022.1097754.

[17] W. Alexan, M. Elkandoz, M. Mashaly, E. Azab, and A. Aboshousha, “Color Image Encryption Through Chaos and KAA Map,” *IEEE Access*, vol. 11, no. October 2022, pp. 11541–11554, 2023, doi: 10.1109/ACCESS.2023.3242311.

[18] X. Wang, X. Zhang, M. Gao, Y. Tian, C. Wang, and H. H. C. Iu, “A Color Image Encryption Algorithm Based on Hash Table, Hilbert Curve and Hyper-Chaotic Synchronization,” *Mathematics*, vol. 11, no. 3, 2023, doi: 10.3390/math11030567.

[19] R. J. Kadhim and H. K. Khafaji, “Unprecedented Security Analysis Results for a Novel Steganography Approach Based on Protein Sequences,” *Int. J. Intell. Eng. Syst.*, vol. 16, no. 2, pp. 464–476, 2023, doi: 10.22266/ijies2023.0430.37.

[20] P. N. Lone, D. singh, and U. H. Mir, “Image encryption using DNA coding and three-dimensional chaotic systems,” *Multimed. Tools Appl.*, vol. 81, no. 4, pp. 5669–5693, 2022, doi: 10.1007/s11042-021-11802-2.

[21] P. N. Khade and P. M. Narnaware, “3D Chaotic Functions for Image Encryption,” *Int. J. Comput. Sci. Issues (IJCSI)*, vol. 9, no. 3, pp. 323–328, 2012.

[22] X. Ma, J. Mou, L. Xiong, S. Banerjee, Y. Cao, and J. Wang, “A novel chaotic circuit with coexistence of multiple attractors and state transition based on two memristors,” *Chaos, Solitons and Fractals*, vol. 152, p. 111363, 2021,

doi: 10.1016/j.chaos.2021.111363.

[23] P. Garapati and S. Musala, "Moore and Mealy Negative Edge detector A VHDL Example for Finite State Machine," *Proc. 2020 IEEE Int. Conf. Commun. Signal Process. ICCSP 2020*, pp. 1159–1161, 2020, doi: 10.1109/ICCSP48568.2020.9182310.

[24] A. Bhowmik, S. Karforma, and J. Dey, "Symmetric key and artificial neural network with mealy machine: A neoteric model of cryptosystem for cloud security," *Mach. Learn. Tech. Anal. Cloud Secur.*, pp. 81–101, 2021, doi: 10.1002/9781119764113.ch5.

[25] B. B. Kodada and D. A. D'Mello, "Symmetric Key Cryptosystem based on Sequential State Machine," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1187, no. 1, p. 012026, 2021, doi: 10.1088/1757-899x/1187/1/012026.

[26] P. Pavithran, S. Mathew, S. Namasudra, and P. Lorenz, "A novel cryptosystem based on DNA cryptography and randomly generated mealy machine," *Comput. Secur.*, vol. 104, p. 102160, 2021, doi: 10.1016/j.cose.2020.102160.

[27] P. Pavithran, S. Mathew, S. Namasudra, and A. Singh, "Enhancing randomness of the ciphertext generated by DNA-based cryptosystem and finite state machine," *Cluster Comput.*, vol. 26, no. 2, pp. 1035–1051, 2023, doi: 10.1007/s10586-022-03653-9.

[28] R. J. Kadhim, "Unprecedented Security Analysis Results for a Novel Key Expansion Algorithm Based on Protein Sequences , Dynamic Mealy Machine , and 3D Logistic Map," vol. 17, no. 3, pp. 171–188, 2024, doi: 10.22266/ijies2024.0630.15.

[29] W. Xingyuan, Z. Junjian, and C. Guanghui, "An image encryption algorithm based on ZigZag transform and LL compound chaotic system," *Optics and Laser Technology*, vol. 119. 2019. doi: 10.1016/j.optlastec.2019.105581.

[30] University of Southern California, "USC-SIPI Image Database", [Online]. Available: https://sipi.usc.edu/database/

[31] X. Chai, Z. Gan, K. Yuan, Y. Chen, and X. Liu, "A novel image encryption scheme based on DNA sequence operations and chaotic systems," *Neural Comput. Appl.*, vol. 31, no. 1, pp. 219–237, 2019, doi: 10.1007/s00521-017-2993-9.

[32] H. M. M. Alibraheemi, M. M. A. Al Ibraheemi, and Z. H. Radhy, "Design and Practical Implementation of a Stream Cipher Algorithm Based on a Lorenz System," *Mesopotamian J. CyberSecurity*, vol. 4, no. 3, pp. 136–151, 2024, doi: 10.58496/MJCS/2024/019.

[33] X. Yan, X. Wang, and Y. Xian, "Chaotic image encryption algorithm based on arithmetic sequence scrambling model and DNA encoding operation," *Multimed. Tools Appl.*, vol. 80, no. 7, pp. 10949–10983, 2021, doi: 10.1007/s11042-020-10218-8.

[34] H. Qiu, X. Xu, Z. Jiang, K. Sun, and C. Xiao,"A color image encryption algorithm based on hyperchaotic map and Rubik's Cube scrambling,"*Nonlinear Dyn.*,vol.110, no.3, pp. 2869–2887, 2022, doi: 10.1007/s11071-022-077561.

[35] Q. Lai and Y. Liu, "A cross-channel color image encryption algorithm using two-dimensional hyperchaotic map," *Expert Syst. Appl.*, vol. 223, no. November 2022, 2023, doi: 10.1016/j.eswa.2023.119923.

[36] C. Li, Y. Zhang, H. Li, and Y. Zhou, "Visual image encryption scheme based on inter-intra-block scrambling and weighted diffusion," *Vis. Comput.*, vol. 40, no. 2, pp. 731–746, 2024, doi: 10.1007/s00371-023-02812-2.

[37] E. Alotaibi, R. B. Sulaiman, and M. Almaiah, "Assessment of cybersecurity threats and defense mechanisms in wireless sensor networks," *J. Cyber Secur. Risk Audit.*, vol. 2025, no. 1, pp. 47–59, 2025.