cryptography

MDPI

Article

A Face Image Encryption Scheme Based on Nonlinear Dynamics and RNA Cryptography

Xiyuan Cheng [1], Tiancong Cheng [2], Xinyu Yang [3], Wenbin Cheng [2,3,*] and Yiting Lin [2]

[1] School of Automation, Guangdong University of Technology, Guangzhou 510006, China; 2112304365@mail2.gdut.edu.cn
[2] School of Electronic Information, University of Electronic Science and Technology of China Zhongshan Institute, Zhongshan 528402, China; 2022030102008@stu.zsc.edu.cn (T.C.); 2024097@zsc.edu.cn (Y.L.)
[3] School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China; 202322011503@std.uestc.edu.cn
* Correspondence: chengwenbin@zsc.edu.cn

Abstract

With the rapid development of big data and artificial intelligence, the problem of image privacy leakage has become increasingly prominent, especially for images containing sensitive information such as faces, which poses a higher security risk. In order to improve the security and efficiency of image privacy protection, this paper proposes an image encryption scheme that integrates face detection and multi-level encryption technology. Specifically, a multi-task convolutional neural network (MTCNN) is used to accurately extract the face area to ensure accurate positioning and high processing efficiency. For the extracted face area, a hierarchical encryption framework is constructed using chaotic systems, lightweight block permutations, RNA cryptographic systems, and bit diffusion, which increases data complexity and unpredictability. In addition, a key update mechanism based on dynamic feedback is introduced to enable the key to change in real time during the encryption process, effectively resisting known plaintext and chosen plaintext attacks. Experimental results show that the scheme performs well in terms of encryption security, robustness, computational efficiency, and image reconstruction quality. This study provides a practical and effective solution for the secure storage and transmission of sensitive face images, and provides valuable support for image privacy protection in intelligent systems.

Keywords: RNA encoding; chaos; face privacy protection; image encryption

Academic Editor: Marek R. Ogiela

Received: 7 August 2025
Revised: 28 August 2025
Accepted: 1 September 2025
Published: 4 September 2025

Citation: Cheng, X.; Cheng, T.; Yang, X.; Cheng, W.; Lin, Y. A Face Image Encryption Scheme Based on Nonlinear Dynamics and RNA Cryptography. *Cryptography* **2025**, *9*, 57. https://doi.org/10.3390/cryptography9030057

1. Introduction

As information and communication technologies have advanced rapidly, there has been an exponential rise in the global volume of digital data. Among various data types, image data stands out due to its rich semantic content, high dimensionality, and widespread use in applications such as social media, healthcare, surveillance, and biometric authentication [1–4]. In this data-driven era, ensuring the security and privacy of image content has become a pressing concern. Traditional privacy protection techniques—such as box blurring or pixelation—are no longer sufficient in confronting modern threats; such methods may offer a superficial level of obfuscation, but fail to resist advanced reconstruction or inference attacks, especially when applied to large-scale visual datasets [4–7]. Moreover, the unique characteristics of image data pose additional challenges to conventional cryptographic techniques [7–10]. Unlike text, digital images exhibit high redundancy,

strong spatial correlation among adjacent pixels, and a typically non-uniform, clustered distribution of features [7,11–13]. These traits significantly reduce the effectiveness of traditional text-based encryption algorithms when directly applied to image domains [14–17]. Simultaneously, the stakes for image security have grown considerably. Facial images, for instance, have emerged as critical biometric identifiers in access control, authentication, and surveillance systems. The leakage or unauthorized manipulation of such images may result in severe consequences, including identity theft, personal data breaches, and financial fraud [18–20]. Therefore, developing specialized image encryption schemes that account for the structural properties of images and are resilient to both statistical and cryptanalytic attacks is imperative [21–24]. To address these challenges, researchers have explored various encryption strategies tailored for image data, including chaotic systems, DNA/RNA coding, fractal transformations, and neural network–based methods [25–28]. Among them, chaos-based encryption has gained particular attention due to its desirable properties such as sensitivity to initial conditions [29–33], ergodicity [34–38], and pseudo-randomness [39–42]—characteristics that align well with security requirements like diffusion and confusion.

Although global encryption algorithms such as AES and DES were once considered the cornerstone of secure communication, their computational intensity imposes significant burdens in large-scale image processing. Moreover, they lack the flexibility required for hierarchical privacy protection. These limitations are particularly pronounced in real-time surveillance systems [43] and cloud-based facial recognition services [44], where there is an urgent need for encryption paradigms that balance both security and efficiency.

Recent research has introduced a range of multidimensional approaches to tackle the challenges of image encryption [4,45,46]. In 2021, Wang et al. [47] proposed a robust triple encryption scheme that integrates 2D chaotic systems, compressed sensing, and 3D discrete cosine transform (3D-DCT), significantly improving both visual security and decryption efficiency. In 2022, Yamni et al. [48] introduced a high-concealment audio watermarking method combining dual-tree complex wavelet transform with fractional Charlier moments. Cao et al. [49] recently developed a robust watermarking technique for screen content images, which dynamically embeds retrievable information and improves extraction accuracy in copyright protection scenarios. These developments indicate a clear evolution from single-algorithm optimizations to multimodal cooperative protection strategies [50].

In the domain of facial privacy protection, encryption technologies have been widely explored [51–56]. Winkler et al. [57] developed TrustCAM, a privacy-aware smart camera that uses edge detection and gradient operators to generate alternative regions of interest (ROIs) and grants access to original ROIs through key-based mechanisms. To maintain a degree of visual usability while preventing unauthorized facial recognition, Zhou et al. (year not specified) proposed the Thumbnail-Preserving Encryption (TPE) method based on Generative Adversarial Networks (GANs), which generates encrypted thumbnails of facial images that block both human and machine recognition. Zhao et al. [58] introduced the Enhanced TPE (E-TPE), leveraging a bijection between pixel triplets and their ranks, along with a novel triplet-rank mapping technique to enable efficient and stable encryption and decryption. Chai et al. [59] proposed TPE-ADE, which integrates Huffman coding with reversible data hiding in JPEG images, enabling reversible encryption while improving visual utility and reducing data expansion. Furthermore, the PR3 method by Zhao et al. [58] replaces the seven least significant bits in an image via summation data embedding, storing any overflow in the most significant bits and preserving thumbnail approximations while allowing additional information to be embedded.

Despite these advancements, research on differential privacy for facial images remains in the exploratory stage due to the complexity of visual data. Existing methods face challenges such as excessive distortion, high computational cost, and the delicate balance between privacy and utility. For example, pixelation with Laplacian noise [60] can render images unrecognizable, while SVD-based noise injection may lead to impractical results. Feature-space approaches such as Eigenface perturbation (PEEP) offer lightweight protection but compromise recognition accuracy. Region-growing techniques with differential privacy effectively obscure sensitive areas but are computationally expensive. Frequency-domain approaches [61] distort images by removing DC components and injecting noise. GAN-based methods [62] perturb latent spaces but may suffer from unstable representations and information loss. The IdentityDP framework [63] anonymizes facial features through identity disruption but may compromise background quality. These methods often suffer from high computational cost and complexity that impair real-time performance. Selective encryption reduces computational load by encrypting only critical image regions, yet heavily relies on key security and lacks flexibility in privacy level adjustments, complicating data usage and model training.

The main contributions of this work are as follows:

- Unlike conventional image encryption methods that process the entire image uniformly—often wasting resources on redundant regions—our approach targets encryption specifically on facial regions. By integrating face detection technology, the proposed scheme focuses computational effort on sensitive areas, thereby improving both security and encryption efficiency while avoiding unnecessary overhead.
- Traditional biological coding methods often suffer from limited adaptability due to fixed encoding schemes and static operational rules, making them susceptible to cryptanalytic attacks. To address this, the proposed algorithm employs dynamic RNA encoding combined with variable rule selection mechanisms, introducing greater randomness and complexity to enhance resistance against unauthorized decryption.
- Many existing encryption frameworks lack structural robustness, particularly in scenarios involving known-plaintext or chosen-plaintext attacks. Drawing on insights from our previous cryptanalysis research, we identified structural vulnerabilities in static key designs and insufficient sensitivity to plaintext changes. To address these issues, our method integrates plaintext-dependent chaotic key generation. This correlation-based design not only enhances dynamic behavior but also significantly strengthens the system's resistance to various cryptographic attacks.

The following section outlines the structure of the rest of this paper. Section 2 introduces the chaotic system used, the MTCNN technique, and the RNA rules. Section 3 introduces the encryption algorithm designed in this paper. Section 4 presents experimental and simulation results. The last section concludes this paper.

2. Related Theories

2.1. Face Detection

Face detection is a critical prerequisite in facial recognition and encryption tasks, as the accurate extraction of facial regions significantly impacts subsequent processing steps. Early approaches such as the Viola–Jones algorithm, which is based on Haar-like features, offer fast detection speeds but suffer from limited accuracy under complex conditions. In recent years, the advancement of deep learning has led to the emergence of numerous face detection methods based on convolutional neural networks (CNNs), including Faster R-CNN [64], RetinaFace [65], and YOLO-Face [66], each balancing detection speed and accuracy differently.

A representative method in this domain is the Multi-task Cascaded Convolutional Network (MTCNN) proposed by Zhang et al. [67], which performs both face detection and facial landmark localization. MTCNN adopts a three-stage cascaded structure consisting of the Proposal Network (P-Net), the Refine Network (R-Net), and the Output Network (O-Net). These subnetworks progressively filter face candidates in a coarse-to-fine manner and regress the precise bounding box positions. In the final stage, the network also predicts five facial landmarks: the centers of both eyes, the nose tip, and the corners of the mouth. The primary rationale for selecting MTCNN lies in its multi-task cascaded architecture, online hard example mining, and lightweight optimization. These advantages enable it to comprehensively outperform traditional methods (e.g., Viola–Jones) and other deep learning models (e.g., single-task CNNs) in accuracy, speed, and robustness. Empirical results demonstrate state-of-the-art (SOTA) performance on challenging benchmarks such as FDDB, WIDER FACE, and AFLW, while meeting real-time requirements (>99 FPS). Consequently, MTCNN is an ideal solution for facial analysis tasks in unconstrained environments, such as surveillance systems or mobile devices.

The core of MTCNN lies in integrating face classification, bounding box regression, and landmark localization into a unified multi-task learning framework. During training, the model jointly optimizes the loss functions of the three subtasks. Face classification uses a cross-entropy loss, while both the bounding box and landmark regression tasks are trained using the Euclidean (L2) loss function. The bounding box regression loss is defined as

$$\mathrm{L}_{\mathrm{box}} = \|\hat{t} - t\|^2 \tag{1}$$

Here, $\hat{t}$ denotes the predicted bounding box offset by the network, while t represents the corresponding ground-truth offset. The loss function for facial landmark localization follows a similar form and is given by

$$\mathrm{L}_{\mathrm{landmark}} = \| \hat{l} - l \|^2 \tag{2}$$

Here, $\hat{l}$ and l denote the predicted and ground-truth facial landmark coordinates, respectively. MTCNN achieved state-of-the-art performance on several public benchmark datasets, including FDDB, WIDER FACE, and AFLW, while maintaining high computational efficiency (up to 99 fps on GPU). Owing to its lightweight architecture and well-integrated multi-task design, MTCNN has been widely adopted in various face-related tasks and has served as a valuable reference for the development of subsequent lightweight object detection frameworks.

Given that the facial recognition encryption task in this study relies on accurate face region extraction and landmark-assisted alignment, MTCNN provides a favorable balance between detection accuracy, model compactness, and landmark localization capability. Therefore, MTCNN is employed in this work as the face detection module in the preprocessing stage to ensure the applicability and stability of the proposed encryption scheme.

2.2. 4D-NDS Chaotic System

We used a novel non-degenerate four-dimensional discrete-time chaotic system (4D-NDS) [25], and the process is as follows: First, a transformation matrix is derived, denoted as M. Next, employing a uniformly bounded anti-control function h together with a control matrix Q, inverse control is applied to M, expressed by the relation

$$z_m(k+1) = M\, z_m(k) + Q\, h(\lambda\, z_m(k), \eta) \tag{3}$$

where $z_m(k)$ represents the iterative sequence, while λ and η are parameters associated with the anti-control mechanism. By performing pole assignment based on Equation (3), one can compute the Lyapunov exponent $LE_+ = m$, indicating the resulting system achieves an m-dimensional discrete hyperchaotic behavior.

Under this framework, the specific form of the four-dimensional discrete non-degenerate chaotic model is given as follows:

$$M = \begin{pmatrix} -0.1667 & 0.0333 & -0.5667 & 0.5333 \\ -0.1667 & -0.0667 & -0.4667 & 0.6333 \\ -0.3667 & -0.0667 & -0.2667 & 0.4333 \\ 0 & 0.3000 & -0.3000 & 0.1000 \end{pmatrix} \tag{4}$$

$$h(\lambda z_k, \eta_k) = \begin{cases} \eta_1 \dfrac{\sin(2\pi z_1(k))}{1+\lambda z_1^2(k)} \\ \eta_2 \dfrac{\sin(2\pi z_2(k))}{1+\lambda z_2^2(k)} \\ \eta_3 \dfrac{\sin(2\pi z_3(k))}{1+\lambda z_3^2(k)} \\ \eta_4 \dfrac{\sin(2\pi z_4(k))}{1+\lambda z_4^2(k)} \end{cases} \tag{5}$$

$$Q = I = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6}$$

Accordingly, the complete discrete-time dynamic system can be written compactly as

$$\begin{pmatrix} z_1(k+1) \\ z_2(k+1) \\ z_3(k+1) \\ z_4(k+1) \end{pmatrix} = M \begin{pmatrix} z_1(k) \\ z_2(k) \\ z_3(k) \\ z_4(k) \end{pmatrix} + I\, h(\lambda z_k, \eta_k) \tag{7}$$

$$\begin{cases} z_1(k+1) = M_{(1,1)} z_1(k) + M_{(1,2)} z_2(k) + M_{(1,3)} z_3(k) + M_{(1,4)} z_4(k) + \eta_1 \dfrac{\sin(2\pi z_1(k))}{1+\lambda z_1^2(k)} \\ z_2(k+1) = M_{(2,1)} z_1(k) + M_{(2,2)} z_2(k) + M_{(2,3)} z_3(k) + M_{(2,4)} z_4(k) + \eta_2 \dfrac{\sin(2\pi z_2(k))}{1+\lambda z_2^2(k)} \\ z_3(k+1) = M_{(3,1)} z_1(k) + M_{(3,2)} z_2(k) + M_{(3,3)} z_3(k) + M_{(3,4)} z_4(k) + \eta_3 \dfrac{\sin(2\pi z_3(k))}{1+\lambda z_3^2(k)} \\ z_4(k+1) = M_{(4,1)} z_1(k) + M_{(4,2)} z_2(k) + M_{(4,3)} z_3(k) + M_{(4,4)} z_4(k) + \eta_4 \dfrac{\sin(2\pi z_4(k))}{1+\lambda z_4^2(k)} \end{cases}$$

Here, the parameters satisfy $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \lambda$, and $\eta_1 = \eta_2 = \eta_3 = \eta_4 = \eta$. The initial conditions are assigned as $z_1(1) = 0.1$, $z_2(1) = 0.2$, $z_3(1) = 0.3$, and $z_4(1) = 0.4$, with $\lambda = 1$ and $\eta = 1$.

2.3. RNA Coding and Operation Rules

Ribonucleic acid (RNA) is a vital biomolecule that plays a central role in the transmission and expression of genetic information within biological systems. Structurally, RNA is a single-stranded polymer composed of ribonucleotides connected via phosphodiester bonds. Each ribonucleotide includes one of four nitrogenous bases: adenine (A), guanine (G), uracil (U), and cytosine (C). These bases exhibit specific complementary pairing behavior—adenine (A) pairs with uracil (U), and cytosine (C) pairs with guanine (G)—forming the basis of information replication and biological coding.

This inherent base-pairing principle, often denoted as A–U and C–G complementarity, provides a natural framework for representing binary or symbolic data in computational systems. Inspired by the precision and stability of genetic coding mechanisms, researchers have introduced RNA-based models into the field of information security, particularly in image encryption. By mimicking the way genetic information is encoded and transformed in living organisms, RNA encoding schemes can introduce nonlinearity, redundancy, and dynamic mapping into cryptographic operations.

In the context of image encryption, RNA coding offers a flexible method for encoding pixel data into symbolic sequences, which can then be manipulated using biologically inspired transformation rules. These rules—typically defined based on the eight valid permutations of base pairings—are capable of achieving confusion and diffusion effects essential for secure encryption. The eight RNA coding rules, as summarized in Table 1, provide a diverse set of mappings that can be dynamically selected based on chaotic sequences or secret keys, enhancing both unpredictability and resistance to cryptanalysis.

Table 1. RNA base complementarity rules.

Rule	1	2	3	4	5	6	7	8
00	A	A	U	U	C	C	G	G
01	C	G	C	G	A	U	A	U
10	G	C	G	C	U	A	U	A
11	U	U	A	A	G	G	C	C

In addition, the RNA has six types of operations: addition, subtraction, addition–complement, subtraction–complement, exclusive OR (XOR) and exclusive NOR (XNOR), like the binary system. The specific operation rules are shown in Tables 2 and 3.

Table 2. RNA base operation.

Base	Addition $+$	AGCU Subtraction $-$	XOR $\oplus$
A	A G C U	A U C G	A G C U
G	G C U A	G A U C	G A U C
C	C U A G	C G A U	C U A G
U	U A G C	U C G A	U C G A

Table 3. RNA base inverse operation.

Base	Add–Complement $+'$	AGCU Sub–Complement $-'$	XNOR $\odot$
A	U C G A	U A G C	U C G A
G	C G A U	C U A G	C U A G
C	G A U C	G C U A	G A U C
U	A U C G	A G C U	A G C U

3. Proposed Encryption Algorithm

The proposed image encryption algorithm consists of four modules: (1) key generation and processing, (2) block permutation encryption, (3) RNA encryption, and (4) bit diffusion encryption. Throughout, we denote the plaintext color image as P of size $H \times W \times 3$ (height H, width W, three color channels), and the final ciphertext image as C of the same dimensions. Each module uses pseudo-random sequences generated from a 4D chaotic system driven by the image contents. Below we describe each module in detail, defining

all variables and operations. The overall diagram of the encryption method proposed in the paper is shown in Figure 1.

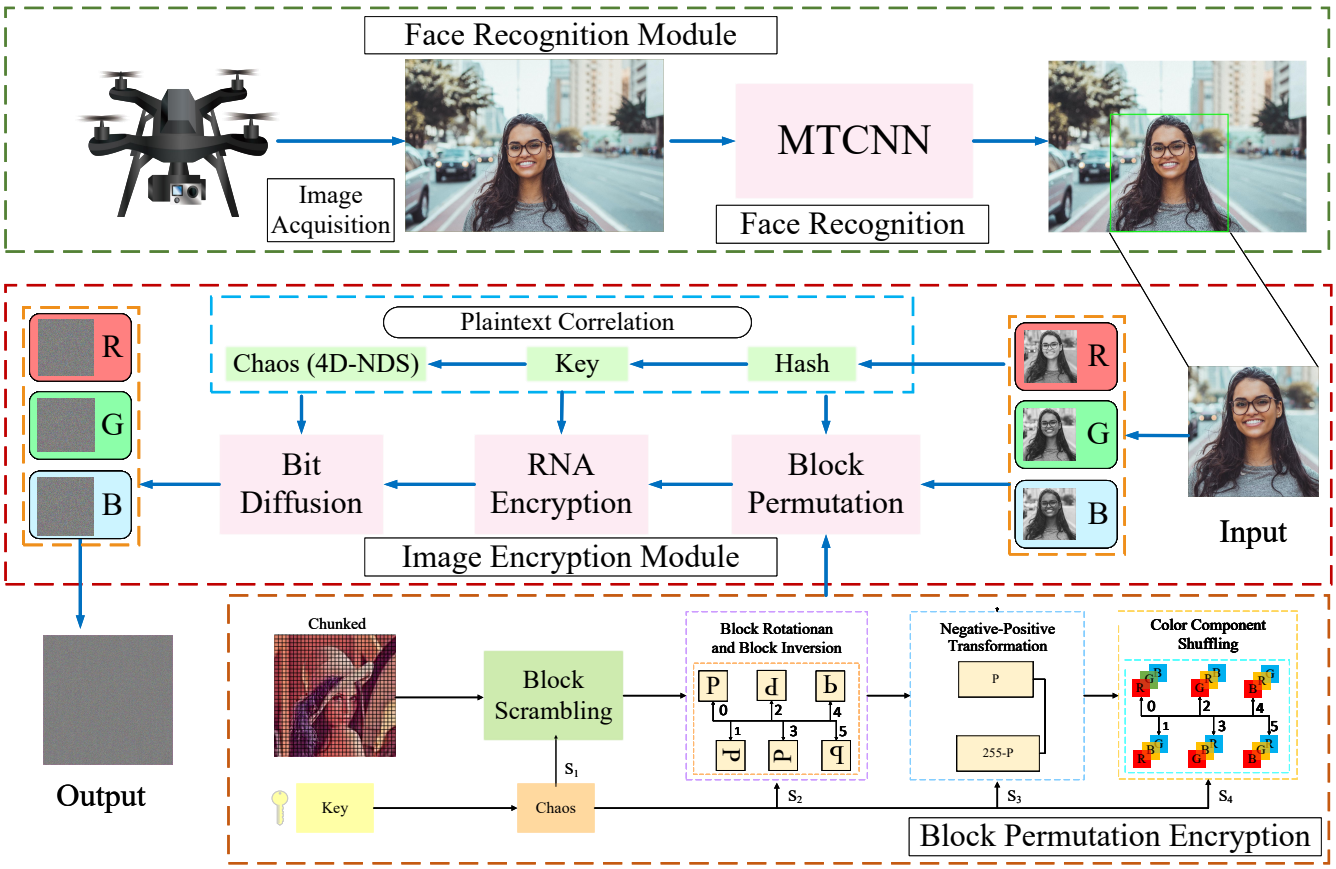


Figure 1. The proposed algorithm flowchart.

3.1. Key Generation and Processing

The encryption keys are derived from the plaintext image P itself using a chaotic system. First, compute the SHA-256 hash of P (viewed as a byte array). Denote this hash (a 64-byte hexadecimal string) by $h = \mathrm{SHA256}(P)$. Define the following integer keys:

$$\mathrm{key}_1 = \left(\sum_{i=0}^{20} h_{3i+1} \right) \bmod 100, \tag{8}$$

$$\mathrm{key}_2 = \left(\sum_{i=0}^{20} h_{3i+2} \right) \bmod 100, \tag{9}$$

$$\mathrm{key}_3 = h_{64} + \frac{1}{10{,}000} \sum_{i=0}^{20} h_{3i+3}. \tag{10}$$

where h_j denotes the j-th byte (in decimal) of the hash (indexing from 1). The plaintext-dependent keys are then used to initialize a 4-dimensional discrete chaotic map.

Set the initial chaotic state vector $\mathbf{X}(1) = (X_1(1), X_2(1), X_3(1), X_4(1))$ by

$$X_1(1) = \frac{\mathrm{key}_1}{2}, \tag{11}$$

$$X_2(1) = \frac{\mathrm{key}_1}{3}, \tag{12}$$

$$X_3(1) = \frac{\mathrm{key}_1}{4}, \tag{13}$$

$$X_4(1) = \frac{\text{key}_1}{5}. \tag{14}$$

Let $E = 1 + \text{key}_2$ and $B = 1 + \text{key}_3$. Define constant matrices as follows:

$$C_0 = \text{diag}(0.1, -0.2, 0.4, -0.7), \tag{15}$$

$$M_0 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \tag{16}$$

$$A = M_0 \cdot C_0 \cdot M_0^{-1}. \tag{17}$$

and compute $A = M_0, C_0, M_0^{-1}$ (so that A is a fixed 4×4 matrix). The chaotic map iterates for $k = 1, 2, \ldots$ as

$$\begin{bmatrix} X_1(k+1) \\ X_2(k+1) \\ X_3(k+1) \\ X_4(k+1) \end{bmatrix} = A \begin{bmatrix} X_1(k) \\ X_2(k) \\ X_3(k) \\ X_4(k) \end{bmatrix} + E \cdot \begin{bmatrix} \frac{\sin(2\pi X_1(k))}{1+BX_1(k)^2} \\ \frac{\sin(2\pi X_2(k))}{1+BX_2(k)^2} \\ \frac{\sin(2\pi X_3(k))}{1+BX_3(k)^2} \\ \frac{\sin(2\pi X_4(k))}{1+BX_4(k)^2} \end{bmatrix}. \tag{18}$$

Here A has rows indexed by the linear combinations in code. Iterate this system for $M = 40, H, W + 5000$ steps, then discard the first 5000 values to eliminate transients. Denote the remaining sequences by

$$X = \{X_1(5002), X_1(5003), \ldots\}, \tag{19}$$

$$Y = \{X_2(5002), X_2(5003), \ldots\}, \tag{20}$$

$$Z = \{X_3(5002), X_3(5003), \ldots\}, \tag{21}$$

$$W = \{X_4(5002), X_4(5003), \ldots\}. \tag{22}$$

Next, form two long 1D sequences by concatenating X and Y, and Z and W:

$$x_2 = [X, Y], \tag{23}$$

$$y_2 = [Z, W]. \tag{24}$$

From these, extract six chaotic key sequences $S_1, S_2, \ldots, S_6$ of lengths proportional to $H \times W$:

- S_1 = the first $8HW$ elements of x_2;
- S_2 = the next $2HW$ elements of x_2;
- S_3 = the first $8HW$ elements of y_2;
- S_4 = the next $8HW$ elements of y_2;
- S_5 = the next $24HW$ elements of y_2;
- S_6 = the elements $10HW + 1$ through $13HW$ of x_2.

Each S_i is thus a 1D array; in later steps, we reshape them to match the image dimensions or color layers as needed. In particular, we will reshape S_6 into three $H \times W$ arrays (for R, G, B) as explained below.

Finally, define a 4-element block-encryption key vector $K = (K_1, K_2, K_3, K_4)$ by taking the first two entries of S_1 and S_2:

$$K_1 = S_1(1), \tag{25}$$

$$K_2 = S_1(2), \tag{26}$$

$$K_3 = S_2(1), \tag{27}$$

$$K_4 = S_2(2). \tag{28}$$

These keys seed the random block-scrambling in the next module. In summary, all sequences $S_1, \ldots, S_6$ and key vector K are derived from the plaintext P; they serve as pseudo-random keys for the subsequent encryption steps.

3.2. Block Permutation Encryption

The first encryption module permutes and substitutes the pixels of P under chaotic control. Denote the working plaintext matrix by $K_C = P$ (converted to double precision). We apply four operations in sequence: (a) spatial channel permutation, (b) chaotic single-plane transform, (c) block scramble, and (d) substitution. The result is an intermediate ciphertext C_1 of size $H \times W \times 3$. We describe each in turn.

3.2.1. Spatial Channel Permutation (SpatialTrans)

Reshape the sequence S_1 into an $H \times W \times 3$ array by extracting three contiguous $H \times W$ slices: let

$$S_{1,R}(i,j,k) = S_1((k-1) \cdot H \cdot W + (i-1) \cdot W + j), \quad k = 1,2,3. \tag{29}$$

or pixel (i,j). At each pixel, $S_{1,R}(i,j,1)$, $S_{1,R}(i,j,2)$, $S_{1,R}(i,j,3)$ form three chaotic values. Sort these three values to obtain a permutation $\pi(i,j)$ of $1,2,3$. Then the pixel's color channels are permuted according to this ordering. Formally, if $[S_{1,R}(i,j,1), S_{1,R}(i,j,2), S_{1,R}(i,j,3)]$ sorted in ascending order corresponds to indices $I(i,j,1), I(i,j,2), I(i,j,3)$, then set

$$K'_C(i,j,k) = K_C(i,j,I(i,j,k)), \quad k = 1,2,3. \tag{30}$$

In effect, the R, G, B values at (i,j) are shuffled by a channel permutation determined by the chaotic sequence. Denote the output of this step as K'_C.

3.2.2. Chaotic Single-Plane Transform (ChaoticMagicTrans)

This step applies additional intra-channel position shifts. Use the chaotic sequence S_2 (reshaped similarly as needed) to generate row-shift and column-shift vectors $R_1, C_1, R_2, C_2, R_3, C_3$. Specifically, let $R_1 = S_2(1 : H)$, $C_1 = S_2(H+1 : H+W)$, $R_2 = S_2(H+W+1 : 2H+W)$, $C_2 = S_2(2H+W+1 : 2H+2W)$, $R_3 = S_2(2H+2W+1 : 3H+2W)$, and $C_3 = S_2(3H+2W+1 : 3H+3W)$. Sort each vector to obtain permutation indices $I_1, J_1, \ldots, I_3, J_3$. Then construct three $H \times W$ shift maps $S^{(1)}, S^{(2)}, S^{(3)}$ by

$$S^{(k)}(m,n) = J_k((n + I_k(m) - 1) \bmod W + 1). \tag{31}$$

for $k = 1,2,3$. Finally, permute each color channel $k = 1,2,3$ of K'_C by

$$K''_C(m,n,k) = K'_C(m, S^{(k)}(m,n), k), \tag{32}$$

and then rearrange rows cyclically by the rule

$$\text{Row index update: } m' = ((m-1) - S^{(k)}(1,n)) \bmod H + 1. \tag{33}$$

The nested loops in code implement these shifts. The output is an intermediate matrix K_C'' of the same size. In summary, the single-plane transform mixes pixel positions within each color plane according to the chaotic maps derived from S_2.

3.2.3. Block Scramble Encryption (BlockScram_EnAlgorithm)

Divide K_C'' into non-overlapping blocks of size $B \times B$ with $B = 4$ (assuming H and W are multiples of 4; if not, pad to the next multiple of 4). Denote the number of blocks by $(H/B) \times (W/B)$. Using the key vector $K = (K_1, K_2, K_3, K_4)$ as random seeds, generate four pseudo-random integer sequences each of length equal to the number of blocks:

- $S_1^{(b)}$: random permutation of $\{1, \ldots, N_b\}$ (seed K_1);
- $S_2^{(b)}$: integers in $\{0, \ldots, 5\}$ (seed K_2);
- $S_3^{(b)}$: bits in $\{0, 1\}$ (seed K_3);
- $S_4^{(b)}$: integers in $\{0, \ldots, 5\}$ (seed K_4).

Perform the following encryption on the blocks $B_{p,q}$ at block–row p and block–column q:

(a) Block Permutation

Rearrange all blocks according to $S_1^{(b)}$. That is, swap block i with block $S_1^{(b)}(i)$ for $i = 1, \ldots, (HW/B^2)$.

(b) Block Rotation/Flip

For each block index i (corresponding to block (p, q)), use $s = S_2^{(b)}(i)$ to rotate or flip the 4×4 block:

- $s = 0$: do nothing (no rotation).
- $s = 1$: rotate the block 90° clockwise.
- $s = 2$: rotate 180°.
- $s = 3$: rotate 270° clockwise.
- $s = 4$: flip horizontally (reflect across vertical center).
- $s = 5$: flip vertically.

These operations are implemented by appropriate index swaps within each 4×4 block.

(c) Negative–Positive Transform

For each block index i, if $S_3^{(b)}(i) = 1$, then XOR every pixel in the block with 1; if $S_3^{(b)}(i) = 0$, then replace each pixel value x by $255 - x$. (This flips bits or inverts values.) The specific algorithm is shown in Algorithm 1.

(d) Color Channel Permutation

For each block index i, let $s = S_4^{(b)}(i)$ and perform a channel permutation inside the block:

- $s = 0$: do nothing (keep RGB order).
- $s = 1$: permute RGB $\rightarrow$ RBG (swap G,B).
- $s = 2$: permute RGB $\rightarrow$ GRB (swap R,G).
- $s = 3$: permute RGB $\rightarrow$ BGR (swap R,B).
- $s = 4$: permute RGB $\rightarrow$ BRG (swap (G,B) then (R,B)).
- $s = 5$: permute RGB $\rightarrow$ GBR (swap (R,G) then (G,B)).

In code, each swap is performed by XOR swapping of the corresponding planes.

Upon completion of the aforementioned block operations, the transformed blocks are reassembled to generate the final permuted image, denoted as $K_C^{(3)}$.

Algorithm 1 Block scrambling procedure.

Require: Intermediate image K_C'' of size $M \times N$, block size $B \times B$, key K
Ensure: Permuted image $K_C^{(3)}$
Divide K_C'' into non-overlapping blocks $B_{p,q}$ of size $B \times B$
Generate chaotic sequences S_1^b, S_2^b, S_3^b, and S_4^b using key K
for $i = 1$ to N_b **do**
    Swap block i with block $S_1^b(i)$
end for
for each block index i **do**
    Rotate or flip the block according to $S_2^b(i)$
    if $S_3^b(i) = 1$ **then**
        $B_i \leftarrow \text{bitxor}(B_i, 1)$
    else
        $B_i \leftarrow 255 - B_i$
    end if
    Permute RGB channels of B_i based on $S_4^b(i)$
end for
Reassemble all processed blocks to form $K_C^{(3)}$

This yields an image heavily scrambled in blocks, orientations, and color channels.

3.2.4. Substitution (Value Diffusion)

Finally, we mix pixel values across the image. Let $P' = K_C^{(3)}$ be the permuted image (still size $H \times W \times 3$). We perform a forward scan over pixels (row by row, within each row left to right, and channel by channel) to compute the intermediate ciphertext C_1. Let $F = 256$ be the modulus (for 8-bit pixels). Using the reshaped chaotic matrix from S_1 (converted to integer mod F) as $S_1'(i, j, k)$, apply the following:

- Initial pixel $(1, 1)$ in red channel: $C_1(1, 1, 1) = [P'(1, 1, 1) + P'(H, W, 3) + S_1'(1, 1, 1)] \bmod F$. (Here $P'(H, W, 3)$ is the blue value of the last pixel of P'.)
- First pixel of other channels $(1, 1, k)$ for $k = 2, 3$: $C_1(1, 1, k) = [P'(1, 1, k) + C_1(H, W, k-1) + S_1'(1, 1, k)] \bmod F$.
- First row, general column $(1, n)$ for $n > 1$, any channel k: $C_1(1, n, k) = [P'(1, n, k) + C_1(1, n-1, k) + S_1'(1, n, k)] \bmod F$.
- General case (m, n, k) with $m > 1$: $C_1(m, n, k) = [P'(m, n, k) + C_1(m-1, n, k) + S_1'(m, n, k)] \bmod F$.

In other words, each pixel of C_1 is the sum (mod 256) of the current plaintext value, the previously encrypted neighbor, and a chaotic offset. This introduces diffusion of values across the image and channels. The resulting matrix C_1 is the output of the block permutation encryption module.

The overall effect of Section 2 is to transform P into a scrambled, diffused intermediate ciphertext C_1 using chaotic permutations and substitutions.

3.3. RNA Encryption

The second module applies a DNA/RNA-based substitution process on C_1, inspired by genetic coding rules. Treat C_1 as three gray-scale images C_R, C_G, C_B of size $H \times W$. For each channel, we perform the following operations using chaotic sequences S_3, S_4, S_5:

3.3.1. Bit-Layer Conversion

Interpret each 8-bit pixel of C_k ($k = R, G, B$) as two 4-bit halves: the high 4 bits and the low 4 bits. Stack the high halves of all $H \times W$ pixels into a binary matrix C^H of size $H \times W \times 12$ (since 3 channels $\times$ 4 bits per channel), and similarly form C^L from the

low halves. (This step is conceptual; in implementation we simply operate on the 8-bit values bitwise).

3.3.2. Octal Synthesis

For the high-bit matrix C^H, group each 3-bit slice into one octal digit. Concretely, for each channel separately, take 3 high-order bits (interpreted in base-2) and form an octal number in $\{0, \ldots, 7\}$. Let C^H_{oct} denote the resulting $H \times W$ matrix of octal values (one per pixel per channel).

3.3.3. Chaotic Encode-Key Generation

Flatten C_k to a vector of length $N = HW$. From S_3 and S_4, generate two integer arrays:

$$\text{encode}(i) = \left(\left\lfloor (|S_3(i)| - \lfloor |S_3(i)| \rfloor) \times 10^{15} \right\rfloor \bmod 10^8 \right) \bmod 8 + 1 \in \{1, \ldots, 8\}, \tag{34}$$

and reshape encode to an $H \times W$ matrix E. These values $1 \ldots 8$ select one of eight RNA mapping rules (per pixel pair of bits).

Similarly, compute

$$\text{decode}(i) = \left(\left\lfloor (|S_4(i)| - \lfloor |S_4(i)| \rfloor) \times 10^{15} \right\rfloor \bmod 10^8 \right) \bmod 8 + 1, \tag{35}$$

and reshape into an $H \times W$ matrix D. These are used for decoding later.

Also extract N chaotic bytes from S_3 to form an $H \times 4W$ key matrix (denoted $os1$) by

$$os1(i,j) = \left(\left\lfloor (|S_3(4(i-1)W + j)| - \lfloor |S_3(4(i-1)W + j)| \rfloor) \times 10^{15} \right\rfloor \bmod 6 \right) + 1. \tag{36}$$

Each value lies in $\{1, \ldots, 6\}$.

3.3.4. Dynamic RNA Encoding

Define two DNA/RNA mapping operations: RNA_Dyn_Encode(M, E) encodes an $H \times W$ numeric matrix M into an $H \times 4W$ character matrix of $\{A, C, G, U\}$ by mapping each pixel's 8 bits into 4 letters using the rule $E(i,j)$ for that column pair. Specifically, each 2-bit nibble of a pixel is converted to one letter according to a genetic code rule chosen by E.

Compute

$$\text{RNA\_img} = \text{RNA\_Dyn\_Encode}(C_k, E), \tag{37}$$

$$\text{RNA\_key} = \text{RNA\_Dyn\_Encode}(K', E), \tag{38}$$

where K' is the matrix formed by the next N bytes from S_3 mod 256 (these serve as a DNA "key image"). Both RNA_img and RNA_key are $H \times 4W$ matrices of letters in $\{A, C, G, U\}$.

3.3.5. Chaotic Genetic Operations

Perform chaotic combination of RNA_img and RNA_key letterwise. For each position (i,j), look up the operation code $op = os1(i,j) \in \{1, \ldots, 6\}$. Then compute

$$\text{RNA\_out}(i,j) = \begin{cases} \text{RNA\_add}(\text{RNA\_img}(i,j), \text{RNA\_key}(i,j)) & \text{if } op = 1, \\ \text{RNA\_sub}(\text{RNA\_img}(i,j), \text{RNA\_key}(i,j)) & \text{if } op = 2, \\ \text{RNA\_comp\_add}(\text{RNA\_img}(i,j), \text{RNA\_key}(i,j)) & \text{if } op = 3, \\ \text{RNA\_comp\_sub}(\text{RNA\_img}(i,j), \text{RNA\_key}(i,j)) & \text{if } op = 4, \\ \text{RNA\_xor}(\text{RNA\_img}(i,j), \text{RNA\_key}(i,j)) & \text{if } op = 5, \\ \text{RNA\_xnor}(\text{RNA\_img}(i,j), \text{RNA\_key}(i,j)) & \text{if } op = 6. \end{cases} \tag{39}$$

Each of these operations takes two bases (A, C, G, U) and returns a base according to a fixed table (as given in the implementation code). The result is a new $H \times 4W$ base matrix RNA_out.

3.3.6. Dynamic RNA Decoding

Finally, convert RNA_out back to bytes. Using the decoding rule $D(i,j) \in \{1,\ldots,8\}$ for each group of 4 bases (each pixel), apply RNA_Dyn_Decode to reconstruct an $H \times W$ numeric matrix. This yields the encrypted channel $\widetilde{C}_k$ for the k-th color.

3.3.7. Final Output

Repeat the steps above for $k = R, G, B$. Let C_2 be the $H \times W \times 3$ image combining $\widetilde{C}_R, \widetilde{C}_G, \widetilde{C}_B$. This is the output of the RNA encryption module.

Each pixel's bits are expanded into RNA bases, combined with a chaotic DNA key via one of six genetic operations, then collapsed back into pixel values. The chaotic sequences S_3, S_4, S_5 determine the encode/decode rules and operations, ensuring high sensitivity.

3.4. Bit Diffusion Encryption

After the RNA encryption, we obtain the intermediate cipher image C_2. In this module, we introduce a multi-directional bit-level diffusion process using chaotic sequences to enhance the pixel correlation disruption and avalanche effect.

3.4.1. Pixel Rearrangement

Let C_2 be of size $H \times W \times 3$, composed of three channels: $\widetilde{C}_R$, $\widetilde{C}_G$, and $\widetilde{C}_B$. First, reshape each color channel into a 1D vector of length $N = H \times W$:

$$\widetilde{C}_k^{vec} = \text{reshape}(\widetilde{C}_k), \quad k \in \{R, G, B\}.$$

Generate a permutation sequence P from the chaotic sequence S_5 as

$$P(i) = \text{mod}\Big(\Big\lfloor (|S_5(i)| - \lfloor |S_5(i)| \rfloor) \times 10^{15} \Big\rfloor, N\Big) + 1, \quad i = 1,\ldots,N.$$

Then perform position permutation on the vectors:

$$\widetilde{C}_k^{perm}(i) = \widetilde{C}_k^{vec}(P(i)), \quad \forall i, \quad k \in \{R, G, B\}.$$

3.4.2. Bit-Level Decomposition and Diffusion

Each 8-bit value in $\widetilde{C}_k^{perm}$ is decomposed into individual bits:

$$B_k^{(b)}(i) = \text{bit}_b(\widetilde{C}_k^{perm}(i)), \quad b = 1,\ldots,8,$$

where $\text{bit}_b(x)$ extracts the b-th bit (from MSB to LSB) of byte x.

Form three binary matrices B_k of size $8 \times N$, with each row corresponding to one bit-plane.

Chaotic Key Mask Generation: Generate $8N$ chaotic bits from S_5, construct a bit-mask matrix K_{bit} of size $8 \times N$:

$$K_{bit}(b,i) = \text{mod}\Big(\Big\lfloor (|S_5(8(i-1)+b)| - \lfloor |S_5(8(i-1)+b)| \rfloor) \times 10^{15} \Big\rfloor, 2\Big),$$

for $b = 1,\ldots,8$, $i = 1,\ldots,N$.

Bitwise Diffusion: Apply circular bit-level XOR and chaotic-driven rotation:

1. For each bit-plane b, perform

$$B_k^{(b)}(i) = B_k^{(b)}(i) \oplus B_k^{(b)}((i-1) \mod N + 1) \oplus K_{bit}(b,i).$$

2. After XOR diffusion, rotate each row b by δ_b bits, where

$$\delta_b = \mathrm{mod}\left(\sum_{i=1}^{N} K_{bit}(b,i), N\right).$$

3.4.3. Bit Recomposition and Output

Recombine the diffused bits back into 8-bit pixel values:

$$\widehat{C}_k^{perm}(i) = \sum_{b=1}^{8} B_k^{(b)}(i) \times 2^{8-b}, \quad i = 1, \ldots, N.$$

Apply inverse permutation P^{-1} to restore image structure:

$$\widehat{C}_k^{vec}(P(i)) = \widehat{C}_k^{perm}(i), \quad i = 1, \ldots, N.$$

Finally, reshape $\widehat{C}_k^{vec}$ to $H \times W$ to get the final encrypted channel:

$$C_k^{enc} = \mathrm{reshape}(\widehat{C}_k^{vec}, H, W), \quad k \in \{R, G, B\}.$$

Let the final ciphertext image be

$$C_{final} = \mathrm{cat}(C_R^{enc}, C_G^{enc}, C_B^{enc}),$$

a three-channel RGB image of size $H \times W \times 3$.

4. Experimental Results and Analysis

The experimental platform utilized was a MacBook Air (M4 chip) with MATLAB R2025a computational software installed. The device features an M4 processor with 10-core CPU and 10-core GPU, 32 GB unified memory, running on the macOS Sequoia 15.5 operating system. All experimental images presented in this study were sourced from the standardized USC-SIPI image database to ensure result reliability.

4.1. Statistics Histogram

Image encryption algorithms need to be adaptable to various application scenarios, ensuring that different types of images can be encrypted into unrecognizable cipher images. The original image can only be fully recovered with the correct key, and without it, no useful information about the original image can be obtained. In this paper, the encryption process is simulated using test images of different colors, and their pixel histograms are displayed in Figure 2. Analyzing the histogram of an image provides valuable information. Furthermore, Figure 3 illustrates the three-dimensional histograms of both the original image and the corresponding encrypted image. Observing these histograms highlights that the encrypted image exhibits an even distribution of pixels on the red, green, and blue planes. This demonstrates the algorithm's effectiveness in encrypting natural images into high-performance cipher images.

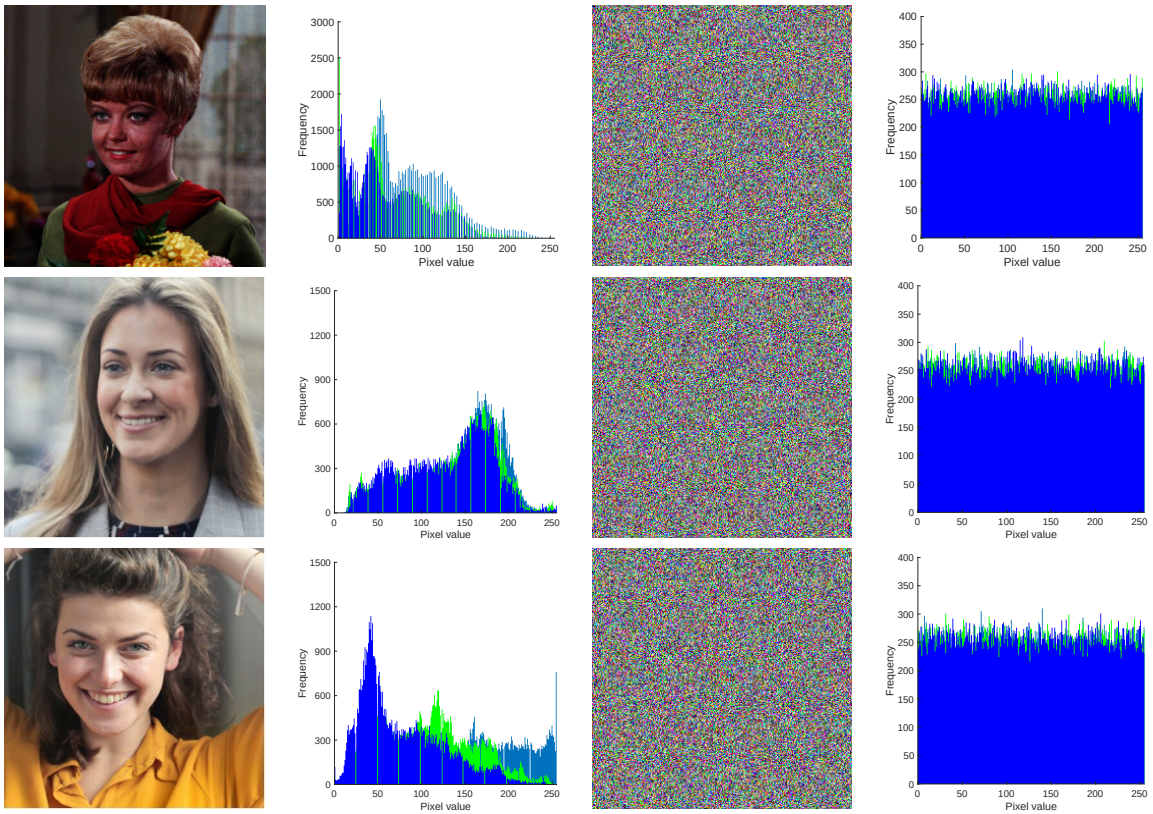


Figure 2. Images before and after encryption: (**a**) original images; (**b**) histogram of (**a**); (**c**) encrypted images; (**d**) histogram of (**c**).

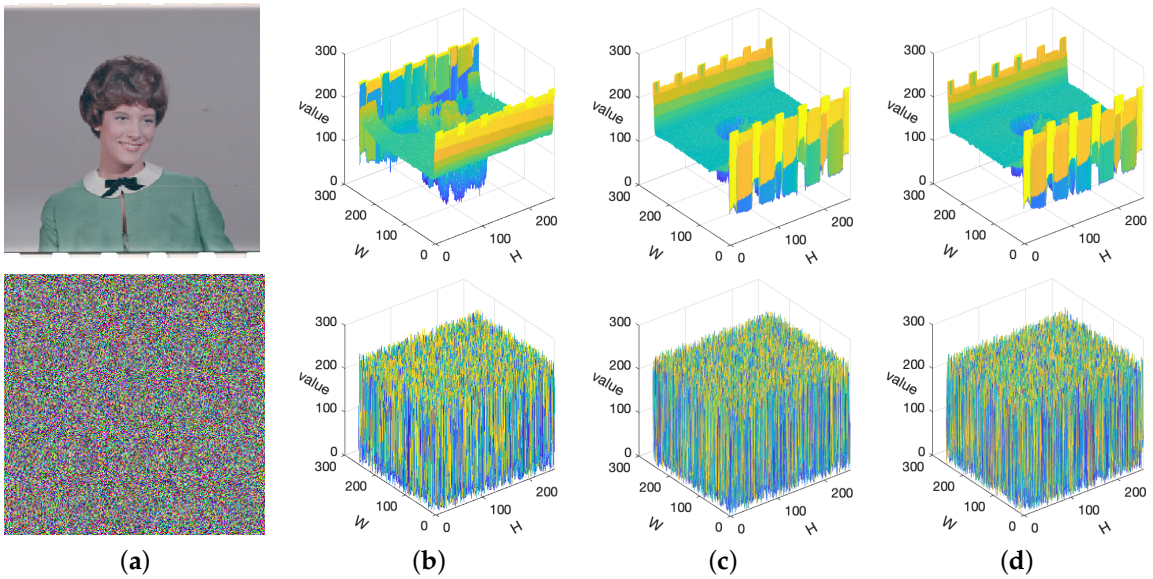


Figure 3. Three-dimensional visualization histogram of plaintext image and ciphertext image. (**a**) Horizontal correlation of plaintext images and ciphertext images; (**b**) R channel; (**c**) G channel; (**d**) B channel.

4.2. Coefficient of Adjacent Pixels

In natural images, adjacent pixels typically exhibit high similarity in intensity values, resulting in strong correlations in horizontal, vertical, and diagonal directions. This inherent redundancy facilitates statistical analysis and potential cryptanalysis. An effective image encryption algorithm should significantly reduce or eliminate such correlations,

thereby impeding any attempt to extract useful information from ciphertext images through statistical means.

The correlation coefficient between adjacent pixels is mathematically defined as

$$\begin{cases} r_{xy} = \frac{\mathrm{cov}(x,y)}{\sqrt{D(x)}\sqrt{D(y)}}, \\ \mathrm{cov}(x,y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))(y_i - E(y)), \\ D(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))^2, \\ E(x) = \frac{1}{N}\sum_{i=1}^{N} x_i, \end{cases} \tag{40}$$

where x_i and y_i denote the intensity values of the i-th pair of neighboring pixels in horizontal, vertical, diagonal, or anti-diagonal directions, and N is the total number of such pixel pairs. Here, $\mathrm{cov}(x,y)$ represents the covariance between pixel intensities x and y, $D(x)$ and $D(y)$ denote their variances, and $E(x)$ and $E(y)$ are their respective expected values. The correlation coefficient r_{xy} thus quantifies the degree of linear dependency between adjacent pixel pairs.

To visually demonstrate the decorrelation effect achieved by the proposed encryption algorithm, the scatter plots in Figure 4 compare the pixel correlations of plaintext images with their corresponding ciphertext images across different directions. As shown, plaintext images exhibit tight clustering along the diagonal line, indicating strong correlations, whereas ciphertext images display random distributions, signifying that adjacent pixel correlations are effectively disrupted by the encryption process.

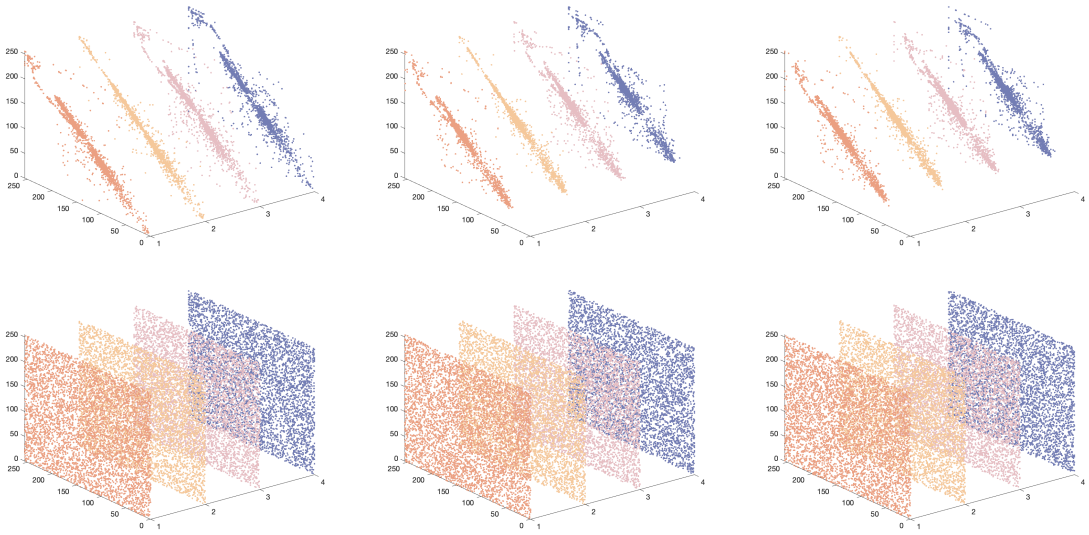


Figure 4. Correlation of adjacent pixels in plaintext and ciphertext images across different directions.

4.3. Information Entropy

Another key metric for evaluating the distribution of grayscale values in an image and measuring the randomness of image information is information entropy, which can be expressed as

$$H(m) = -\sum_{i=1}^{L} p(m_i)\log_2 p(m_i) \tag{41}$$

where L is the total number of symbols $m(i) \in m$ and $p(m_i)$ denotes the probability of the symbols. The experimental results are shown in Table 4. We can see that the experimental

results are close to the ideal value of 8, so the proposed algorithm has good information entropy properties.

Table 4. Information entropy results for various test images.

Filename	Description	Size	Type	H_CR	H_CG	H_CB	H_CAvg
2.1.01	San Diego (Miramar NAS)	512	Color	7.9994	7.9993	7.9992	7.9993
2.1.02	San Diego	512	Color	7.9993	7.9993	7.9993	7.9993
2.1.03	San Francisco (Golden Gate)	512	Color	7.9993	7.9993	7.9994	7.9994
2.1.04	Oakland	512	Color	7.9994	7.9993	7.9993	7.9993
2.1.05	San Diego (North Island NAS)	512	Color	7.9993	7.9994	7.9993	7.9994
2.1.06	Woodland Hills, Ca.	512	Color	7.9993	7.9992	7.9994	7.9993
2.1.07	Foster City, Ca.	512	Color	7.9993	7.9993	7.9994	7.9993
2.1.08	San Diego	512	Color	7.9994	7.9993	7.9992	7.9993
2.1.09	San Diego (Point Loma)	512	Color	7.9994	7.9994	7.9993	7.9993
2.1.10	San Diego (Shelter Island)	512	Color	7.9993	7.9993	7.9993	7.9993
2.1.11	Earth from space	512	Color	7.9993	7.9992	7.9994	7.9993
2.1.12	San Diego (Downtown)	512	Color	7.9994	7.9993	7.9992	7.9993
2.2.01	San Diego	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.02	San Diego	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.03	San Diego	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.04	Richmond, Ca.	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.05	San Diego (Miramar NAS)	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.06	San Francisco (Bay Bridge)	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.07	Oakland	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.08	San Diego	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.09	San Francisco	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.10	Richmond and San Rafael	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.11	Stockton	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.12	San Francisco and Oakland	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.13	Stockton	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.14	Shreveport	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.15	San Francisco	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.16	San Francisco	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.17	San Francisco	1024	Color	7.9999	7.9998	7.9998	7.9998
2.2.18	Stockton	1024	Color	7.9998	7.9999	7.9998	7.9998
2.2.19	Stockton	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.20	Stockton	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.21	San Francisco	1024	Color	7.9998	7.9998	7.9999	7.9998
2.2.22	San Francisco and Oakland	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.23	San Diego	1024	Color	7.9998	7.9998	7.9998	7.9998
2.2.24	Stockton	1024	Color	7.9998	7.9998	7.9998	7.9998
4.1.01	Female (NTSC test image)	256	Color	7.9978	7.9971	7.9972	7.9973
4.1.02	Couple (NTSC test image)	256	Color	7.9974	7.9972	7.997	7.9972
4.1.03	Female (from Bell Labs?)	256	Color	7.9965	7.9973	7.9975	7.9971
4.1.04	Female	256	Color	7.9969	7.997	7.9973	7.9971
4.1.05	House	256	Color	7.9974	7.9976	7.9972	7.9974
4.1.06	Tree	256	Color	7.997	7.9969	7.9973	7.9971
4.1.07	Jelly beans	256	Color	7.9976	7.9973	7.9974	7.9975
4.1.08	Jelly beans	256	Color	7.9969	7.9973	7.9977	7.9973
4.2.01	Splash	512	Color	7.9994	7.9993	7.9993	7.9993
4.2.03	Mandrill (a.k.a. Baboon)	512	Color	7.9993	7.9993	7.9993	7.9993
4.2.05	Airplane (F-16)	512	Color	7.9994	7.9994	7.9993	7.9993
4.2.06	Sailboat on lake	512	Color	7.9992	7.9994	7.9993	7.9993
4.2.07	Peppers	512	Color	7.9993	7.9993	7.9993	7.9993
house	House	512	Color	7.9992	7.9992	7.9994	7.9992

4.4. Differential Statistical Analysis

Differential statistical analysis is a critical metric for evaluating the resistance of image encryption algorithms against differential attacks. In particular, two widely adopted indicators, the Number of Pixel Change Rate (NPCR) and the Unified Average Changing Intensity (UACI), are used to quantify the sensitivity of the encryption algorithm to slight changes in the plaintext image. An ideal encryption algorithm should ensure that even a single-pixel modification in the plaintext image results in a significant and unpredictable change in the corresponding ciphertext image.

Mathematically, NPCR and UACI between two ciphertext images are defined as

$$\begin{cases} NPCR = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} D(i,j) \times 100\%, \\ UACI = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} \frac{|v_1(i,j) - v_2(i,j)|}{255} \times 100\%, \end{cases} \tag{42}$$

where $H \times W$ denotes the size of the image, and $v_1(i,j)$ and $v_2(i,j)$ are the pixel intensities of the two ciphertext images generated from plaintext images that differ by only a single pixel. The binary function $D(i,j)$ is defined as

$$D(i,j) = \begin{cases} 0, & \text{if } v_1(i,j) = v_2(i,j), \\ 1, & \text{if } v_1(i,j) \neq v_2(i,j). \end{cases} \tag{43}$$

The ideal theoretical values of NPCR and UACI for 8-bit grayscale or color images are approximately 99.6094% and 33.4635%, respectively. Values approaching these theoretical thresholds indicate high sensitivity of the encryption algorithm to plaintext changes, which is crucial for resisting differential attacks.

In this study, NPCR and UACI were computed for a series of standard test images from various open-source datasets. For each image, a single pixel in the plaintext was modified, and the corresponding encrypted images were compared across the red (R), green (G), and blue (B) channels. The experimental results are summarized in Table 5.

The data reveal that the proposed algorithm consistently achieves NPCR values around 99.6% and UACI values close to 33% across different images and resolutions. These results demonstrate that the algorithm exhibits excellent diffusion properties, ensuring that even minimal changes in the plaintext propagate extensively throughout the ciphertext. Consequently, the proposed encryption scheme demonstrates strong resistance to differential attacks.

Table 5. NPCR and UACI results for various test images.

Filename	NPCR_CR	NPCR_CG	NPCR_CB	NPCR_Avg	UACI_CR	UACI_CG	UACI_CB	UACI_Avg
2.1.01	0.9963	0.9961	0.9962	0.9962	0.2857	0.2869	0.2712	0.2813
2.1.02	0.9962	0.9961	0.996	0.9961	0.3039	0.2826	0.3048	0.2971
2.1.03	0.9961	0.9962	0.996	0.9961	0.3769	0.2586	0.3034	0.313
2.1.04	0.9959	0.9959	0.996	0.9959	0.2945	0.2645	0.2886	0.2825
2.1.05	0.996	0.9962	0.9962	0.9961	0.3019	0.2906	0.2878	0.2934
2.1.06	0.9962	0.9959	0.9962	0.9961	0.2818	0.2926	0.2843	0.2862
2.1.07	0.9961	0.9963	0.996	0.9961	0.2676	0.2891	0.3408	0.2991
2.1.08	0.9962	0.9961	0.9961	0.9961	0.2747	0.3191	0.3371	0.3103
2.1.09	0.996	0.996	0.9959	0.996	0.2683	0.2906	0.3466	0.3018
2.1.10	0.9959	0.9962	0.9961	0.9961	0.274	0.2977	0.3386	0.3034
2.1.11	0.9962	0.9962	0.9961	0.9962	0.2722	0.306	0.2689	0.2824
2.1.12	0.9961	0.996	0.9961	0.9961	0.2755	0.3127	0.3394	0.3092
2.2.01	0.9961	0.996	0.9961	0.9961	0.322	0.3212	0.282	0.3084

Table 5. *Cont.*

Filename	NPCR_CR	NPCR_CG	NPCR_CB	NPCR_Avg	UACI_CR	UACI_CG	UACI_CB	UACI_Avg
2.2.02	0.9961	0.9961	0.9962	0.9961	0.2679	0.29	0.3456	0.3012
2.2.03	0.996	0.9961	0.996	0.9961	0.276	0.3165	0.3433	0.312
2.2.04	0.9961	0.9961	0.996	0.9961	0.2763	0.2951	0.3441	0.3052
2.2.05	0.996	0.9961	0.9961	0.9961	0.2846	0.2826	0.2764	0.2812
2.2.06	0.9962	0.9961	0.9961	0.9961	0.2642	0.2984	0.3352	0.2993
2.2.07	0.9961	0.9961	0.9961	0.9961	0.2807	0.3032	0.34	0.308
2.2.08	0.9961	0.9961	0.9961	0.9961	0.3173	0.3178	0.2843	0.3065
2.2.09	0.996	0.9961	0.996	0.996	0.2695	0.29	0.3445	0.3013
2.2.10	0.996	0.9961	0.996	0.996	0.2688	0.2918	0.3432	0.3013
2.2.11	0.9961	0.9961	0.9962	0.9961	0.266	0.2696	0.3299	0.2885
2.2.12	0.996	0.9961	0.9962	0.9961	0.2634	0.261	0.3168	0.2804
2.2.13	0.9961	0.996	0.9961	0.9961	0.3028	0.2804	0.2775	0.2869
2.2.14	0.996	0.9961	0.9961	0.9961	0.2661	0.2615	0.306	0.2779
2.2.15	0.9963	0.9961	0.9961	0.9961	0.2641	0.264	0.3273	0.2852
2.2.16	0.9961	0.9961	0.9961	0.9961	0.2681	0.2605	0.3011	0.2766
2.2.17	0.9962	0.9961	0.9961	0.9961	0.274	0.2661	0.3085	0.2829
2.2.18	0.9961	0.996	0.9961	0.9961	0.2675	0.2691	0.3256	0.2874
2.2.19	0.996	0.9962	0.9962	0.9961	0.2676	0.2757	0.3331	0.2921
2.2.20	0.996	0.9961	0.9962	0.9961	0.265	0.2655	0.3182	0.2829
2.2.21	0.9961	0.9961	0.996	0.996	0.2823	0.2932	0.3388	0.3047
2.2.22	0.9961	0.9961	0.996	0.9961	0.2722	0.2937	0.343	0.303
2.2.23	0.9961	0.9962	0.9961	0.9961	0.2659	0.2893	0.3448	0.3
2.2.24	0.996	0.9961	0.9961	0.9961	0.304	0.2701	0.2812	0.2851
4.1.01	0.9958	0.9964	0.9965	0.9962	0.3197	0.3647	0.3745	0.353
4.1.02	0.9964	0.9963	0.9962	0.9963	0.3825	0.4106	0.4182	0.4037
4.1.03	0.9957	0.9961	0.9956	0.9958	0.2701	0.2682	0.2674	0.2686
4.1.04	0.9958	0.9957	0.9965	0.996	0.3117	0.3068	0.2747	0.2977
4.1.05	0.996	0.9957	0.9958	0.9958	0.2735	0.2986	0.3143	0.2955
4.1.06	0.9963	0.9962	0.9957	0.9961	0.3019	0.3422	0.3166	0.3202
4.1.07	0.9958	0.9961	0.9964	0.9961	0.3099	0.3259	0.2804	0.3054
4.1.08	0.996	0.9967	0.9961	0.9963	0.3082	0.3192	0.2829	0.3034
4.2.01	0.996	0.996	0.9963	0.9961	0.3418	0.3563	0.3197	0.3392
4.2.03	0.9961	0.9961	0.996	0.9961	0.299	0.2863	0.3126	0.2993
4.2.05	0.9961	0.996	0.996	0.996	0.3193	0.3312	0.327	0.3258
4.2.06	0.9959	0.9961	0.9962	0.9961	0.2791	0.3426	0.3433	0.3217
4.2.07	0.9961	0.9959	0.9959	0.996	0.2899	0.3396	0.3388	0.3227
house	0.9963	0.9958	0.9962	0.9961	0.3017	0.3126	0.3124	0.3089

4.5. Image Quality Analysis

To further evaluate the visual quality and structural integrity of decrypted images, the mean squared error (MSE), peak signal-to-noise ratio (PSNR), and structural similarity index (SSIM) were employed.

The MSE is defined as

$$\mathrm{MSE} = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(I(i,j) - I'(i,j) \right)^2, \tag{44}$$

where I and I' denote the original and decrypted images of size $M \times N$, respectively. A lower MSE indicates higher similarity between I and I'.

The PSNR, derived from MSE, is given by

$$\mathrm{PSNR} = 10 \cdot \log_{10} \left(\frac{\mathrm{MAX}^2}{\mathrm{MSE}} \right), \tag{45}$$

where MAX is the maximum possible pixel value (255 for 8-bit images). A higher PSNR reflects better reconstruction quality and reduced noise distortion.

The SSIM metric measures perceptual similarity and is defined as

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{46}$$

where μ_x and μ_y are the mean intensities of images x and y, σ_x and σ_y are their standard deviations, σ_{xy} is the covariance, and C_1 and C_2 are small constants to stabilize the division.

As presented in Table 6, the average MSE values of the decrypted images range from 24.64 to 174.58, yielding corresponding PSNR values within 25.59–34.75 dB. Images such as *4.1.02* achieve the highest PSNR (34.75 dB), demonstrating accurate reconstruction with minimal distortion.

Meanwhile, Table 7 shows that the SSIM values of encrypted images remain close to zero (below 0.012), indicating a high degree of structural decorrelation between the encrypted and plaintext images, which effectively conceals perceptual content during encryption.

These results confirm that the proposed scheme ensures robust encryption security by suppressing structural information in ciphertexts, while maintaining high-quality image recovery upon decryption.

Table 6. MSE and PSNR results for various test images

Filename	MSE_R	MSE_G	MSE_B	MSE_Avg	PSNR_R	PSNR_G	PSNR_B	PSNR_Avg
2.1.01	115.3963	92.7205	143.4964	117.2044	27.5089	28.459	26.5624	27.5101
2.1.02	149.6673	130.0471	170.5832	150.0992	26.3795	26.9898	25.8114	26.3936
2.1.03	35.6215	131.6412	174.7076	113.9901	32.6137	26.9369	25.7077	28.4194
2.1.04	92.2015	130.921	165.0415	129.388	28.4834	26.9607	25.9549	27.133
2.1.05	115.733	107.3988	157.738	126.9566	27.4962	27.8208	26.1514	27.1562
2.1.06	139.7274	141.0925	132.727	137.849	26.678	26.6358	26.9012	26.7383
2.1.07	126.9915	163.1052	193.0151	161.0373	27.0931	26.0061	25.2749	26.1247
2.1.08	148.2466	181.8745	191.3266	173.8159	26.421	25.5331	25.313	25.7557
2.1.09	107.2837	164.482	195.2686	155.6781	27.8255	25.9696	25.2245	26.3399
2.1.10	122.9607	165.5697	191.5297	160.02	27.2331	25.941	25.3084	26.1609
2.1.11	130.1045	85.2937	133.6447	116.3476	26.9879	28.8216	26.8713	27.5603
2.1.12	144.5868	177.5396	192.1222	171.4162	26.5295	25.6379	25.295	25.8208
2.2.01	104.2289	89.4356	148.9209	114.1951	27.9509	28.6157	26.4012	27.656
2.2.02	106.8286	164.5637	194.9323	155.4415	27.8439	25.9675	25.232	26.3478
2.2.03	149.0797	180.6178	194.043	174.5801	26.3966	25.5632	25.2518	25.7372
2.2.04	129.5561	164.6102	193.9574	162.7079	27.0062	25.9662	25.2537	26.0754
2.2.05	121.2284	96.9315	148.9244	122.3614	27.2948	28.2662	26.4011	27.3207
2.2.06	138.8768	170.9696	190.588	166.8115	26.7045	25.8016	25.3298	25.9453
2.2.07	150.692	172.6146	192.5303	171.9457	26.3499	25.76	25.2858	25.7986
2.2.08	106.4131	90.3006	153.5392	116.7509	27.8609	28.5739	26.2686	27.5678
2.2.09	118.2176	164.4282	194.4914	159.0458	27.404	25.971	25.2418	26.2056
2.2.10	126.5903	165.8016	194.0118	162.1346	27.1068	25.9349	25.2525	26.0981
2.2.11	134.3176	145.3584	187.7714	155.8158	26.8495	26.5064	25.3945	26.2501
2.2.12	117.8228	131.2753	181.1427	143.4136	27.4185	26.949	25.5506	26.6394
2.2.13	70.0395	82.786	155.8502	102.8919	29.6774	28.9512	26.2037	28.2774
2.2.14	122.5186	117.0882	175.2503	138.2857	27.2488	27.4457	25.6942	26.7962
2.2.15	124.2053	139.9086	186.7192	150.2777	27.1894	26.6724	25.4189	26.4269
2.2.16	102.2023	113.8248	172.2223	129.4165	28.0362	27.5684	25.7699	27.1248
2.2.17	122.3057	129.4309	176.1727	142.6364	27.2563	27.0104	25.6714	26.6461

Table 6. *Cont.*

Filename	MSE_R	MSE_G	MSE_B	MSE_Avg	PSNR_R	PSNR_G	PSNR_B	PSNR_Avg
2.2.18	132.2896	141.9878	185.6139	153.2971	26.9155	26.6083	25.4447	26.3228
2.2.19	140.2621	152.8728	189.2467	160.7939	26.6614	26.2875	25.3605	26.1031
2.2.20	118.6761	133.5933	181.6862	144.6519	27.3872	26.873	25.5376	26.5992
2.2.21	124.6644	162.0091	191.6422	159.4386	27.1734	26.0354	25.3059	26.1716
2.2.22	127.1823	166.3269	193.7634	162.4242	27.0865	25.9212	25.2581	26.0886
2.2.23	107.5325	164.8097	194.5041	155.6154	27.8154	25.961	25.2415	26.3393
2.2.24	69.2189	91.5905	158.9835	106.5976	29.7286	28.5123	26.1173	28.1194
4.1.01	65.0755	42.8132	37.3077	48.3988	29.9966	31.815	32.4128	31.4082
4.1.02	33.4481	21.7413	18.7389	24.6428	32.8871	34.7579	35.4034	34.3495
4.1.03	127.3148	129.5744	133.6162	130.1685	27.082	27.0056	26.8722	26.9866
4.1.04	119.0783	89.1085	114.7772	107.6547	27.3725	28.6316	27.5322	27.8454
4.1.05	136.5734	123.0152	130.8857	130.1581	26.7771	27.2312	26.9619	26.9901
4.1.06	121.7899	114.7762	132.174	122.9134	27.2747	27.5323	26.9193	27.2421
4.1.07	168.561	170.4008	131.0764	156.6794	25.8632	25.8161	26.9556	26.2116
4.1.08	164.16	160.5918	116.8781	147.21	25.9781	26.0736	27.4535	26.5017
4.2.01	165.3986	61.0408	69.4279	98.6224	25.9455	30.2746	29.7155	28.6452
4.2.03	126.7614	118.1083	102.7166	115.8621	27.1009	27.408	28.0144	27.5078
4.2.05	166.538	166.8156	179.3667	170.9068	25.9157	25.9084	25.5934	25.8058
4.2.06	120.1247	113.8343	104.6481	112.869	27.3345	27.5681	27.9335	27.612
4.2.07	139.0956	105.8134	56.6829	100.5307	26.6977	27.8854	30.5963	28.3931
house	144.7683	157.4318	131.4148	144.5383	26.5241	26.1599	26.9444	26.5428

Table 7. SSIM result for various test images.

Filename	Description	Size	Type	SSIMR	SSIMG	SSIMB	SSIM_Avg
2.1.01	San Diego (Miramar NAS)	512	Color	0.0104	0.0104	0.0104	0.0104
2.1.02	San Diego	512	Color	0.0109	0.0109	0.0109	0.0109
2.1.03	San Francisco (Golden Gate)	512	Color	0.0061	0.0061	0.0061	0.0061
2.1.04	Oakland	512	Color	0.0091	0.0091	0.0091	0.0091
2.1.05	San Diego (North Island NAS)	512	Color	0.0104	0.0104	0.0104	0.0104
2.1.06	Woodland Hills, Ca.	512	Color	0.011	0.011	0.011	0.011
2.1.07	Foster City, Ca.	512	Color	0.0102	0.0102	0.0102	0.0102
2.1.08	San Diego	512	Color	0.0103	0.0103	0.0103	0.0103
2.1.09	San Diego (Point Loma)	512	Color	0.0106	0.0106	0.0106	0.0106
2.1.10	San Diego (Shelter Island)	512	Color	0.0115	0.0115	0.0115	0.0115
2.1.11	Earth from space	512	Color	0.0111	0.0111	0.0111	0.0111
2.1.12	San Diego (Downtown)	512	Color	0.0111	0.0111	0.0111	0.0111
2.2.01	San Diego	1024	Color	0.0086	0.0086	0.0086	0.0086
2.2.02	San Diego	1024	Color	0.0106	0.0106	0.0106	0.0106
2.2.03	San Diego	1024	Color	0.0105	0.0105	0.0105	0.0105
2.2.04	Richmond, Ca.	1024	Color	0.0102	0.0102	0.0102	0.0102
2.2.05	San Diego (Miramar NAS)	1024	Color	0.0106	0.0106	0.0106	0.0106
2.2.06	San Francisco (Bay Bridge)	1024	Color	0.0108	0.0108	0.0108	0.0108
2.2.07	Oakland	1024	Color	0.0106	0.0106	0.0106	0.0106
2.2.08	San Diego	1024	Color	0.0084	0.0084	0.0084	0.0084
2.2.09	San Francisco	1024	Color	0.0106	0.0106	0.0106	0.0106
2.2.10	Richmond and San Rafael	1024	Color	0.0108	0.0108	0.0108	0.0108
2.2.11	Stockton	1024	Color	0.0103	0.0103	0.0103	0.0103
2.2.12	San Francisco and Oakland	1024	Color	0.011	0.011	0.011	0.011
2.2.13	Stockton	1024	Color	0.0092	0.0092	0.0092	0.0092
2.2.14	Shreveport	1024	Color	0.0112	0.0112	0.0112	0.0112
2.2.15	San Francisco	1024	Color	0.0105	0.0105	0.0105	0.0105
2.2.16	San Francisco	1024	Color	0.0102	0.0102	0.0102	0.0102

Table 7. *Cont.*

Filename	Description	Size	Type	SSIMR	SSIMG	SSIMB	SSIM_Avg
2.2.17	San Francisco	1024	Color	0.01	0.01	0.01	0.01
2.2.18	Stockton	1024	Color	0.0101	0.0101	0.0101	0.0101
2.2.19	Stockton	1024	Color	0.0105	0.0105	0.0105	0.0105
2.2.20	Stockton	1024	Color	0.0109	0.0109	0.0109	0.0109
2.2.21	San Francisco	1024	Color	0.0094	0.0094	0.0094	0.0094
2.2.22	San Francisco and Oakland	1024	Color	0.0108	0.0108	0.0108	0.0108
2.2.23	San Diego	1024	Color	0.0105	0.0105	0.0105	0.0105
2.2.24	Stockton	1024	Color	0.0091	0.0091	0.0091	0.0091
4.1.01	Female (NTSC test image)	256	Color	0.0083	0.0083	0.0083	0.0083
4.1.02	Couple (NTSC test image)	256	Color	0.0063	0.0063	0.0063	0.0063
4.1.03	Female (from Bell Labs?)	256	Color	0.0102	0.0102	0.0102	0.0102
4.1.04	Female	256	Color	0.0109	0.0109	0.0109	0.0109
4.1.05	House	256	Color	0.0087	0.0087	0.0087	0.0087
4.1.06	Tree	256	Color	0.01	0.01	0.01	0.01
4.1.07	Jelly beans	256	Color	0.0098	0.0098	0.0098	0.0098
4.1.08	Jelly beans	256	Color	0.0099	0.0099	0.0099	0.0099
4.2.01	Splash	512	Color	0.0096	0.0096	0.0096	0.0096
4.2.03	Mandrill (a.k.a. Baboon)	512	Color	0.0103	0.0103	0.0103	0.0103
4.2.05	Airplane (F-16)	512	Color	0.0101	0.0101	0.0101	0.0101
4.2.06	Sailboat on lake	512	Color	0.0106	0.0106	0.0106	0.0106
4.2.07	Peppers	512	Color	0.0103	0.0103	0.0103	0.0103
house	House	512	Color	0.0101	0.0101	0.0101	0.0101

4.6. Sensitivity Analysis

In this section, the performance metrics of the algorithm are analyzed in terms of both key and plaintext sensitivity, respectively. The security algorithm should be highly sensitive, which means that if there is a slight change in the key or plaintext image information during encryption or decryption, it will have a huge impact on the result of the subsequent encryption.

4.6.1. Analysis of Sensitivity to the Key

Key sensitivity is evaluated by examining the ciphertexts produced when encrypting the same plaintext image with two keys that differ only by a minute perturbation. Specifically, the plaintext image is encrypted using the original key, denoted as key, and then re-encrypted using perturbed keys defined as $key + 10^{-9}$, $key + 10^{-10}$, $key + 10^{-11}$, $key + 10^{-12}$, $key + 10^{-13}$, and $key + 10^{-14}$, respectively. The differences between the resulting ciphertexts are quantified using the NPCR and UACI metrics, which are defined in Equation (42).

As shown in Table 8, even a minimal perturbation of the encryption key results in ciphertext images with significant differences, as reflected by NPCR and UACI values approaching their theoretical ideals of 99.6094% and 33.4635%, respectively. These findings demonstrate that the proposed encryption algorithm exhibits a high sensitivity to slight key variations, thereby ensuring strong resistance against differential attacks.

4.6.2. Analysis of Plaintext Sensitivity

Plaintext sensitivity refers to the extent of variation in the ciphertext resulting from minor changes in the plaintext. An encryption algorithm with poor plaintext sensitivity may be vulnerable to plaintext attacks, as adversaries could exploit the relationships between plaintext–ciphertext pairs to infer the encryption mechanism. Therefore, strong plaintext sensitivity is essential for resisting such attacks.

In this section, we evaluate the plaintext sensitivity of the proposed algorithm by modifying the pixel value of the plaintext image by 1 at four distinct positions: $(\frac{1}{4}H, \frac{1}{4}W)$, $(\frac{1}{4}H, \frac{3}{4}W)$, $(\frac{3}{4}H, \frac{1}{4}W)$, and $(\frac{3}{4}H, \frac{3}{4}W)$. The resulting ciphertexts are then compared with the original ciphertext using NPCR and UACI metrics. The results, summarized in Table 9, demonstrate that even a single-pixel modification yields NPCR values approaching the ideal 99.6094% and UACI values near the theoretical 33.4635%.

Table 8. Key sensitivity test of NPCR and UACI.

Filename	−9		−10		−11		−12		−13		−14	
	NPCR	UACI	NPCR	UACI	NPCR	UACI	NPCR	UACI	NPCR	UACI	NPCR	UACI
2.1.02	0.9962	0.3344	0.9962	0.3351	0.9962	0.3346	0.996	0.3345	0.9961	0.3351	0.9961	0.3347
2.1.03	0.9961	0.3346	0.9961	0.3344	0.9961	0.3346	0.9961	0.3355	0.9961	0.3349	0.996	0.3349
2.1.04	0.9962	0.3347	0.9961	0.3349	0.996	0.3348	0.9961	0.3343	0.996	0.335	0.9962	0.3349
2.1.05	0.9962	0.3344	0.9962	0.3347	0.9962	0.3345	0.9961	0.3347	0.9961	0.3347	0.9962	0.3352
2.1.06	0.9962	0.3347	0.996	0.3347	0.9961	0.3344	0.9962	0.335	0.9961	0.3344	0.996	0.3344
2.1.08	0.9961	0.3346	0.9961	0.3343	0.9961	0.3343	0.996	0.3347	0.9961	0.3344	0.9961	0.3344
4.1.01	0.9961	0.3334	0.996	0.3359	0.9961	0.3334	0.9962	0.3339	0.9961	0.3346	0.9963	0.3352
4.1.02	0.9963	0.3345	0.9961	0.3345	0.9961	0.3344	0.996	0.3337	0.9961	0.3358	0.9962	0.3344
4.1.03	0.9958	0.3346	0.9965	0.3351	0.9959	0.3341	0.9958	0.3346	0.9959	0.3347	0.9962	0.3345
4.1.04	0.9962	0.3351	0.9963	0.3345	0.9961	0.3352	0.996	0.3345	0.9962	0.3339	0.996	0.3347
4.1.06	0.996	0.3351	0.9958	0.3359	0.9961	0.3354	0.996	0.3353	0.9958	0.3351	0.9961	0.3356
4.1.07	0.9958	0.3346	0.9962	0.3349	0.9962	0.334	0.9962	0.334	0.9961	0.3351	0.996	0.3336
4.1.08	0.9961	0.3347	0.9961	0.3342	0.9959	0.3341	0.9963	0.3346	0.9961	0.3344	0.9961	0.3341
4.2.01	0.996	0.3349	0.996	0.3345	0.9961	0.3346	0.996	0.3349	0.9961	0.3344	0.996	0.3346
4.2.03	0.9962	0.3348	0.9961	0.334	0.9961	0.3344	0.9961	0.3343	0.9959	0.3345	0.9959	0.3346
4.2.05	0.9961	0.3347	0.996	0.3343	0.996	0.3347	0.9961	0.3345	0.9962	0.3348	0.9962	0.3347
4.2.06	0.9961	0.3346	0.996	0.3343	0.9961	0.3345	0.9962	0.3346	0.9962	0.3347	0.9961	0.3346
4.2.07	0.996	0.3352	0.9963	0.3347	0.9961	0.335	0.9962	0.3345	0.9961	0.3344	0.9961	0.3344
house	0.9962	0.3344	0.9962	0.3348	0.9961	0.3348	0.996	0.3346	0.9963	0.3349	0.9962	0.3349

Table 9. The plaintext sensitivity test of NPCR and UACI.

Filename	1/4 × H, 1/4×W		1/4 × H, 3 × 1/4 × W		3 × 1/4 × H, 1/4 × W		3 × 1/4 × H, 3 × 1/4 × W	
	NPCR	UACI	NPCR	UACI	NPCR	UACI	NPCR	UACI
2.1.02	0.9962	0.3347	0.9961	0.3345	0.996	0.3342	0.996	0.3344
2.1.03	0.9962	0.3347	0.9962	0.3345	0.9961	0.3344	0.9962	0.3345
2.1.04	0.996	0.3351	0.9962	0.3348	0.996	0.335	0.9961	0.3346
2.1.05	0.9961	0.3342	0.996	0.3346	0.9962	0.3352	0.9961	0.3345
2.1.06	0.9962	0.3344	0.996	0.3347	0.996	0.3348	0.9961	0.3341
2.1.07	0.996	0.3344	0.9961	0.3348	0.9961	0.3337	0.9961	0.3352
2.1.08	0.9963	0.3347	0.996	0.3346	0.9961	0.3344	0.9961	0.3341
4.1.01	0.9962	0.335	0.996	0.3344	0.996	0.3347	0.9962	0.334
4.1.02	0.9964	0.335	0.996	0.3353	0.9961	0.335	0.9958	0.3339
4.1.03	0.9961	0.3346	0.9961	0.3351	0.9961	0.3346	0.9961	0.3345
4.1.04	0.9961	0.3355	0.996	0.3348	0.9962	0.3347	0.9963	0.3348
4.1.06	0.9961	0.3352	0.9961	0.3344	0.996	0.336	0.996	0.3343
4.1.07	0.9964	0.3348	0.9964	0.3348	0.9961	0.3322	0.9963	0.3338
4.1.08	0.9961	0.3343	0.9962	0.3351	0.9962	0.3347	0.996	0.3356
4.2.01	0.9961	0.3346	0.9961	0.3344	0.9961	0.3348	0.9961	0.3349
4.2.03	0.996	0.3345	0.9961	0.3347	0.9962	0.3345	0.9961	0.3344
4.2.05	0.996	0.3348	0.996	0.334	0.9961	0.335	0.9961	0.3343
4.2.06	0.9961	0.3348	0.9961	0.3345	0.996	0.3345	0.996	0.3346
4.2.07	0.9959	0.3348	0.9962	0.3344	0.996	0.3345	0.996	0.3341
house	0.996	0.3346	0.9961	0.3346	0.9961	0.3347	0.9962	0.3347

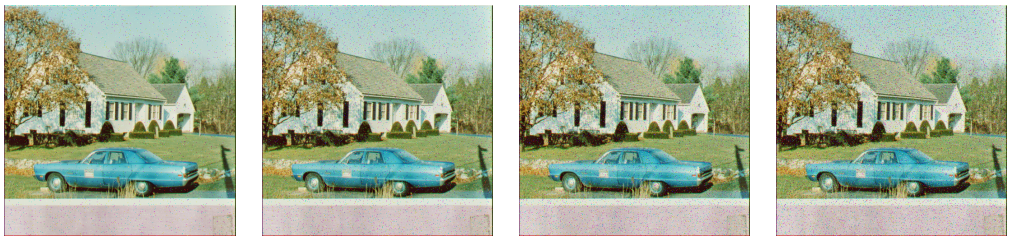
These results indicate that a slight alteration in plaintext leads to substantial differences in the corresponding ciphertext, effectively preventing adversaries from exploiting

plaintext–ciphertext correlations. Consequently, the proposed encryption scheme exhibits strong plaintext sensitivity and is robust against plaintext attacks.

4.7. Analysis of the Robustness

4.7.1. Cropping Attack

Cropping attacks are a common type of attack during information transmission, inevitably leading to the loss of ciphertext data. Therefore, the robustness of encryption algorithms is crucial to their security. By artificially cropping ciphertext into different sizes and positions and then decrypting it, we can determine the preservation of image information after decryption. The decryption results are shown in Figure 5.

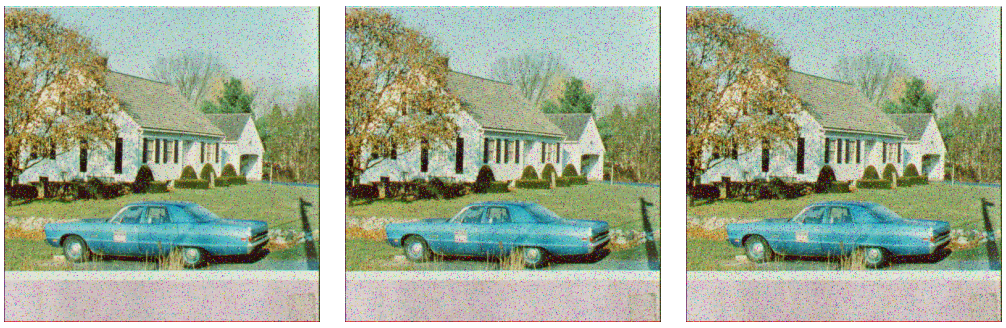


(**a**) (**b**) (**c**) (**d**)

Figure 5. Visualization of the proposed image encryption scheme: 'house' decrypted images with cropping. (**a**) Cropping ratio of 3%; (**b**) cropping ratio of 5%; (**c**) cropping ratio of 8%; (**d**) cropping ratio of 10%.

4.7.2. Salt and Pepper Noise

Salt and pepper noise effectively simulates noise interference and attacks during communications and is also effective in verifying the robustness of encryption algorithms. By adding varying degrees of salt and pepper noise to ciphertext and then decrypting it, we can determine the information retained in the decrypted image. The results are shown in Figure 6. These two experiments demonstrate the algorithm's excellent robustness, retaining key image information even after experiencing data loss.



(**a**) (**b**) (**c**)

Figure 6. Visualization of proposed image encryption scheme: 'house' decrypted images with salt and pepper noise. (**a**) Level of 3%; (**b**) level of 5%; (**c**) level of 8%.

5. Conclusions

This paper proposes an innovative facial image encryption scheme designed to address the increasingly serious issue of image privacy leakage in the era of big data and artificial intelligence, particularly the increased security risks posed by images containing sensitive information, such as faces. The scheme integrates facial detection technology with a multi-level encryption framework, employing a multi-task convolutional neural network (MTCNN) to precisely extract facial regions, ensuring processing efficiency and positioning

accuracy. Furthermore, a layered encryption process is constructed, including chaotic systems based on nonlinear dynamics (e.g., the 4D-NDS model), lightweight block permutation, dynamic RNA cryptography encoding (leveraging eight RNA rules and operations), and a bit diffusion mechanism to increase data complexity and unpredictability. Experimental validation demonstrates that the proposed scheme excels across multiple dimensions. In terms of security, it significantly reduces the correlation between adjacent pixels (nearly zero), achieves near-ideal information entropy (approximately 8.0), and demonstrates strong resistance to known-plaintext and chosen-plaintext attacks, demonstrated by high NPCR (approximately 99.6%) and UACI (approximately 33.46%) values. In terms of efficiency, the algorithm optimizes computational load, focusing on encryption of sensitive areas, reducing redundant processing overhead, and achieving high PSNR (e.g., 34.75 dB) in image reconstruction quality, ensuring practicality and robustness. Furthermore, the introduction of a plaintext-dependent key generation mechanism (based on dynamic feedback from image content) further enhances the system's attack resistance. Overall, this research not only provides a solution that balances security and efficiency, suitable for scenarios such as real-time surveillance and cloud biometrics, but also lays a technical foundation for image privacy protection in intelligent systems. Future applications could be extended to high-resolution video or resource-constrained devices, broadening its scope.

Author Contributions: X.C. is mainly responsible for the supervision and leadership of the planning and implementation of scientific research activities. X.C. and T.C. are mainly responsible for the research design, code writing and article writing. Y.L. is mainly responsible for literature search and format proofreading. X.Y. and W.C. are mainly responsible for Latex typesetting and drawing. All authors reviewed the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets used and analyzed during the current study available from the corresponding author on reasonable request. All data generated or analyzed during this study are included in this article.

Conflicts of Interest: The authors declare no competing interests.

References

1. Ye, C.; Tan, S.; Wang, J.; Shi, L.; Zuo, Q.; Feng, W. Social Image Security with Encryption and Watermarking in Hybrid Domains. *Entropy* **2025**, *27*, 276. [CrossRef]
2. Peng, Y.; Lan, Z.; Sun, K.; Xu, W. A simple color image encryption algorithm based on a discrete memristive hyperchaotic map and time-controllable operation. *Opt. Laser Technol.* **2023**, *165*, 109543. [CrossRef]
3. Gao, S.; Iu, H.H.; Erkan, U.; Şimşek, C.; Toktas, A.; Cao, Y.; Wu, R.; Mou, J.; Li, Q.; Wang, C. A 3D Memristive Cubic Map with Dual Discrete Memristors: Design, Implementation, and Application in Image Encryption. *IEEE Trans. Circuits Syst. Video Technol.* **2025**, *35*, 7706–7718. [CrossRef]
4. Gao, S.; Iu, H.H.C.; Mou, J.; Erkan, U.; Liu, J.; Wu, R.; Tang, X. Temporal action segmentation for video encryption. *Chaos Solitons Fractals* **2024**, *183*, 114958. [CrossRef]
5. Kocak, O.; Erkan, U.; Toktas, A.; Gao, S. PSO-based image encryption scheme using modular integrated logistic exponential map. *Expert Syst. Appl.* **2023**, *237*, 121452. [CrossRef]
6. Gao, S.; Ding, S.; Iu, H.H.C.; Erkan, U.; Toktas, A.; Simsek, C.; Wu, R.; Xu, X.; Cao, Y.; Mou, J. A three-dimensional memristor-based hyperchaotic map for pseudorandom number generation and multi-image encryption. *Chaos Interdiscip. J. Nonlinear Sci.* **2025**, *35*, 073105. [CrossRef] [PubMed]
7. Toktas, A.; Erkan, U.; Gao, S.; Pak, C. A robust bit-level image encryption based on Bessel map. *Appl. Math. Comput.* **2024**, *462*, 128340. [CrossRef]
8. Alghamdi, Y.; Munir, A. Evaluating ASCON Lightweight Encryption Algorithm for Image Encryption. *SN Comput. Sci.* **2024**, *5*, 1043. [CrossRef]

9. Liu, H.; Teng, L.; Zhang, Y.; Si, R.; Liu, P. Mutil-medical image encryption by a new spatiotemporal chaos model and DNA new computing for information security. *Expert Syst. Appl.* **2023**, *235*, 121090. [CrossRef]
10. Li, Q.; Wang, X.; Wang, H.; Ye, X.; Zhou, S.; Gao, S.; Shi, Y. A secure image protection algorithm by steganography and encryption using the 2D-TSCC. *Chin. Phys. B* **2021**, *30*, 110501. [CrossRef]
11. Akbacak, E.; Toktas, A.; Erkan, U.; Gao, S. MLMQ-IR: Multi-label multi-query image retrieval based on the variance of Hamming distance. *Knowl.-Based Syst.* **2024**, *283*, 111193.
12. Li, Q.; Ma, B.; Wang, X.; Wang, C.; Gao, S. Image Steganography in Color Conversion. *IEEE Trans. Circuits Syst. II Express Briefs* **2024**, *71*, 106–110. [CrossRef]
13. Gao, S.; Wu, R.; Wang, X.; Liu, J.; Li, Q.; Tang, X. EFR-CSTP: Encryption for face recognition based on the chaos and semi-tensor product theory. *Inf. Sci.* **2022**, *621*, 766–781.
14. Agrawal, S.; Madhu, B.R. A Modified Henon Map Based Image Encryption Framework. *SN Comput. Sci.* **2025**, *6*, 697. [CrossRef]
15. Erkan, U.; Gökrem, L.; Enginoğlu, S. Different applied median filter in salt and pepper noise. *Comput. Electr. Eng.* **2018**, *70*, 789–798. [CrossRef]
16. Erkan, U.; Thanh, D.N.; Hieu, L.M.; Enginoğlu, S. An Iterative Mean Filter for Image Denoising. *IEEE Access* **2019**, *7*, 167847–167859. [CrossRef]
17. Erkan, U.; Gökrem, L. A new method based on pixel density in salt and pepper noise removal. *Turk. J. Electr. Eng. Comput. Sci.* **2018**, *26*, 162–171. [CrossRef]
18. Chai, X.; Song, S.; Gan, Z.; Long, G.; Tian, Y.; He, X. CSENMT: A deep image compressed sensing encryption network via multi-color space and texture feature. *Expert Syst. Appl.* **2024**, *241*, 122562.
19. Erkan, U.; Toktas, A.; Enginoğlu, S.; Akbacak, E.; Thanh, D.N.H. An image encryption scheme based on chaotic logarithmic map and key generation using deep CNN. *Multimed. Tools Appl.* **2022**, *81*, 7365–7391. [CrossRef]
20. Li, C.; Tan, K.; Feng, B.; Lv, J. The Graph Structure of the Generalized Discrete Arnold's Cat Map. *IEEE Trans. Comput.* **2022**, *71*, 364–377.
21. Lai, Q.; Liu, Y.; Yang, L. Image encryption using memristive hyperchaos. *Appl. Intell.* **2023**, *23*, 22863–22881. [CrossRef]
22. Wen, W.; Huang, H.; Qi, S.; Zhang, Y.; Fang, Y. Joint Coverless Steganography and Image Transformation for Covert Communication of Secret Messages. *IEEE Trans. Netw. Sci. Eng.* **2024**, *11*, 2951–2962. [CrossRef]
23. Chai, X.; Tang, Z.; Gan, Z.; Lu, Y.; Wang, B.; Zhang, Y. SE-NDEND: A novel symmetric watermarking framework with neural network-based chaotic encryption for Internet of Medical Things. *Biomed. Signal Process. Control* **2024**, *90*, 105877. [CrossRef]
24. Zhang, Y.; Ye, X.; Xiao, X.; Xiang, T.; Li, H.; Cao, X. A Reversible Framework for Efficient and Secure Visual Privacy Protection. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 3334–3349. [CrossRef]
25. Liao, Y.; Lin, Y.; Li, Q.; Xing, Z.; Yuan, X. Lightweight Image Encryption Algorithm Using 4D-NDS: Compound Dynamic Diffusion and Single-Round Efficiency. *IEEE Access* **2025**, *13*, 74652–74662. [CrossRef]
26. Memis, S.; Enginoglu, S.; Erkan, U. Numerical Data Classification via Distance-Based Similarity Measures of Fuzzy Parameterized Fuzzy Soft Matrices. *IEEE Access* **2021**, *9*, 88583–88601. [CrossRef]
27. Lin, Y.; Xie, Z.; Chen, T.; Cheng, X.; Wen, H. Image privacy protection scheme based on high-quality reconstruction DCT compression and nonlinear dynamics. *Expert Syst. Appl.* **2024**, *257*, 124891. [CrossRef]
28. Mahmoudvand, S.; Ghazavi, M.R.; Farrokhabadi, A. Nonlinear dynamic modeling and chaos analysis of aircraft landing gear under two- and three-point landings. *Nonlinear Sci. Control Eng.* **2025**, 025280001. [CrossRef]
29. Moysis, L.; Lawnik, M.; Alexan, W.; Goudos, S.K.; Baptista, M.S.; Fragulis, G. Exploiting Circular Shifts for Efficient Chaotic Image Encryption. *IEEE Access* **2025**, *13*, 92997–93016. [CrossRef]
30. Alexan, W.; Hosny, K.; Gabr, M. A new fast multiple color image encryption algorithm. *Clust. Comput.* **2025**, *28*, 325. [CrossRef]
31. Alexan, W.; Youssef, M.; Hussein, H.H.; Ahmed, K.K.; Hosny, K.M.; Fathy, A.; Mansour, M.B. A new multiple image encryption algorithm using hyperchaotic systems, SVD, and modified RC5. *Sci. Rep.* **2025**, *15*, 9775. [CrossRef] [PubMed]
32. Gabr, M.; Korayem, Y.; Chen, Y.L.; Yee, P.L.; Ku, C.S.; Alexan, W. R 3—Rescale, Rotate, and Randomize: A Novel Image Cryptosystem Utilizing Chaotic and Hyper-Chaotic Systems. *IEEE Access* **2023**, *11*, 119284–119312. [CrossRef]
33. Alexan, W.; El-Damak, D.; Gabr, M. Image Encryption Based on Fourier-DNA Coding for Hyperchaotic Chen System, Chen-Based Binary Quantization S-Box, and Variable-Base Modulo Operation. *IEEE Access* **2024**, *12*, 21092–21113. [CrossRef]
34. Yan, S.; Wu, X.; Jiang, J. Dynamics analysis and predefined-time sliding mode synchronization of multi-scroll systems based on a single memristor model. *Chaos Solitons Fractals* **2025**, *196*, 116337. [CrossRef]
35. Zhang, H.; Yan, S.; Zhang, Y. A locally active memristive synapse-coupled HR-FN neural network and predefined time synchronization. *Neurocomputing* **2025**, *645*, 130540. [CrossRef]
36. Jiang, D.; Yan, S. Reversible adaptive chaotic multi-image privacy protection scheme for efficient batch encryption of various images. *Math. Comput. Simul.* **2025**, *238*, 201–221. [CrossRef]
37. Wu, X.; Yan, S. Adaptive sliding mode synchronization of a controllable grid-multi-wing chaotic system with parameter identification. *Appl. Math. Model.* **2026**, *150*, 116351. [CrossRef]

38. Yan, S.; Zhang, H.; Jiang, D. Multi-wing chaotic system based on smooth function and its predefined time synchronization. *Commun. Nonlinear Sci. Numer. Simul.* **2024**, *138*, 108178. [CrossRef]
39. Xie, E.Y.; Li, C.; Yu, S.; Lu, J. On the cryptanalysis of Fridrich's chaotic image encryption scheme. *Signal Process.* **2016**, *132*, 150–154. [CrossRef]
40. Liao, Y.; Lin, Y.; Zheng, X.; Yuan, X. Privacy Image Secrecy Scheme Based on Chaos-Driven Fractal Sorting Matrix and Fibonacci Q-Matrix. *Vis. Comput.* **2025**, *41*, 6931–6941. [CrossRef]
41. Zhang, L.; Lin, Y.; Yang, X.; Chen, T.; Cheng, X.; Cheng, W. From Sample Poverty to Rich Feature Learning: A New Metric Learning Method for Few-Shot Classification. *IEEE Access* **2024**, *12*, 124990–125002. [CrossRef]
42. Memiş, S.; Enginoğlu, S.; Erkan, U. A new classification method using soft decision-making based on an aggregation operator of fuzzy parameterized fuzzy soft matrices. *Turk. J. Electr. Eng. Comput. Sci.* **2021**, *30*, 871–890. [CrossRef]
43. Toktas, F.; Erkan, U.; Yetgin, Z. Cross-channel color image encryption through 2D hyperchaotic hybrid map of optimization test functions. *Expert Syst. Appl.* **2024**, *249*, 123583. [CrossRef]
44. Rao, H.; Zhao, Y.; Chen, H.P. Predicting chaotic system behavior using machine learning techniques. *Nonlinear Sci. Control Eng.* **2025**, 025290003. [CrossRef]
45. Gao, S.; Zhang, Z.; Iu, H.H.C.; Ding, S.; Mou, J.; Erkan, U.; Toktas, A.; Li, Q.; Wang, C.; Cao, Y. A Parallel Color Image Encryption Algorithm Based on a 2-D Logistic-Rulkov Neuron Map. *IEEE Internet Things J.* **2025**, *12*, 18115–18124. [CrossRef]
46. Gao, S.; Liu, J.; Iu, H.H.C.; Erkan, U.; Zhou, S.; Wu, R.; Tang, X. Development of a video encryption algorithm for critical areas using 2D extended Schaffer function map and neural networks. *Appl. Math. Model.* **2024**, *134*, 520–537. [CrossRef]
47. Wang, X.; Liu, C.; Jiang, D. A novel triple-image encryption and hiding algorithm based on chaos, compressive sensing and 3D DCT. *Inf. Sci.* **2021**, *574*, 505–527. [CrossRef]
48. Yamni, M.; Karmouni, H.; Sayyouri, M.; Qjidaa, H. Robust audio watermarking scheme based on fractional Charlier moment transform and dual tree complex wavelet transform. *Expert Syst. Appl.* **2022**, *203*, 117325. [CrossRef]
49. Cao, F.; Guo, D.; Wang, T.; Yao, H.; Li, J.; Qin, C. Universal screen-shooting robust image watermarking with channel-attention in DCT domain. *Expert Syst. Appl.* **2024**, *238*, 122062. [CrossRef]
50. Jamal, S.S.; Hazzazi, M.M.; Khan, M.F.; Bassfar, Z.; Aljaedi, A.; ul Islam, Z. Region of interest-based medical image encryption technique based on chaotic S-boxes. *Expert Syst. Appl.* **2024**, *238*, 122030. [CrossRef]
51. Zhang, Y. A new unified image encryption algorithm based on a lifting transformation and chaos. *Inf. Sci.* **2021**, *547*, 307–327. [CrossRef]
52. Xie, Z.; Xie, W.; Cheng, X.; Yuan, Z.; Cheng, W.; Lin, Y. Image Privacy Protection Communication Scheme by Fibonacci Interleaved Diffusion and Non-Degenerate Discrete Chaos. *Entropy* **2025**, *27*, 790. [CrossRef]
53. Zeng, W.; Zhang, C.; Xia, J.; Liang, X.; Lin, Y.; Li, Y.; Wang, S.; Yang, G. Chaotic 4D Modulation With Intrusion Detection for Secure Data Centers. *J. Light. Technol.* **2025**, *43*, 7530–7539. [CrossRef]
54. Zeng, W.; Zhang, C.; Liang, X.; Lin, Y.; Xia, J.; Li, Y. A Novel Secure Key Stream Generator Based on Chaotic Multistate Cellular Automata. *IEEE Internet Things J.* **2025**, *12*, 35705–35716. [CrossRef]
55. Zeng, W.; Zhang, C.; Liang, X.; Luo, Y.; Wang, X.; Qiu, K. Chaotic phase noise-like encryption based on geometric shaping for coherent data center interconnections. *Opt. Express* **2024**, *32*, 1595. [CrossRef] [PubMed]
56. Zeng, W.; Zhang, C.; Liang, X.; Xia, J.; Lin, Y.; Lin, Y. Intrusion detection-embedded chaotic encryption via hybrid modulation for data center interconnects. *Opt. Lett.* **2025**, *50*, 4450–4453. [CrossRef]
57. Winkler, T.; Rinner, B. Trustcam: Security and privacy-protection for an embedded smart camera based on trusted computing. In Proceedings of the 2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance, Boston, MA, USA, 29 August 2010–1 September 2010; pp. 593–600.
58. Zhao, R.; Zhang, Y.; Wen, W.; Lan, R.; Xiang, Y. E-TPE: E fficient T humbnail-P reserving E ncryption for Privacy Protection in Visual Sensor Networks. *ACM Trans. Sens. Netw.* **2024**, *20*, 1–26. [CrossRef]
59. Chai, X.; Ma, Y.; Wang, Y.; Gan, Z.; Zhang, Y. TPE-ADE: Thumbnail-preserving encryption based on adaptive deviation embedding for JPEG images. *IEEE Trans. Multimed.* **2023**, *26*, 6102–6116. [CrossRef]
60. Fan, L. Image pixelization with differential privacy. In Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy, Bergamo, Italy, 16–18 July 2018; pp. 148–162.
61. Ji, J.; Wang, H.; Huang, Y.; Wu, J.; Xu, X.; Ding, S.; Zhang, S.; Cao, L.; Ji, R. Privacy-preserving face recognition with learnable privacy budgets in frequency domain. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; pp. 475–491.
62. Chen, Z.; Zhu, T.; Xiong, P.; Wang, C.; Ren, W. Privacy preservation for image data: A gan-based method. *Int. J. Intell. Syst.* **2021**, *36*, 1668–1685. [CrossRef]
63. Wen, Y.; Liu, B.; Ding, M.; Xie, R.; Song, L. Identitydp: Differential private identification protection for face images. *Neurocomputing* **2022**, *501*, 197–211. [CrossRef]

64. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*. [CrossRef] [PubMed]
65. Deng, J.; Guo, J.; Zhou, Y.; Yu, J.; Kotsia, I.; Zafeiriou, S. Retinaface: Single-stage dense face localisation in the wild. *arXiv* **2019**, arXiv:1905.00641.
66. Yu, Z.; Huang, H.; Chen, W.; Su, Y.; Liu, Y.; Wang, X. Yolo-facev2: A scale and occlusion aware face detector. *Pattern Recognit.* **2024**, *155*, 110714. [CrossRef]
67. Zhang, K.; Zhang, Z.; Li, Z.; Qiao, Y. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Process. Lett.* **2016**, *23*, 1499–1503. [CrossRef]