mathematics

MDPI

Article

Multiple-Image Encryption Scheme Based on an N-Dimensional Chaotic Modular Model and Overlapping Block Permutation–Diffusion Using Newly Defined Operation

Ziqi Zhou [1], Xuemei Xu [1,*], Zhaohui Jiang [2] and Kehui Sun [1]

[1] School of Physics and Electronics, Central South University, Changsha 410083, China; zhouziqi@csu.edu.cn (Z.Z.)
[2] School of Automation, Central South University, Changsha 410083, China
* Correspondence: xxm999@csu.edu.cn

Abstract: Some existing chaotic maps have the drawbacks of a narrow range of chaotic parameters and discontinuities, which may be inherited by new chaotic systems generated from them as seed maps. We propose a chaotic model that can generate N-dimensional chaotic systems to overcome the problem. By fixing the original parameters of the seed map in the chaotic range, we then introduce new parameters and use modular operations to widen the range of the parameters and increase the complexity. Simulation results show that the generated chaotic system has good chaotic dynamics. Based on this chaotic model, we propose a multiple-image encryption algorithm that is not limited by image type, number, and size. The resistance to plaintext attacks is enhanced by a permutation–diffusion algorithm based on overlapping blocks. We design a newly defined lookup table operation based on Latin squares with enhanced nonlinearity and randomness. By adjusting the overlapping block parameters and the number of Latin squares, users can design different encryption levels to balance encryption efficiency and encryption effectiveness. The experimental results show that the proposed image encryption algorithm can effectively encrypt multiple images, and all the evaluation indexes reach the expected value.

Keywords: multiple-image; image encryption; new chaotic model; permutation–diffusion

MSC: 37N99; 68P25

Citation: Zhou, Z.; Xu, X.; Jiang, Z.; Sun, K. Multiple-Image Encryption Scheme Based on an N-Dimensional Chaotic Modular Model and Overlapping Block Permutation–Diffusion Using Newly Defined Operation. *Mathematics* **2023**, *11*, 3373. https://doi.org/10.3390/math11153373

Academic Editor: Lingfeng Liu

Received: 13 July 2023
Revised: 29 July 2023
Accepted: 30 July 2023
Published: 1 August 2023

1. Introduction

In the era of information technology, multimedia carriers have experienced rapid development due to their rich means of expressing information. Digital images can directly convey visual information while having small storage sizes, making them one of the most commonly used means of information communication [1]. However, images can be susceptible to illegal attacks, leading to privacy breaches and property losses, both during transmission and storage. Encrypting images into unrecognizable noisy images is an effective means of protection. Access without the correct key will be unable to extract valid information from them. Various types of image encryption schemes have been proposed nowadays, such as chaotic systems, fractal theory [2,3], quantum computing [4,5], and optical methods [6,7].

Among the aforementioned technologies, chaotic systems have received increasing attention and are widely used in image encryption algorithms. Chaos is a pseudo-random dynamic complex phenomenon generated by deterministic nonlinear systems. Chaotic systems exhibit characteristics such as sensitivity to control parameters and initial conditions, pseudo-randomness, ergodicity, and unpredictability, making them highly suitable for cryptography. Chaotic systems are usually classified into one-dimensional (1D) and high-dimensional (HD) chaotic systems. One-dimensional chaotic systems have characteristics

of simple structure, ease of implementation, and fast iteration speed. HD chaotic systems have characteristics of high complexity, multiple parameters, and long periods. Some encryption schemes directly utilize classical chaotic systems to complete the encryption. Kumar et al. [8] utilized the logistic map to generate chaotic sequences for key generation and then employed an improved shuffling technique to permute the image at the pixel level. Chen et al. [9] used the hyperchaotic Lorentz system to generate the key stream and proposed a nonsequential encryption mechanism, with improved permutation and diffusion methods as the core of encryption. However, some existing chaotic systems may exhibit defects such as a narrow and discontinuous range of chaotic parameters. Researchers have developed methods to improve chaotic systems to address these defects. Hua et al. [10] utilize sinusoidal functions as nonlinear transformations and apply them to the output of 1D chaotic maps to enhance complexity. Wang et al. [11] apply exponential and sinusoidal functions to both 1D and HD maps. However, due to the inherent inhomogeneity of the output from the nonlinear functions they use, the enhanced chaotic system also possesses the weakness of accumulating at the edges of the phase space. Modular arithmetic is used in chaotic systems because of its boundedness and uniform output. Hua et al. [12,13] improved the performance of the original chaotic maps by applying modular arithmetic to increase the chaotic complexity. However, this approach is limited to existing chaotic systems and does not create new ones. Zhou et al. [14] nonlinearly coupled two seed chaotic maps to improve the chaotic range using modular arithmetic, which is limited to 2D chaotic systems. Additionally, various chaotic maps have been employed as seed maps to generate new chaotic maps. Lin et al. [15] construct a crossed 2D hyperchaotic map using a nonlinear function and two chaotic maps with crossed structures. Sun [16] design a chaotic model based on cascaded modulation coupling. Gao [17] multiplies the outputs and employs nonlinear transformations to construct a 2D chaotic model. These methods typically retain the original parameters of the seed maps, which may inherit the disadvantages of the seed maps. Consequently, when one or more of the seed maps exist within a nonchaotic parameter space, the new chaotic system may also fail to enter the chaotic state.

Most image encryption algorithms employ a permutation–diffusion structure. The permutation phase involves rearranging the pixel positions in the image, making it challenging for attackers to identify the original pixel positions and the relationships between adjacent pixels. The diffusion phase propagates subtle changes throughout the entire image, introducing complex transformations to the pixel values. Diffusion enhances the complexity and unpredictability of the relationship between pixel values and encryption keys, improving resistance against plaintext attacks. Encryption can be performed at different levels, such as block level [18,19], pixel level [20,21], bit level [22,23], and DNA level [24]. Wang et al. [25] utilize a modified zigzag transform permutation at the pixel level, index sequence permutation, and dynamic encoding diffusion at the DNA level. Wei et al. [26] horizontally and vertically extend to a bit-level plane, exchanging row/column vectors for the permutation process. Bit-level and DNA-level permutations not only alter pixel positions but also pixel values, potentially requiring the generation of additional chaotic sequences. Hua et al. [27] partition the image into blocks equal to the block size during the disarrangement phase, evenly spreading pixels within blocks to other blocks. Wang et al. [28] divide the image into equal-sized blocks during the diffusion phase and select pixels from different sub-blocks using the Joseph problem, applying modular sum and XOR operations to change pixel values. Block-based operations can enhance encryption efficiency, but dividing the image into equal-sized blocks may introduce risks. The diffusion step transforms the plaintext image pixels using chaotic pixels to mask the pixels randomly. Common transformations include XOR operation, modular sum operation, and finite field operation. Wang et al. [29] combine XOR and modular sum operations to design four diffusion formulas, dynamically selected using chaotic sequences. To avoid information loss when the multiplier is 0, Xu et al. [30] utilize the multiplication table of GF(257) for

diffusion operations instead of the multiplication table of GF(2^8). Existing diffusion operations often exhibit monotonicity and vulnerability to attacks like differential attacks.

Image encryption technology finds applications in various fields such as personal, medical, industrial, and military, and the demand for image encryption is diverse. Firstly, different levels of confidentiality require different encryption requirements. Generally, as the security of encryption increases, the efficiency of encryption decreases. Therefore, finding a balance between encryption efficiency and security is crucial. The appropriate encryption scheme should be selected based on the desired level of confidentiality. Secondly, users may have diverse requirements in terms of the number, size, and type of images they need to encrypt. However, many existing image encryption schemes have limitations, such as only supporting a specific image type or images of the same size. Lastly, the design of encryption keys varies based on different needs. Users should have the option to use either a randomly generated key provided by the encryption scheme or a user-defined fixed key. In existing image encryption schemes, hash functions are often used to design keys to resist plaintext attacks, but this restricts user participation in the key design process.

To address the aforementioned issues, we propose a multidimensional chaos model based on modular arithmetic and a multiple-image encryption algorithm. In the proposed chaotic model, we fix the original parameters of the seed maps within the chaotic range, introduce new parameters through the nonlinear coupling of multiple outputs, and collapse the outputs using modular arithmetic. The chaotic model is not limited to some fixed number of dimensions, and it can generate chaotic systems of arbitrary dimensions. Users can design chaotic systems of varying dimensions based on the desired level of encryption confidentiality. We generate three chaotic systems using the model, with dimensions of two, three, and four, and validate their dynamics through simulation experiments, including phase diagrams, bifurcation diagrams, Lyapunov exponent diagrams, and sample entropy. The simulation results demonstrate that the generated chaotic systems possess a wide and continuous range of chaotic parameters, exhibiting good ergodicity and parameter sensitivity. In the proposed multiple-image encryption algorithm, we combine multiple images into a 1D array for encryption, making it adaptable to any image type, number, and size. For key design, we present two methods: the first method involves combining the hash value generated from plaintext images with random noise to generate the key, while the second method allows the user to set the key themselves. By incorporating chaotic systems, we achieve strong key sensitivity. We introduce a permutation–diffusion algorithm based on overlapping blocks, where permutation and diffusion mutually affect each other. This approach effectively reduces the correlation between adjacent pixels, significantly enhancing resistance against plaintext attacks. We provide adjustable parameters for block size and overlapping parts, enabling users to achieve different levels of security by adjusting these parameters. Additionally, we define a novel lookup table operation based on the Latin square. Through chaotic sequences, we generate several different Latin squares to reach dynamic operations in the form of row and column lookups. This enhances the nonlinearity and diversity of diffusion operations. We have conducted several simulation tests, and the experimental results demonstrate that the proposed image encryption algorithm effectively encrypts multiple images with strong security and efficiency.

The paper is organized as follows. In Section 2, a new chaotic model is introduced and three example chaotic systems are designed based on it. In Section 3, the performance of these chaotic systems is analyzed. In Section 4, the proposed new multi-image encryption algorithm is presented. In Section 5, the simulation results and the security of the algorithm are analyzed. In Section 6, conclusions about this thesis are given.

2. N-Dimensional Modular Chaotic Model

This section describes the construction of the N-dimensional coupled modular chaos model (ND-CMCM) and creates three different dimensional chaotic systems as examples by this model.

2.1. Construction of ND-CMCM

In order to improve the performance of chaotic maps, while users can construct suitable chaotic systems according to different needs, ND-CMCM is proposed in this paper.

Consider an N dimension of input $\mathbf{x} = (x_1, x_2, \ldots x_N)^T$. $F_i(x_i, \mu_i)$ $(i = 1, 2, \ldots N)$ are the N-selected 1D seed chaotic maps, where μ_i are the control parameters. The original control parameters of the seed chaotic maps are fixed at a number that can enter the chaotic state. Create new control parameters a_i $(i = 1, 2, \ldots N)$. We define the structure of ND-CMCM as Equation (1).

$$\begin{cases} x_{1,j+1} = \mathrm{mod}\left(a_1 F_1\left(x_{1,j}\right) + a_1 f_1\left(\mathbf{x}_j\right) + a_2 f_2\left(\mathbf{x}_j\right) \cdots + a_N f_N\left(\mathbf{x}_j\right), C\right) \\ x_{2,j+1} = \mathrm{mod}\left(a_2 F_2\left(x_{2,j}\right) + a_1 f_{N+1}\left(\mathbf{x}_j\right) + a_2 f_{N+2}\left(\mathbf{x}_j\right) \cdots + a_N f_{2N}\left(\mathbf{x}_j\right), C\right) \\ \quad \vdots \\ x_{N,j+1} = \mathrm{mod}\left(a_N F_N\left(x_{N,j}\right) + a_1 f_{N^2-N+1}\left(\mathbf{x}_j\right) + a_2 f_{N^2-N+2}\left(\mathbf{x}_j\right) \cdots + a_N f_{N^2}\left(\mathbf{x}_j\right), C\right) \end{cases} \tag{1}$$

where $\mathbf{x}_j$ is an N-dimensional vector which is the j-th state of the chaotic system iteration, $f_i(\mathbf{x})$ $(i = 1, 2, \ldots N^2)$ are nonlinear functions that couple multidimensional outputs, mod is the modular arithmetic, and C is the modulus coefficient. In the general case, the model can be simplified to Equation (2).

$$\begin{cases} x_{1,j+1} = \mathrm{mod}\left(a_1 F_1\left(x_{1,j}\right) + a_2 f_1\left(\mathbf{x}_j\right), C\right) \\ x_{2,j+1} = \mathrm{mod}\left(a_2 F_2\left(x_{2,j}\right) + a_3 f_2\left(\mathbf{x}_j\right), C\right) \\ \quad \vdots \\ x_{N-1,j+1} = \mathrm{mod}\left(a_{N-1} F_{N-1}\left(x_{N-1,j}\right) + a_N f_{N-1}\left(\mathbf{x}_j\right), C\right) \\ x_{N,j+1} = \mathrm{mod}\left(a_N F_N\left(x_{N,j}\right) + a_1 f_N\left(\mathbf{x}_j\right), C\right) \end{cases} \tag{2}$$

The chaotic system designed by this model has a more complex dynamical behavior than the seed chaotic maps. The modular arithmetic can keep the output in a finite range, which not only allows a_i to be set to any larger value but also increases the complexity of the chaotic system. Compared to other bounded functions, modular operations can make the output more uniform in phase space. The modulus coefficient C can be arbitrarily set to a relatively small number compared to the parameter a_i, which increases the diversity of the system. By utilizing various 1D chaotic systems and nonlinear functions, ND-CMCM can generate multiple chaotic systems, thus simplifying the design process and giving the user the freedom to design the desired chaotic system.

2.2. Examples of the Proposed Chaotic Model

In order to verify the excellent performance of ND-CMCM, we choose some classical 1D chaotic maps and some 1D chaotic maps from recent literature as examples, which are combined into three different dimensions of chaotic maps.

2.2.1. 2D Logistic–PWLCM Coupled Modular Map

The logistic map [31] is one of the most classical and commonly used chaotic maps. It is defined as Equation (3).

$$x_{n+1} = \mu x_n (1 - x_n) \tag{3}$$

where $x_n \in (0, 1)$, $\mu \in [0, 4]$ is a control parameter. When the parameter μ is in the interval (3.57, 4], the logistic map enters the chaotic state.

The piece-wise linear chaotic map (PWLCM) [32] is widely used for its simple structure and good traversability. PWLCM is defined as Equation (4).

$$x_{n+1} = T(x_n, p) = \begin{cases} x_n / p & 0 \le x_i < p \\ (x_n - p)/(0.5 - p) & p \le x_i < 0.5 \\ T(1 - x_n, p) & 0.5 \le x_i < p \end{cases} \tag{4}$$

where $x_n \in (0, 1)$, $p \in (0, 0.5)$ is the control parameter and the map is in the chaotic state within this range.

The 2D logistic–PWLCM coupled modular map (2D-LPCMM) is obtained by combining the logistic map and PWLCM, with the parameter μ of the logistic map fixed at 4 and the parameter p of the PWLCM fixed at 0.3. The expression is as Equation (5).

$$\begin{cases} x_{n+1} = \mathrm{mod}(4ax_n(1-x_n) + by_n(1-x_n), 1) \\ \begin{cases} y_{n+1} = \mathrm{mod}(by_n/0.3 + ax_n(1-y_n), 1) & 0 \le y_n < 0.3 \\ y_{n+1} = \mathrm{mod}(b(y_n-0.3)/0.2 + ax_n(1-y_n), 1) & 0.3 \le y_n < 0.5 \\ y_{n+1} = \mathrm{mod}(b(0.7-y_n)/0.2 + ax_n(1-y_n), 1) & 0.5 \le y_n < 0.7 \\ y_{n+1} = \mathrm{mod}(b(1-y_n)/0.3 + ax_n(1-y_n), 1) & 0.7 \le y_n < 1 \end{cases} \end{cases} \tag{5}$$

where $x_n \in (0, 1)$, a and b are the two newly created control parameters. Since the mod is a bounded function that collapses the output, a and b can be very large values. In this paper, we investigate the properties of 2D-LPCMM for a, b in the range [1, 1000].

2.2.2. 3D Cubic–Fraction–IST Coupled Modular Map

The form of the cubic map [33] can be expressed by Equation (6).

$$x_{n+1} = \rho x_n \left(1 - x_n^2\right) \tag{6}$$

where $x_n \in (0, 1)$, $\rho \in [0, 3]$ is a control parameter. When the parameter ρ is in the interval (2.59,3], the cubic map exhibits complex behavior.

The fraction map [34] was discovered in the study of evolutionary algorithms. This map is more complex than the logistic map and its expression is Equation (7).

$$x_{n+1} = \frac{1}{x^2 + 0.1} - px_n \tag{7}$$

where $x_n \in [-10.0025, 10.0025]$, $p \in [-0.999, 0.999]$ is a control parameter. In the interval $p \in [0.699, 0.1510] \cup [0.2470, 0.3590] \cup [0.3770, 0.5170] \cup [0.5680, 0.6280] \cup [0.6310, 0.6770]$, the system is chaotic.

The improved sine–tangent map (IST map) [35] is derived from the sine map, the tangent function, and the first class of Chebyshev polynomials. It is defined as follows.

$$x_{n+1} = \sin\left(\alpha \tan\left(3{x_n}^2 - 1.5\right)\right) \tag{8}$$

where $x_n \in [-1, 1]$, $\alpha \in (0, 8]$ is a control parameter and in this interval the map is chaotic.

The 3D cubic–fraction–IST coupled modular map (3D-CFICMM) is obtained by combining the cubic map, fraction map, and IST map, with the parameter ρ of the cubic map fixed at 3, the parameter p of the fraction map fixed at 0.5, and the parameter α of the IST fixed at 7. The expression is as Equation (9).

$$\begin{cases} x_{n+1} = \mathrm{mod}\left(3ax_n(1-{x_n}^2) + by_nz_n\cos(x_n), 2\right) \\ y_{n+1} = \mathrm{mod}\left(b/\left({y_n}^2 + 0.1\right) - 0.5by_n + cx_nz_n\cos(y_n), 2\right) \\ z_{n+1} = \mathrm{mod}\left(c\sin\left(7\tan\left(3{z_n}^2 - 1.5\right)\right) + ax_ny_n\cos(z_n), 2\right) \end{cases} \tag{9}$$

where $x_n \in (0, 2)$, a, b, and c are the three newly created control parameters. a, b, and c can be very large values. In this paper, we investigate the properties of 3D-CFICMM for $a, b, c \in [1, 1000]$.

2.2.3. 4D ICMIC–ICMIC–1DSP–1DCP Coupled Modular Map

The iterative chaotic map with infinite collapses (ICMIC) [36] is also a commonly used 1D chaotic system and is often used as a seed map to construct new functions. The formal definition of the ICMIC is given by Equation (10).

$$x_{n+1} = \sin\left(\frac{\beta}{x_n}\right) \tag{10}$$

where $x_n \in [-1, 1]$, $\beta \in (0, \infty)$ is a control parameter. This map has many periodic windows that lead to discontinuities in the chaotic range. For example, there are large periodic windows for the parameter $\beta \in (0, 1.8197) \cup (4.1411, 4.8143) \cup (7.5073, 7.9163)$. There are also many narrow period windows such as $\beta \in (2.5584, 2.5686)$.

The 1D sine powered chaotic map (1DSP) [37] was designed inspired by the sine map and the sine map was used to avoid exponential growth of the output. The 1DSP is defined as Equation (11).

$$x_{n+1} = (x_n(\alpha + 1))^{\sin(\beta\pi + x_i)} \tag{11}$$

where $x_n \in [0, 1]$, $\alpha \in (0, \infty)$ and $\beta \in [0, 1]$ are two control parameters. For $\alpha \in (2, \infty)$ and $\beta \in [0.011, 0.46]$, 1DSP exhibits continuous chaotic behavior.

The 1D cosine polynomial (1DCP) [38] is designed as a simple structure with highly chaotic behavior. The 1DCP is defined as Equation (12).

$$x_{n+1} = \cos\left(\mu\left(x_n^3 + x_n\right)\right) \tag{12}$$

where $x_n \in [-1, 1]$, $\mu \in (-\infty, \infty)$ is a control parameter. When the parameter μ is small, there is a narrow period window.

The 4D ICMIC–ICMIC–1DSP–1DCP coupled modular map (4D-IISCCMM) is obtained by combining the ICMIC c map, 1DSP, and 1DCP, with the parameter β of the ICMIC map fixed at 6 and 8, the parameter α and β of the 1DSP fixed at 4.4926 and 0.3306, and the parameter μ of the 1DCP fixed at 1000. ICMIC is used twice, implying that ND-CMCM does not necessarily require different seed chaotic maps. The expression is as Equation (13).

$$\begin{cases} x_{n+1} = \mathrm{mod}(a\sin(6/x_n) + b(z_n w_n + \sin(x_n y_n)), 3) \\ y_{n+1} = \mathrm{mod}(b\sin(8/y_n) + c(w_n x_n + \sin(y_n z_n)), 3) \\ z_{n+1} = \mathrm{mod}\left(cz_n(4.4926 + 1)^{\sin(0.3306\pi + z_n)} + d(x_n y_n + \sin(z_n w_n)), 3\right) \\ w_{n+1} = \mathrm{mod}\left(d\cos\left(1000\left(w_n^3 + w_n\right)\right) + a(y_n z_n + \sin(w_n x_n)), 3\right) \end{cases} \tag{13}$$

where $x_n \in (0, 3)$, a, b, c, and d are the four newly created control parameters. a, b, c, and d can be very large values. In this paper, we investigate the properties of 4D-IISCCMM for a, b, c, $d \in [1, 1000]$.

The 3D-CFICMM and 4D-IISCCMM suffer from computational overload. We cite them as examples to demonstrate the generation of chaotic systems with different dimensions, using various nonlinear function coupling methods and modulus coefficients. Therefore, these two chaotic systems are designed without significant consideration of efficiency. The computational workload of the generated chaotic system is related to the seed chaotic maps and the nonlinear function it employs, both of which are determined by the designer. Users have the freedom to design chaotic systems according to their individual needs and preferences.

3. Performance Analysis of New Systems

In this section, we conduct a quantitative analysis of three chaotic systems generated by ND-CMCM to verify their complexity, ergodicity, and sensitivity to parameters. The evaluation encompasses three aspects: bifurcation and phase diagrams, Lyapunov exponents, and sample entropy.

3.1. Bifurcation Diagram and Phase Diagram

The dynamical behavior of a chaotic system can be revealed by using bifurcation diagrams and phase diagrams, which are visualizations of the trajectory and ergodicity of the chaotic system. The bifurcation diagram shows the distribution of the output of a chaotic system in the phase space over a specific range of parameters. The phase diagram draws the values of two or more state variables against each other, representing the intrinsic relationship of the state variables. Figures 1 and 2 show the bifurcation and phase diagrams of 2D-LPCMM, 3D-CFICMM, and 4D-IISCCMM, respectively. The 3D and 4D systems require multiple figures to show all the bifurcation and phase diagrams because of the high dimensionality and many parameters. Since the contents of these figures are similar, only some of them are shown. From the figures, we can see that the chaotic trajectory spreads over the whole phase space. There is no period window over a large range of parameters. The chaotic output values are not clustered at the edges of the phase space and are extremely uniformly distributed.

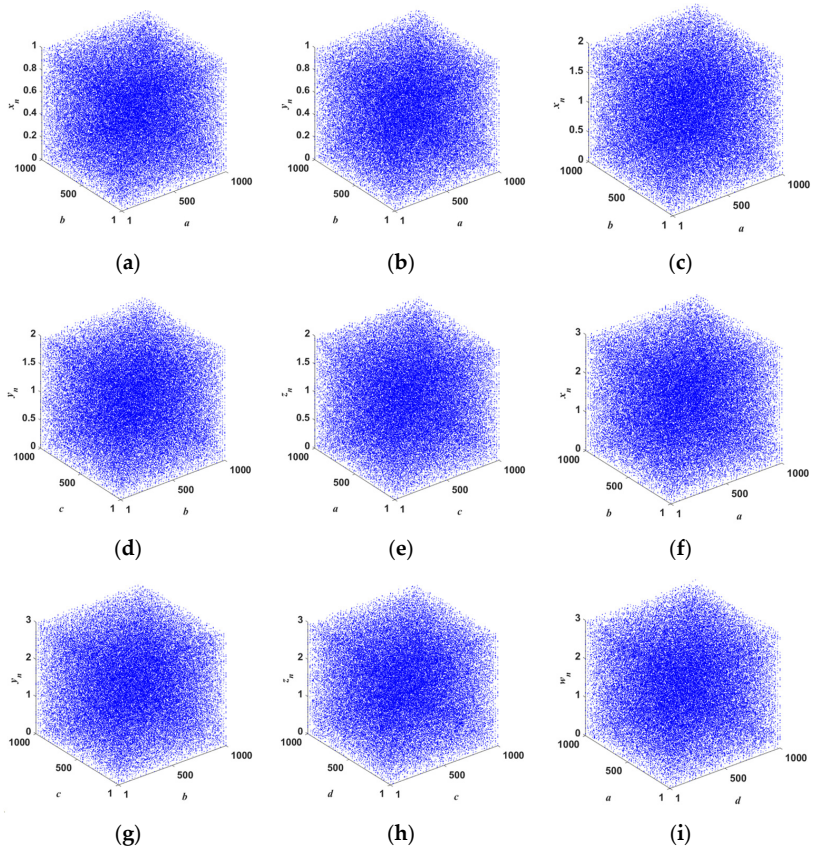

Figure 1. Bifurcation of (**a**,**b**) 2D-LPCMM, (**c**–**e**) 3D-CFICMM, and (**f**–**i**) 4D-IISCCMM.

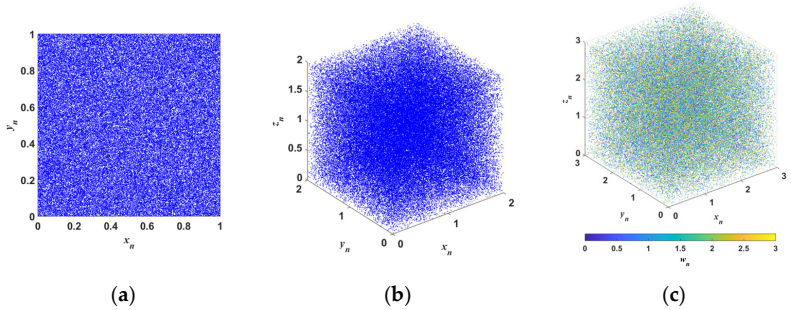

Figure 2. Phase diagrams of (**a**) 2D-LPCMM, (**b**) 3D-CFICMM, and (**c**) 4D-IISCCMM.

3.2. Lyapunov Exponent

The Lyapunov exponent (LE) is an index for the quantitative evaluation of chaotic systems. It reflects the sensitivity of a chaotic system to the initial conditions and enables determination of whether the chaotic system is in a chaotic or periodic state with specific parameters. The number of Lyapunov exponents of a chaotic system is equal to the number of dimensions of its phase space. One positive Lyapunov exponent indicates that the system exhibits chaotic behavior, and more than one positive Lyapunov exponent indicates that the system exhibits hyperchaotic behavior. The spectrum of Lyapunov exponents is shown in Figure 3. All three systems have positive Lyapunov exponents equal to the dimensionality in all parameter ranges. To visualize the superiority of our proposed system, the maximum Lyapunov exponent (MLE) spectrum of the proposed systems and the seed maps are plotted in Figure 4. The MLE spectrums of chaotic systems are scaled to the figure with horizontal coordinates in the range $a \in [1,1000]$. For example, the parameter range of the logistic map is $\mu \in [\mu_{\min},\mu_{\max}] = [0,4]$, and the value of μ as the horizontal coordinate a varies can be obtained by Equation (14). The other parameter of 2D-LPCMM was set to $b = 500$. The other parameters of 3D-CFICMM were set to $b = 500$ and $c = 500$. The other parameters of 4D-IISCCMM were set to $b = 500$, $c = 500$, and $d = 500$. The other parameter of 1DSP was set to $\alpha = 4.4926$.

$$\mu = \frac{a-1}{(1000-1)} \times (\mu_{\max} - \mu_{\min}) + \mu_{\min} \tag{14}$$

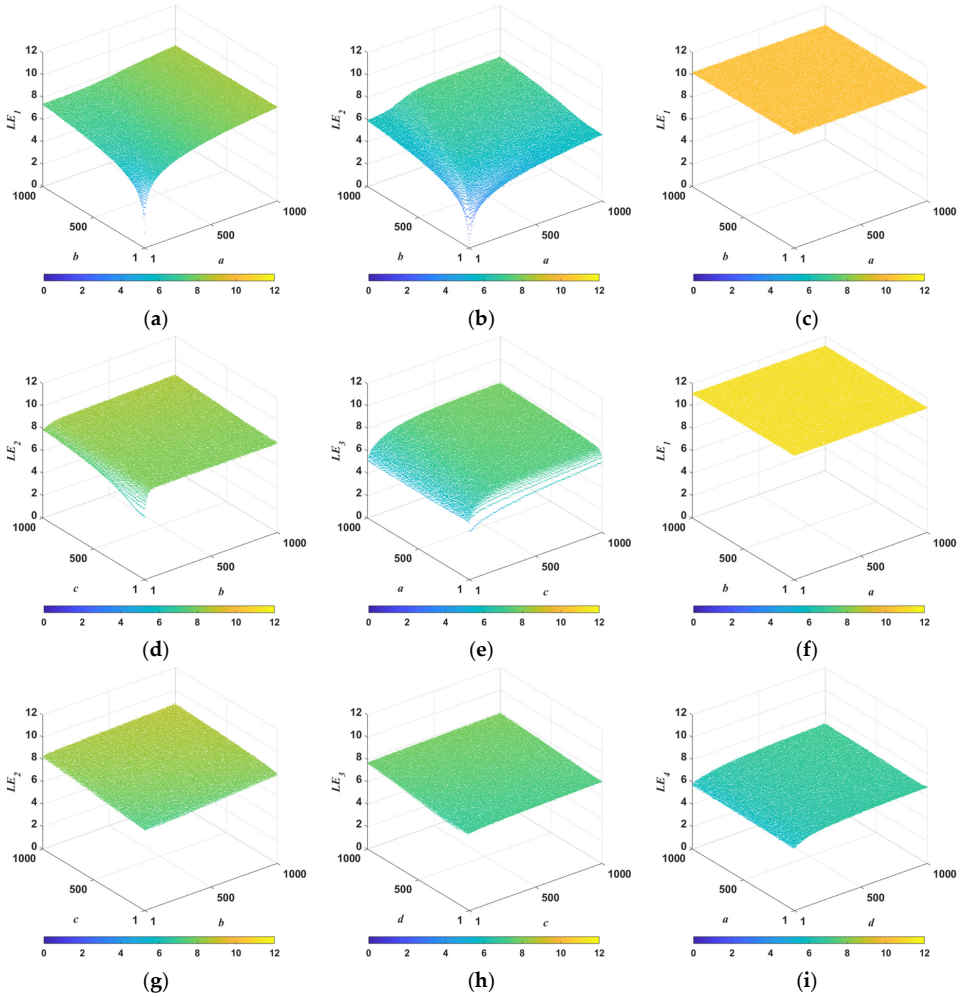

Figure 3. Lyapunov exponent of (**a**,**b**) 2D-LPCMM, (**c**–**e**) 3D-CFICMM, and (**f**–**i**) 4D-IISCCMM.

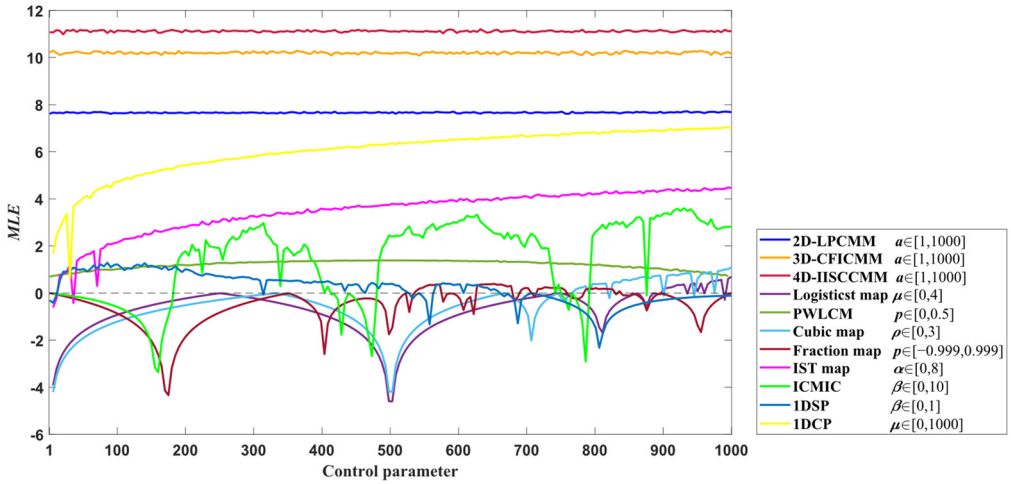

Figure 4. Comparison chart of MLE.

3.3. Sample Entropy

Sample entropy (SE) is a measure that quantifies the level of regularity or complexity observed in a time series, allowing us to assess the predictability and unpredictability of the resulting chaotic output. Consider a time series of the form $\{x_1, x_2, \ldots, x_N\}$. Set the value of the parameter m, which means the length of the template vector. For each template vector $X_m(i)$, defined as $\{x_i, x_{i+1}, \ldots, x_{i+m-1}\}$, we calculate the Chebyshev distance between each pair of template vectors and denote it as $d[X_m(i), X_m(j)]$. Set another parameter r, which means tolerance threshold. The number of vectors with Chebyshev distances below the threshold r is calculated and denoted as B. The number of vectors with Chebyshev distances below the threshold r is calculated for each pair of template vectors after increasing the length of the template by 1, i.e., $m + 1$, and denoted as A. SE can be defined as

$$SE(m, r, N) = -\log \frac{A}{B} \tag{15}$$

A lower SE value indicates a higher regularity in the chaotic sequence, with more certainty and order. On the contrary, higher SE values indicate higher complexity and irregularity. The parameters m and r affect the magnitude of the calculated SE, and in this paper we set them to 2 and 0.2, respectively. Figure 5 plots part of the SE of 2D-LPCMM, 3D-CFICMM, and 4D-IISCCMM, and it can be seen that in all parameter ranges SE is greater than 2. Figure 6 shows a comparison of the SE of the proposed systems and the seed maps. The other parameter settings for the multiparameter chaotic systems are the same as in Figure 4.

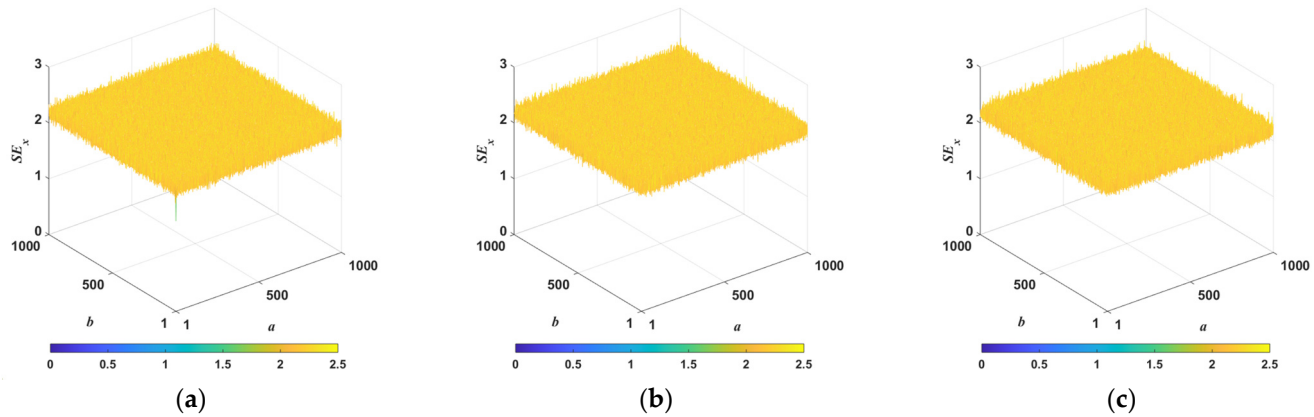

Figure 5. SE of (**a**) 2D-LPCMM, (**b**) 3D-CFICMM, and (**c**) 4D-IISCCMM.

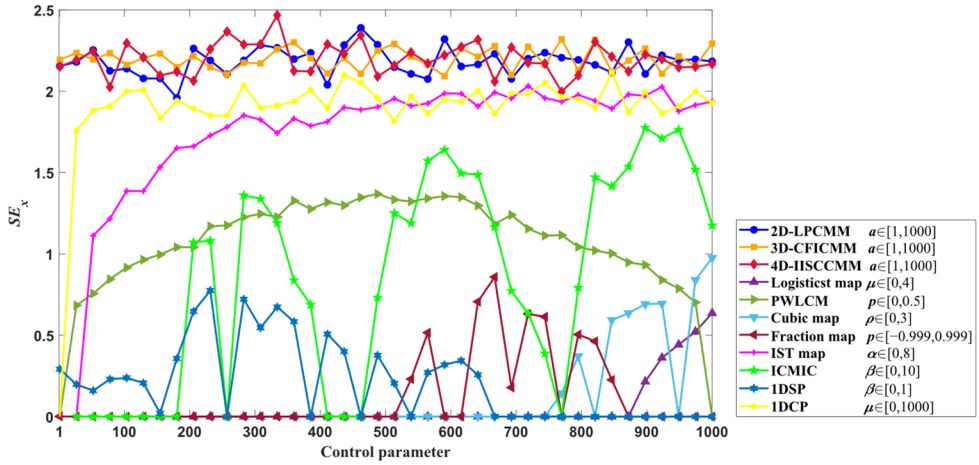

Figure 6. Comparison chart of SE.

4. Proposed Encryption Algorithm

Chaotic systems are generally used as pseudo-random sequence generators in image encryption algorithms. The performance of the chaotic system affects the effectiveness of image encryption. Our proposed chaotic system has excellent sensitivity to parameters, complexity, and unpredictability, while the chaotic trajectory is very uniform and covers the whole phase space. The chaotic system generated by ND-CMCM is well suited for application in image encryption. Based on this, we have developed an algorithm that can be used regardless of the number, type, and size of images. The flowchart of this algorithm is shown in Figure 7.

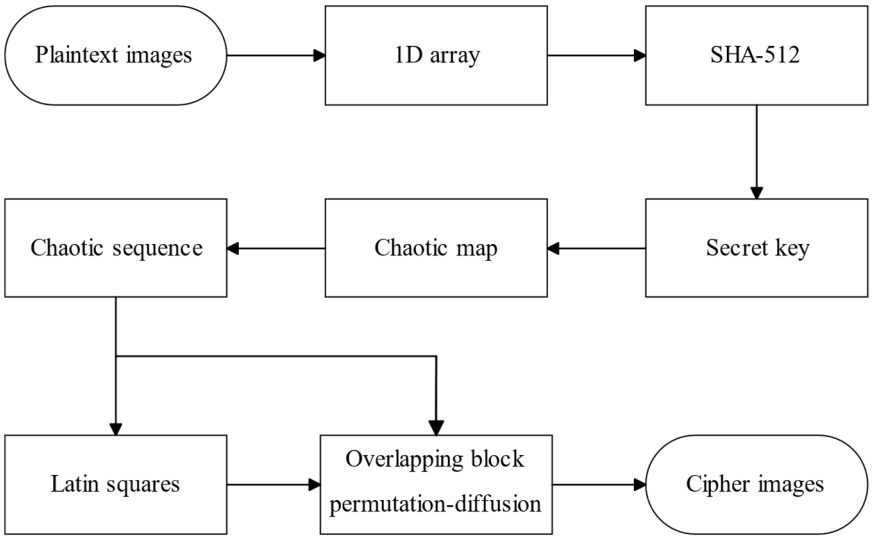

Figure 7. Flowchart of the encryption process.

4.1. Overlapping Block

The overlapping block is designed to increase the complexity of image encryption while designing adjustable parameters to balance the encryption time and encryption effect. The length of the overlapping blocks is random and generated by the chaotic sequence. We designed the parameters $B_{\min}$ and $B_{\max}$, which are the minimum and maximum values of the random length of the overlapping blocks, respectively, and they determine the percentage of overlapping blocks.

The method of overlapping block partition is shown in Figure 8. First, determine the values of $B_{\min}$ and $B_{\max}$. The smaller the $B_{\min}$ and the larger the $B_{\max}$, the longer

the encryption time and the better the encryption effect. Second, calculate the number of overlapping blocks:

$$B_n = \text{ceil}(B_{len} / B_{\min}) \tag{16}$$

where B_{len} is the length of the 1D array being partitioned and ceil is an upward rounding function. Finally, the random length of overlapping blocks $S(i)$ ($i = 1, 2, \ldots, B_n$) is generated by chaotic sequences. The overlapping parts are repeatedly involved in the permutation–diffusion process. For a detailed algorithm on overlapping block permutation–diffusion, please refer to Section 4.3.

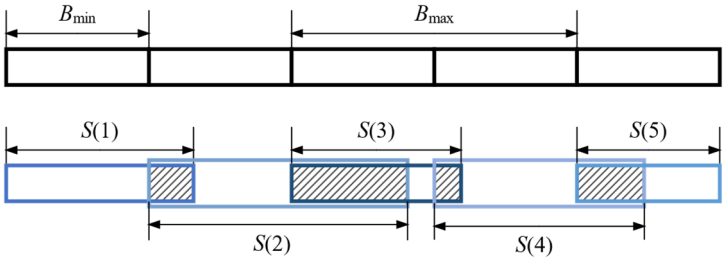

Figure 8. Overlapping block partition diagram.

4.2. Newly Defined Operations Based on Latin Squares and Lookup Table

A Latin square is a mathematical structure that is a square array of $n \times n$. In this $n \times n$ square, there are n different elements, and each different element appears only once in the same row or column. Elements can be numbers, letters, symbols, etc. We use Algorithm 1 [39] to generate a n-th-order Latin squares with elements of positive integers from 0 to n.

A lookup table operation is a way to quickly access values based on data in a table or similar array. We define a new operation using the Latin square as a lookup table. Consider a Latin square L of order $n + 1$ with constituent elements 0, 1, …, n. The lookup table operation function based on the Latin square (LL) is defined as

$$k = \text{LL}(i, j, L) = L[i + 1, j + 1] \tag{17}$$

where LL is the name of the function, $L[i,j]$ represents taking out the elements of the i-th row and j-th column of matrix L. We take an example of a lookup table based on the four-order Latin square L_0 to illustrate the operation specifically, as shown in Figure 9a, where the white squares are the Latin squares L_0, the gray column squares are the i in Equation (17), and the gray row squares are the j. It is easy to know that performing an addition operation on 2 and 3 yields 5 and performing a bitxor operation on 2 and 3 yields 1. However, when we perform the L_0-based LL operation on 2 and 3, i.e., LL(2,3,L_0), the result of the operation is the number in row 3, column 4 of L_0, i.e., $L_0[2 + 1,3 + 1] = L_0[3,4] = 1$. In order to obtain i through k and j, we need to perform an reverse LL operation. Specifically, we obtain the reverse Latin square through Equation (18), and then we perform the LL operation on k and j. The algorithm for generating the reverse Latin square is shown in Algorithm 2, and the reverse Latin square L_0' of L_0 is shown in Figure 9b.

$$i = \text{LL}(k, j, L') = L'[(L[i + 1, j + 1] + 1, j + 1) \tag{18}$$

More examples of LL operations are shown below.

$$\text{LL}(1, 2, L_0) = L_0[2, 3] = 0 \tag{19}$$

$$\text{LL}(0, 2, L_0') = L_0'[1, 3] = 1 \tag{20}$$

$$\text{LL}(3, 1, L_0) = L_0[4, 2] = 2 \tag{21}$$

$$\mathrm{LL}(2,1,L_0{}') = L_0{}'[3,2] = 3 \tag{22}$$

LL is an operation with both input and output in the range of integers 0 to n. It has more complex nonlinear properties than additive operations, XOR operations, and Galois field, and it is suitable to be used in confusion diffusion of image encryption. In order to visually compare the differences between the proposed operation and other different operations, we generated three Latin squares L_1, L_2, and L_3 and drew the results of their operations. Figure 10 shows a discrete plot of the results of the different operations, where x and y are integers in the range 0 to 255 and the blue color represents the points in the 3D plot determined by x, y and the results obtained from x and y. It can be seen that our proposed operation is more complex compared to other operations. In addition, different Latin squares can give different results, so the operations are diverse.

	0	1	2	3
0	1	0	3	2
1	2	1	0	3
2	0	3	2	1
3	3	2	1	0

(**a**)

	0	1	2	3
0	2	0	1	3
1	0	1	3	2
2	1	3	2	0
3	3	2	0	1

(**b**)

Figure 9. Lookup table based on the four-order Latin square L_0 (**a**) and reverse Latin square $L_0{}'$ (**b**).

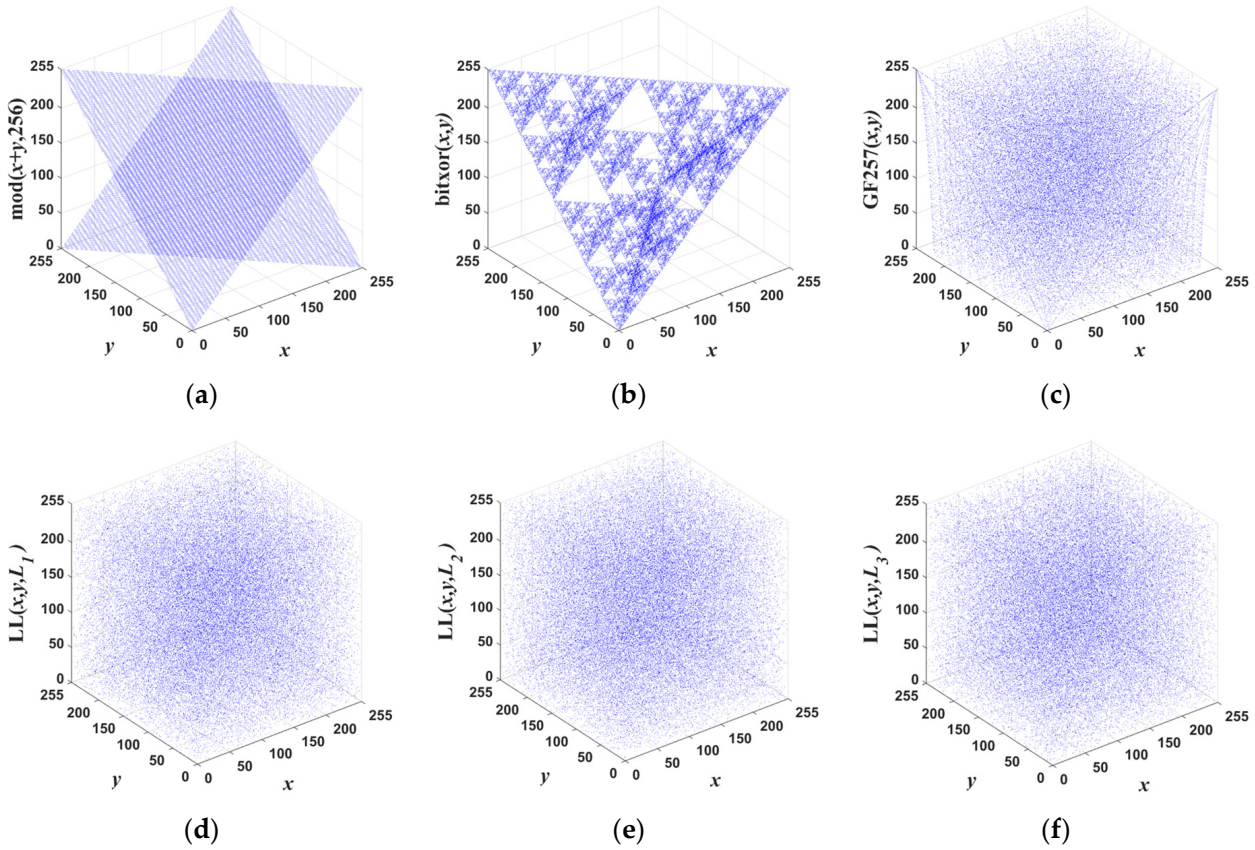

Figure 10. Results of different operations in the range 0 to 255: (**a**) modular sum, (**b**) bitxor, (**c**) GF (257), (**d**) LL operation based on L_1, (**e**) LL operation based on L_2, (**f**) LL operation based on L_3.

Algorithm 1 Pseudo-code for generating Latin squares

Input: chaotic sequences $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ of length n
Output: nth order Latin square $\boldsymbol{L}$
[~,$\boldsymbol{I}_1$] = sort($\boldsymbol{Q}_1$);
[~,$\boldsymbol{I}_2$] = sort($\boldsymbol{Q}_2$);
$\boldsymbol{I}_1 = \boldsymbol{I}_1 - 1$;
for $i = 0$ to $n - 1$
    $\boldsymbol{L}(i + 1,:)$ = circshift($\boldsymbol{I}_1$,$\boldsymbol{I}_2(i + 1)$);
end for

Algorithm 2 Pseudo-code for generating reverse Latin squares

Input: nth Latin square $\boldsymbol{L}$
Output:nth reverse Latin square $\boldsymbol{L}'$
for $i = 0$ to $n - 1$
    for $j = 0$ to $n - 1$
        $\boldsymbol{L}'(\boldsymbol{L}(i + 1,j + 1) + 1,j + 1) = i$;
    end for
end for

4.3. Encryption Process

The proposed encryption algorithm provides parameters that can balance the encryption time and efficiency. The parameters we provide are $B_{\min}$, $B_{\max}$, and L_{num}; the first two are used to determine the length of the random overlapping block and the latter determines the number of Latin squares used for the LL operation.

4.3.1. Generation of Secret Key and Chaotic Sequences

The proposed encryption algorithm offers two methods for setting the key, which is the parameter of the chaotic system. The first method involves dividing it into an internal key and an external key. The internal key is generated using the hash function, while the external key is generated by random noise. By combining the sensitivity of the hash function to the plaintext and the sensitivity of the chaotic system to the parameters, even a small change in the plaintext image will lead to a completely different encryption result. The external key is derived from random noise, increasing security and making the encryption result different each time. The second method allows users to set their own keys. The diffusion phase in the proposed encryption algorithm can propagate small changes in the plaintext image to the whole image, and in combination with the newly defined nonlinear operations, the encryption becomes unpredictable. In this paper, we use 2D-LPCMM for encryption.

The steps to generate the chaotic sequences are as follows.

Step 1: Input n images and record their sizes.

Step 2: Expand the input images into a 1D array in column-major order and concatenate them horizontally, denoted as P. Calculate the length of P, denoted as B_{len}. Calculate B_n according to Equation (16).

Step 3: In this step, the key is generated by the hash function. It can be skipped if the key is customized. The hash function we use is SHA-512. Input P into the hash function to obtain a binary number H of length 512. Convert H to decimal and divide it into 32 parts named $k_1, k_2, k_3, \ldots, k_{32}$. H and k_i are calculated as follows.

$$H = \text{hash}\left(P,'\,\text{SHA} - 512'\right) \tag{23}$$

$$k_i = \text{bin2dec}(H(16(i-1) + 1 : 16i), \quad i = 1, 2, \cdots 32 \tag{24}$$

Obtain the control and initial parameters of the chaotic system from the following equation.

$$\begin{cases} a = \text{sum}(k_1, k_2, k_3, k_4) \times \text{sum}(k_5, k_6, k_7, k_8)/2^{36} \times 1000{+}5 \\ b = \text{sum}(k_9, k_{10}, k_{11}, k_{12}) \times \text{sum}(k_{13}, k_{14}, k_{15}, k_{16})/2^{36} \times 1000{+}5 \\ x_0 = \text{mod}(\text{sum}(k_{17}, k_{18}, k_{19}, k_{20})/\text{sum}(k_{21}, k_{22}, k_{23}, k_{24}) + \text{rand}(1), 1) \\ y_0 = \text{mod}(\text{sum}(k_{25}, k_{26}, k_{27}, k_{28})/\text{sum}(k_{29}, k_{30}, k_{31}, k_{32}) + \text{rand}(1), 1) \\ t = \text{mod}(\text{ceil}(mean(k_1, k_2, \ldots, k_{32})), 1000) + 1000 \end{cases} \quad (25)$$

where a and b are the control parameters, x_0 and y_0 are the initial parameters of the chaotic system, t denotes the first t values of the chaotic sequence discarded to prevent transient effects, and rand is a function that generates uniform random noise.

Step 4: Substitute a, b, x_0, y_0 into 2D-LPCMM for iteration, and after discarding the first t values, chaotic sequence CS_0 of length $512 \times L_{num}$ and chaotic sequences CS_1, CS_2, CS_3, CS_4, CS_5, CS_6, CS_7, CS_8 of length B_n are generated. Each overlapping block length S_1 and S_2 can be calculated by the following equation.

$$S_1 = \text{mod}\left(\text{ceil}\left(CS_1 \times 10^{10}\right), B_{\max} - B_{\min}\right) + B_{\min} \quad (26)$$

$$S_2 = \text{mod}\left(\text{ceil}\left(CS_2 \times 10^{10}\right), B_{\max} - B_{\min}\right) + B_{\min} \quad (27)$$

$$B_{S1} = \text{sum}(S_1) \quad (28)$$

$$B_{S2} = \text{sum}(S_2) \quad (29)$$

where B_{s1} and B_{s2} are the sum of the lengths of all overlapping blocks. Then, generate the chaotic sequences CS_9 and CS_{10} of length B_{S1}, CS_{11} of length B_{S2}.

4.3.2. Overlapping Block Permutation–Diffusion

Permutation and diffusion are simultaneous and are divided into forward permutation–diffusion and backward diffusion. The permutation can scramble the values within a block into the entire 1D array. The diffusion includes inter-block diffusion, intra-block diffusion, and mask diffusion.

To facilitate understanding, we first present an example before summarizing the encryption process, as shown in Figure 11. In this example, the parameter $B_{\min}$ is set to 4, $B_{\max}$ is set to 8, and L_{num} is set to 2.

Step 1: Generate L_{num} (2) Latin squares denoted $L\{1\}$ and $L\{2\}$, as shown in Figure 12.

Step 2: Input two images of size 3×3 and expand them column-wise into a 1D array P of length 18.

Step 3: Divide P into 5 (ceil(18/4), see Equation (16)) overlapping blocks. The initial position of each overlapping block in the figure is indicated by the black dashed line. The length of each block is between $B_{\min}$ (4) and $B_{\max}$ (8) and is determined by the chaotic sequence.

Step 4: Choose the first overlapping block, framed by the black line, whose length is 7. Record the index of the first overlapping block as A, A = [1, 2, 3, 4, 5, 6, 7].

Step 5: Generate 7 random numbers E ranging from 1 to 18, E = [17, 13, 18, 15, 10, 3, 13].

Step 6: Process arrays A and E. As there are many duplicate numbers in A and E, this may cause data loss. Combine E and A into a temporary array *Temp* = [17, 13, 18, 15, 10, 3, 13, 1, 2, 3, 4, 5, 6, 7], remove the duplicates, and split it into two arrays E_1 = [17, 13, 18, 15, 10, 3] and A_1 = [1, 2, 4, 5, 6, 7]. The process is represented by Algorithm 3.

Step 7: Swap the pixel of P indexed by E_1 and A_1, shown in the figure with black bi-directional arrows. For example, the 17th pixel (7) and the first pixel (0) swap positions.

Step 8: Perform the $L\{2\}$-based LL operation on the first two pixels of the first overlapping block and the first two pixels of the last overlapping block. This operation is indicated by orange arrows in the figure. For example, the first pixel of the first block is 7, and the

first pixel of the last block is 0. Operate on them to obtain LL(7,0,L{2}) = 7. Similarly, the second pixel of these two blocks is operated on to obtain LL(3,3,L{2}) = 1.

Step 9: Perform the L{1}-based LL operation on each pixel of the first overlapping block and its previous pixel. This operation is indicated by purple arrows in the figure. For example, the first pixel of the first block is 7, and the last pixel of the last block is 3. Operating on them obtains LL(7,3,L{1}) = 1. Similarly, the second pixel of the first block is 1, and now the first pixel of the first block is 1. Operating on them obtains LL(1,1,L{1}) = 4.

Step 10: Generate 7 random numbers D ranging from 0 to 8, D = [5, 8, 8, 1, 3, 6, 2].

Step 11: Perform the L{1}-based LL operation on each pixel of the first overlapping block and numbers in array D. This operation is indicated by gray arrows in the figure. For example, the first pixel of the first overlapping block is 1 and the first number in D is 5. Operating on them obtains LL(1,5,L{1}) = 5. The second pixel of the first overlapping block is 4 and the second number in D is 8. Operating on them obtains LL(4,8,L{1}) = 6.

Step 12: The first $B_{\min}$ (4) blocks of the first overlapping block are no longer involved in the subsequent permutations. This means that the index of the 1D array P starts at the first pixel of the second overlapping block.

Step 13: Choose the second overlapping block, framed by the black line, whose length is 8. Record the index of the first overlapping block as A, A = [1, 2, 3, 4, 5, 6, 7, 8].

Step 14: Repeat step 5 to step 11. The generated random arrays E and D are E = [7, 13, 2, 3, 6, 8, 6, 7], D = [8, 1, 6, 5, 0, 5, 5, 4]. The Latin squares used in steps 8, 9, and 11 are L{1}, L{2}, and L{2}, respectively.

Step 15: Recover the encrypted 1D array P into two images of size 3 × 3.

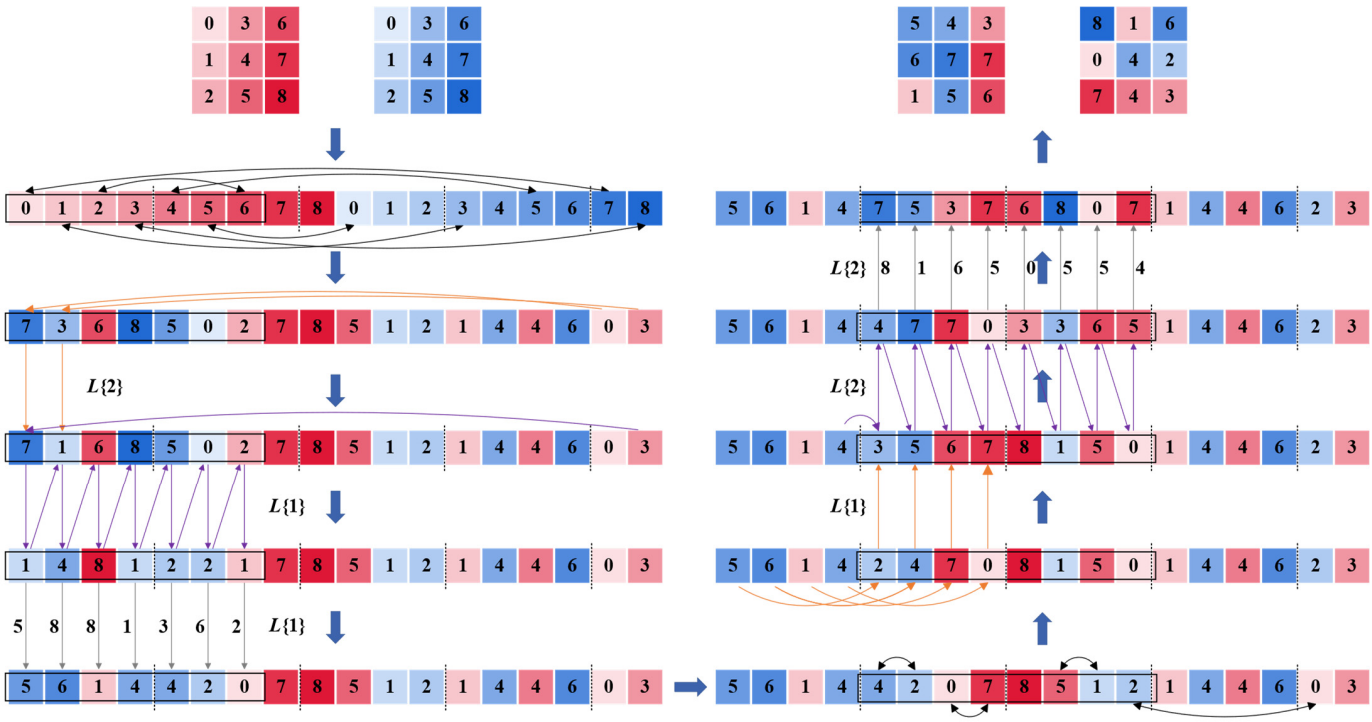

Figure 11. An example of forward permutation–diffusion.

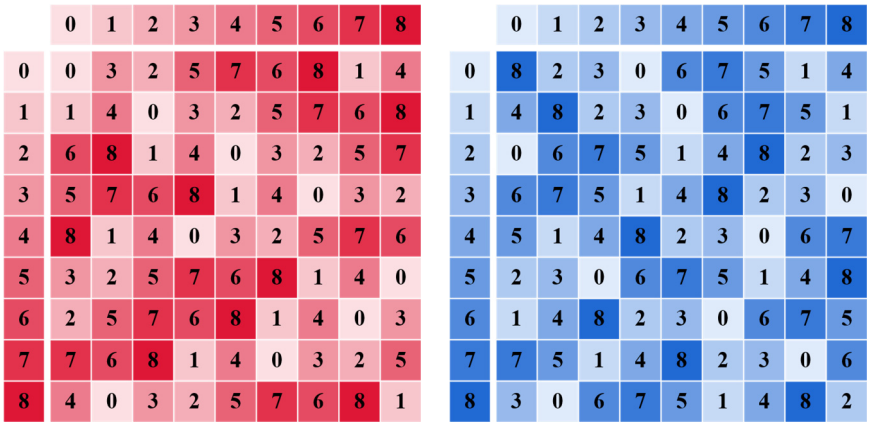

Figure 12. Two examples of LL operations: (**a**) $L\{1\}$, (**b**) $L\{2\}$.

Algorithm 3 Pseudo-code for index processing

Input: E, A
Output: E_1, A_1
1: $Temp$ = [E A];
2: $Temp$ = unique($Temp$,'stable');
3: S_{temp} = length($Temp$);
4: **if** mod(S_{temp},2) = 1
5: $Temp$(end) = [];
6: S_{temp} = S_{temp} − 1;
7: **end if**
8: E_1 = $Temp$(1: S_{temp}/2);
9: A_1 =$Temp$(S_{temp}/2 + 1:end);

The detailed encryption process is summarized as follows.

Step 1: Confirm the parameters B_{min}, B_{max}, and L_{num}. B_{min} and B_{max} determine the length range of the overlapping blocks, and L_{num} determines the number of Latin squares to be generated.

Step 2: Generate L_{num} Latin squares. Divide CS_0 into L_{num} groups, each group has 2 arrays of length 256, and generate L_{num} Latin squares $L\{i\}$ (i = 1, 2, …, L_{num}) by Algorithm 1.

Step 3: Obtain the random number used to select Latin squares L_1, L_2, and L_3 through the chaotic sequences CS_3, CS_4, CS_5.

Step 4: Input n images and record their sizes. Expand the input images into a 1D array in column-major order and concatenate them horizontally, denoted as P. Calculate the length of P, denoted as B_{len}.

Step 5: Divide P into B_n overlapping blocks, where B_n = ceil(B_{len}/B_{min}).

Step 6: Select the i-th (i = 1, 2, …, B_n) overlapping block. Denote the length of the block as S, where $S = S_1(i)$ and S_1 is obtained from Equation (26).

Step 7: Denote the length of the array in which the overlapping blocks are to be arranged as P_{len}, where $P_{len} = B_{len} - (i-1) \times B_{min}$. When P_{len} is less than $S_1(i)$, $S = P_{len}$.

Step 8: Generate S random numbers E ranging from 1 to P_{len} through the chaotic sequences CS_9, where E = mod(ceil($CS_9 \times 10^{14}$),P_{len}) + 1.

Step 9: Permutation. Let A be the array from 1 to $S_1(i)$. Obtain E_1 and A_1 by inputting E and A into Algorithm 3. Swap the pixels of P indexed by E_1 and A_1.

Step 10: Inter-block diffusion. Perform the LL operation based on the L_1-th Latin square $L\{L_1\}$ for the first B_{min} pixels of the i-th overlapping block and the first B_{min} pixels of the $i-1$-th overlapping block.

Step 11: Intra-block diffusion. Perform the LL operation based on the L_2-th Latin square $L\{L_2\}$ for the k-th (k = 1, 2,. . ., S) pixel and the $k-1$-th pixel of the i-th overlapping block.

Step 12: Generate S random numbers D ranging from 0 to 255 through the chaotic sequences CS_{10}, where D = mod(ceil($CS_{\mathbf{10}} \times 10^{14}$),256).

Step 13: Mask diffusion. Perform LL operations based on the L_3-th Latin square $L\{L_3\}$ for the whole i-th overlapping block and D.

Step 14: Forward permutation–diffusion. Add 1 to i ($i = i + 1$) and loop step 4 to step 11 until $i = B_{\mathrm{n}}$. The specific pseudo-code is shown in Algorithm 4.

Step 15: Flip the array P from left to right P = fliplr(P).

Step 16: Select the i-th (i = 1, 2, . . ., B_{n}) overlapping block. Denote the length of the block as S, where $S = S_2(i)$ and S_2 is obtained from Equation (27).

Step 17: Backward diffusion. Add 1 to i ($i = i + 1$) and loop step 8 to step 11 until $i = B_{\mathrm{n}}$.

Step 18: Recover the encrypted 1D array P into multiple images based on the sizes of the recorded original images.

Algorithm 4 Pseudo-code for forward permutation–diffusion

Input: $\boldsymbol{P}$, B_{len}, B_{min}, $\boldsymbol{S_1}$, $\boldsymbol{CS_3}$, $\boldsymbol{CS_4}$, $\boldsymbol{CS_5}$, $\boldsymbol{CS_9}$, $\boldsymbol{CS_{10}}$, $\boldsymbol{CS_{13}}$,
Output: $\boldsymbol{P}$

$Ct = 1$; $P_{\mathrm{len}} = B_{\mathrm{len}}$;
for i = 1 to B_{n}
  if $P_{\mathrm{len}} < \boldsymbol{S_1}(i)$
    $S = P_{\mathrm{len}}$;
  end if
  L_1 = mod(ceil($\boldsymbol{CS_3}(i) \times 10^{14}$),$L_{\mathrm{num}}$) + 1; L_2 = mod(ceil($\boldsymbol{CS_4}(i) \times 10^{14}$),$L_{\mathrm{num}}$) + 1;
  L_3 = mod(ceil($\boldsymbol{CS_5}(i) \times 10^{14}$),$L_{\mathrm{num}}$) + 1; $\boldsymbol{E}$ = mod(ceil($\boldsymbol{CS_9}(Ct{:}Ct + S - 1) \times 10^{14}$),$P_{\mathrm{len}}$) + 1;
  $\boldsymbol{D}$ = mod(ceil($\boldsymbol{CS_{10}}(Ct{:}Ct + S - 1) \times 10^{14}$),256); $Ct = Ct + S$;
  $\boldsymbol{A}$ = 1 to S;
  [$\boldsymbol{E_1}$,$\boldsymbol{A_1}$] = **Algorithm 2**($\boldsymbol{E}$,$\boldsymbol{A}$); $\boldsymbol{E_1} = \boldsymbol{E_1} + (i-1) \times B_{\mathrm{min}}$; $\boldsymbol{A_1} = \boldsymbol{A_1} + (i-1) \times B_{\mathrm{min}}$;
  $\boldsymbol{P}(\boldsymbol{E_1}) \Leftrightarrow \boldsymbol{P}(\boldsymbol{A_1})$
  if $i > 1$
    if $P_{\mathrm{len}} > B_{\mathrm{min}}$
      $\boldsymbol{P}((i-1) \times B_{\mathrm{min}} + 1{:}i \times B_{\mathrm{min}}) = \mathrm{LL}(\boldsymbol{P}((i-1) \times B_{\mathrm{min}} + 1{:}\mathrm{i} \times B_{\mathrm{min}}), \boldsymbol{P}((i-2) \times B_{\mathrm{min}} + 1{:}(i-1) \times B_{\mathrm{min}}), \boldsymbol{L}\{L_1\})$;
    else
      $\boldsymbol{P}((i-1) \times B_{\mathrm{min}} + 1{:}\mathrm{end}) = \mathrm{LL}(\boldsymbol{P}((i-1) \times B_{\mathrm{min}} + 1{:}\mathrm{end}), \boldsymbol{P}((i-2) \times B_{\mathrm{min}} + 1{:}(i-1) \times B_{\mathrm{min}}), \boldsymbol{L}\{L_1\})$;
    end if
    for k = 1 to S
      $\boldsymbol{P}((i-1) \times B_{\mathrm{min}} + k) = \mathrm{LL}(\boldsymbol{P}((i-1) \times B_{\mathrm{min}} + k), \boldsymbol{P}((i-1) \times B_{\mathrm{min}} + k - 1), \boldsymbol{L}\{L_2\})$;
    end for
  else
    $\boldsymbol{P}(1{:}B_{\mathrm{len}} - ((B_{\mathrm{n}} - 1) \times B_{\mathrm{min}})) = \mathrm{LL}(\boldsymbol{P}(1{:}B_{\mathrm{len}} - ((B_{\mathrm{n}} - 1) \times B_{\mathrm{min}})), \boldsymbol{P}((B_{\mathrm{n}} - 1) \times B_{\mathrm{min}} + 1{:}\mathrm{end}), \boldsymbol{L}\{L_1\})$;
    $\boldsymbol{P}(1) = \mathrm{LL}(\mathrm{P}(1), \boldsymbol{P}(B_{\mathrm{n}}), \boldsymbol{L}\{L_2\})$;
    for k = 2 to S
      $\boldsymbol{P}(k) = \mathrm{LL}(\boldsymbol{P}(k), \boldsymbol{P}(k-1), \boldsymbol{L}\{L_2\})$;
    end for
  end if
  $\boldsymbol{P}((i-1) \times B_{\mathrm{min}} + 1{:}(i-1) \times B_{\mathrm{min}} + S) = \mathrm{LL}(\mathrm{P}((i-1) \times B_{\mathrm{min}} + 1{:}(i-1) \times B_{\mathrm{min}} + S), \boldsymbol{D}, \boldsymbol{L}\{L_3\})$;
  $P_{\mathrm{len}} = P_{\mathrm{len}} - B_{\mathrm{min}}$;
end for

4.4. Decryption Process

The decryption process is the inversion of the encryption. The specific decryption process is as follows.

Step 1: Record the size of the ciphertext images and expand them into a 1D array in column-major order, concatenate them horizontally to obtain a 1D array P. Calculate the length of P, denoted as B_{len}. Calculate B_n according to Equation (16).

Step 2: Generate chaotic sequences from keys.

Step 3: Generate the Latin squares used in the encryption process and input them into Algorithm 2 to obtain the reverse Latin squares.

Step 4: Perform the reverse operation of overlapping block permutation–diffusion to obtain the decrypted 1D array P, where the Latin square used is changed to the reverse Latin square.

Step 5: Recover the decrypted 1D array P into multiple images based on the sizes of the recorded original images.

5. Experimental Results and Security Analysis

In this section, to test the performance of the image encryption algorithm, we encrypt multiple color and grayscale images of different sizes at the same time. These tests analyze the effectiveness of the proposed algorithm from various aspects. The test images were obtained from the USC-SIPI Image Database (https://sipi.usc.edu/database/, accessed on 13 July 2023). The test images we used include $256 \times 256 \times 3$ 4.1.01.tiff, $512 \times 512 \times 1$ 5.2.10.tiff, $512 \times 512 \times 3$ 4.2.05.tiff, and $1024 \times 1024 \times 1$ 5.3.01.tiff. In Section 4.1, we set parameters that can balance encryption time and encryption effect. We used three different groups of parameters to set up three different levels of encryption. The first group is $B_{\min} = \text{ceil}(\text{power}(B_{\text{len}},1/2)) \times 3$, $B_{\max} = 1.5 \times B_{\min}$, $L_{\text{num}} = 20$. The second group is $B_{\min} = \text{ceil}(\text{power}(B_{\text{len}}/3,1/2))$, $B_{\max} = 2 \times B_{\min}$, $L_{\text{num}} = 50$. The third group is $B_{\min} = \text{ceil}(\text{power}(B_{\text{len}},1/3))$, $B_{\max} = 4 \times B_{\min}$, $L_{\text{num}} = 100$. To accommodate different numbers of pixels, $B_{\min}$ and $B_{\max}$ are related to the total number of image pixels B_{len}. Power is a function used to raise a number or array element-wise to a specified power. These three groups of parameters are labeled as Level 1, Level 2, and Level 3, with the speeds ranging from slow to fast and the effects ranging from low to high. If not specified, the second group of parameters is used by default in the following tests.

The encrypted and decrypted images are shown in Figure 13. To facilitate the layout, we have resized the images to the same size, but the actual amount of pixels encrypted remains the same. It can be seen that the images are encrypted into noisy images and their original contents cannot be recognized.

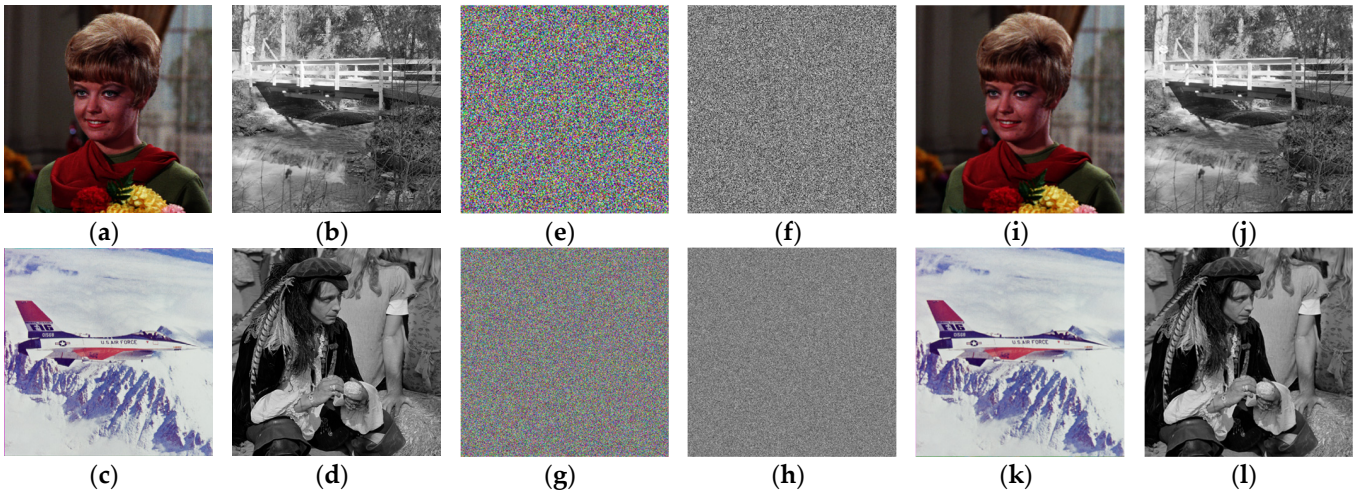

Figure 13. Encryption and decryption images: (**a**–**d**) plaintext images, (**e**–**h**) ciphertext images, and (**i**–**l**) decrypted images.

5.1. Key Space Analysis

The key space refers to the total number of possible keys. In this paper, the keys are set as parameters of the chaotic system, including control parameters a, b and initial

parameters x_0, y_0 and iteration parameters t. In a platform with a computational accuracy of 10^{-14}, the key space is $10^{14\times4} > 2^{186}$. A key space of at least 2^{100} is required to resist brute force attacks [40]. Since ND-CMCM can generate multidimensional chaotic systems, it is only necessary to generate chaotic systems of two dimensions or more to resist brute force attacks.

5.2. Key Sensitivity Analysis

A key-sensitive algorithm means that even a very small change in the key can produce completely different results. The proposed chaotic system has good sensitivity to the initial conditions, and very small changes in the parameters generate completely different chaotic sequences, thus making the decryption results different. In the case of using SHA-512 to generate the key, a = 128.2116103055887, b = 103.4568560525077, x_0 = 0.803690307174427, y_0 = 0.750177111375968, t = 1973, after modifying the number at the end of a, a = 128.2116103055888, and the decrypted image obtained is shown in Figure 14. In the case of the custom key, we set a = 123.45, b = 567.89, x_0 = 0.12345, y_0 = 0.56789, t = 1000 and the decrypted image shown in Figure 14 is after adding 10^{-14} to x_0. It can be seen that the decrypted images are all unrecognizable, which shows that the proposed encryption algorithm has good key sensitivity.

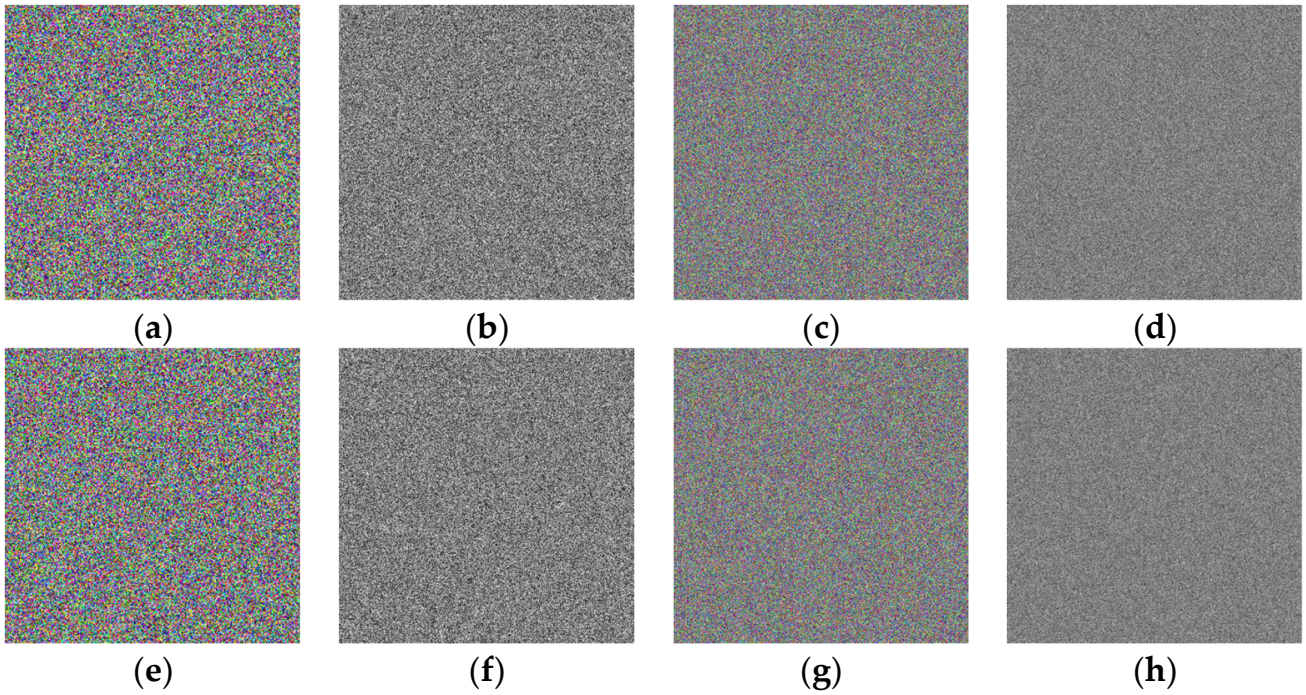

Figure 14. Key analysis: (**a**–**d**) decryption with a modified, (**e**–**h**) decryption with x_0 modified.

5.3. Histogram Analysis

The histogram of the image shows the distribution of all pixel values and visualizes the frequency of pixel occurrences. The histogram of the test image counts the number of pixels from 0 to 255, as shown in Figure 15. The color image has three channels, while the grayscale image has only one. The histograms of the ciphertext images are close to horizontal, indicating that the pixels appear almost equally often and the attacker cannot decrypt them from the statistical analysis.

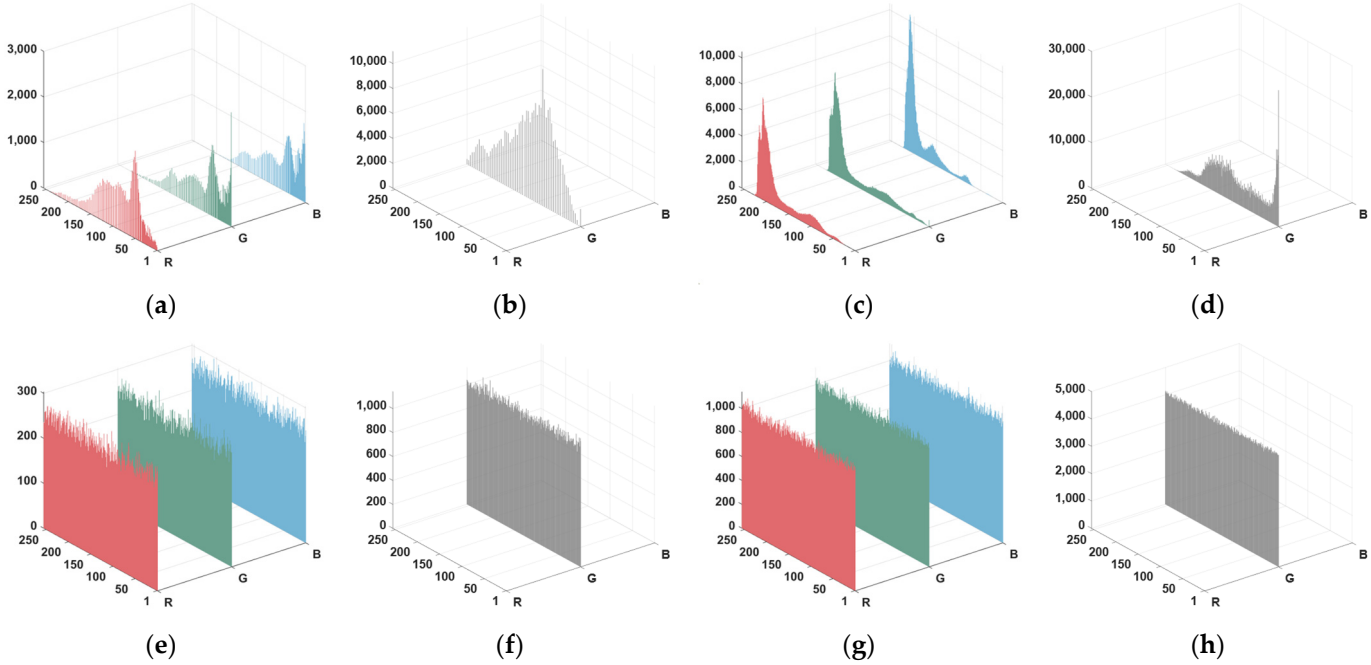

Figure 15. Histogram of (**a**–**d**) plaintext images and (**e**–**h**) ciphertext images.

5.4. Chosen Plaintext Attack Analysis

An attacker can break the encryption by choosing to encrypt a specific plaintext to obtain a plaintext–ciphertext pair. We encrypted a $512 \times 512 \times 3$ pure black image and a pure white image, and the test results are represented in Figure 16. It can be seen that the special images are still unrecognizable as noisy images after encryption. The test data of pure black and pure white are also added in the subsequent tests.

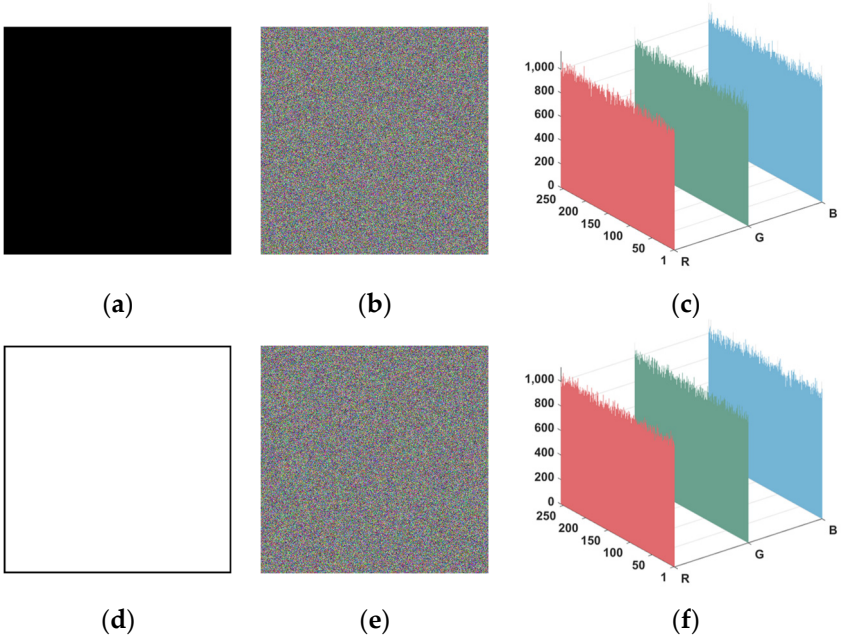

Figure 16. Chosen plaintext attack analysis: plaintext images, ciphertext images, and histograms of (**a**–**c**) pure black and (**d**–**f**) pure white images.

5.5. Differential Attack Analysis

Differential attack is a method that involves continuously modifying plaintext images, encrypting them, and comparing the resulting ciphertext images to obtain key information.

The number of pixels change rate (NPCR) and the uniform average change intensity (UACI) are commonly used to test the resistance of image encryption algorithms to differential attacks. Considering encrypting a plaintext image of size $M \times N$ to obtain a ciphertext image C_1 and modifying the plaintext image by a one-pixel value to obtain a ciphertext image C_2, the NPCR and UACI are calculated as follows.

$$NPCR = \frac{\sum_{ij} D(i,j)}{M \times N} \times 100\% \tag{30}$$

$$UACI = \frac{1}{M \times N}\left[\sum_{ij} \frac{|C_1(i,j) - C_2(i,j)|}{255}\right] \times 100\% \tag{31}$$

$$D(i,j) = \left\{ \begin{array}{ll} 1 & C_1(i,j) \neq C_2(i,j) \\ 0 & C_1(i,j) = C_2(i,j) \end{array} \right\} \tag{32}$$

When the NPCR is close to 99.6094070% and the UACI is close to 33.4635070%, the image encryption algorithm can be considered to have a strong resistance to differential attacks [41,42]. The NPCR and UACI for encrypting different images are listed in Table 1. This table indicates that the secret key generated by SHA-512 maintains good resistance to differential attacks at any size. However, the customized key exhibits performance degradation at small image sizes, possibly due to the limited number of diffusions caused by the small image sizes.

Table 1. NPCR and UACI for encrypting different images.

Key Type	Image	Size	Level 1		Level 2		Level 3	
			NPCR (%)	UACI (%)	NPCR (%)	UACI (%)	NPCR (%)	UACI (%)
Key generated by SHA-512	5.1.11	256 × 256 × 1	99.6184	33.4516	99.6120	33.4797	99.6445	33.5498
	1.1.01	512 × 512 × 1	99.6088	33.4600	99.6132	33.4604	99.6235	33.5580
	1.3.01	1024 × 1024 × 1	99.6108	33.4651	99.6078	33.4424	99.6058	33.4144
	4.1.01	256 × 256 × 3	99.6049	33.4838	99.6119	33.4793	99.5900	33.4514
	4.2.01	512 × 512 × 3	99.6146	33.4744	99.6131	33.4682	99.6260	33.4593
	2.2.01	1024 × 1024 × 3	99.6087	33.4596	99.6102	33.4637	99.6088	33.4732
	Pure black	512 × 512 × 3	99.6084	33.4465	99.6111	33.4456	99.6241	33.4705
	Mean	-	99.6107	33.4630	99.6113	33.4627	99.6175	33.4824
Customized key	5.1.12	256 × 256 × 1	99.5911	33.2539	99.6030	33.5674	99.6124	33.5947
	5.2.10	512 × 512 × 1	99.5975	33.3703	99.6088	33.4564	99.6029	33.4332
	5.3.01	1024 × 1024 × 1	99.6031	33.4669	99.6107	33.4592	99.6011	33.4326
	4.1.05	256 × 256 × 3	99.5967	33.4399	99.6014	33.4807	99.6282	33.4856
	4.2.05	512 × 512 × 3	99.6015	33.4614	99.6084	33.4686	99.6160	33.4847
	6.2.02	1024 × 1024 × 3	99.6111	33.4849	99.6099	33.4630	99.6044	33.4669
	Pure white	512 × 512 × 3	99.6168	33.4971	99.6032	33.4590	99.6170	33.4838
	Mean	-	99.6025	33.4249	99.6065	33.4792	99.6117	33.4831

5.6. The Correlation between Adjacent Pixels Analysis

The correlation of adjacent pixels of an image can be measured by the correlation coefficient. The correlation coefficient is calculated as:

$$Cor_{xy} = \frac{\sum_{i=1}^{N}\left[x_i - \frac{1}{N}\sum_{i=1}^{N} x_i\right]\left[y_i - \frac{1}{N}\sum_{i=1}^{N} x_i\right]}{\sqrt{\sum_{i=1}^{N}\left[x_i - \frac{1}{N}\sum_{i=1}^{N} x_i\right]^2} \times \sqrt{\sum_{i=1}^{N}\left[y_i - \frac{1}{N}\sum_{i=1}^{N} x_i\right]^2}} \tag{33}$$

where x_i and y_i are two sequences of pixels of length N and y_i is an adjacent pixel of x_i. There is the presence of a strong correlation among adjacent pixels within a plaintext image.

A smaller correlation coefficient of neighboring pixels of the ciphertext image indicates a better performance of the algorithm. Good image encryption algorithms aim to reduce the adjacent pixel correlation coefficient to close to 0. We randomly selected 10,000 pairs of adjacent pixels in the horizontal, vertical, and diagonal directions of image "4.2.05", respectively, and plotted the correlation graph in Figure 17. In addition, Table 2 shows the specific values of the correlation coefficients for the plaintext and ciphertext images at different levels of multiple images.

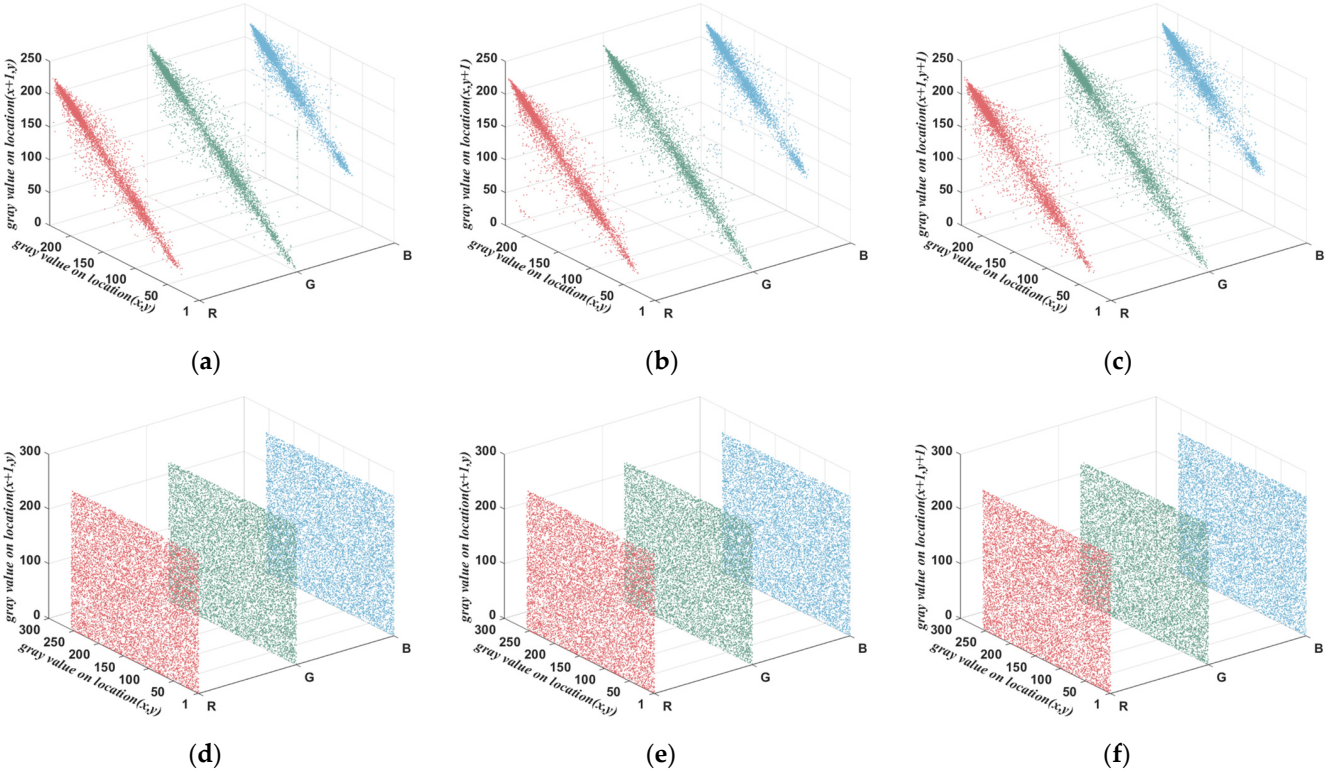

Figure 17. Correlation of horizontal, vertical, and diagonal directions of image "4.2.05": (**a**–**c**) plaintext image, (**d**–**f**) ciphertext image.

Table 2. Correlation coefficients for the plaintext images and ciphertext images.

Image	Size	Direction	Plaintext Image	Ciphertext Image		
				Level 1	Level 2	Level 3
4.1.01	256 × 256 × 3	H	0.9679	−0.0032	0.0040	0.0019
		V	0.9588	0.0065	0.0068	0.0067
		D	0.9447	0.0019	−0.0010	0.0028
5.2.10	512 × 512 × 1	H	0.9317	0.0199	−0.0054	−0.0022
		V	0.9435	−0.0029	0.0000	0.0063
		D	0.9036	−0.0075	0.0003	0.0009
4.2.05	512 × 512 × 3	H	0.9604	0.0022	−0.0044	−0.0015
		V	0.9543	0.0066	0.0036	0.0009
		D	0.9251	0.0027	−0.0011	−0.0014
5.3.01	1024 × 1024 × 1	H	0.9818	−0.0062	0.0051	0.0001
		V	0.9769	0.0074	0.0005	−0.0018
		D	0.9665	0.0087	0.0047	0.0001
Pure black	512 × 512 × 3	H	-	−0.0045	−0.0020	−0.0014
		V	-	0.0082	0.0005	−0.0065
		D	-	0.0066	0.0040	−0.0013

5.7. Information Entropy Analysis

Information entropy is a quantification of the randomness of the image pixels and helps to compare the degree of randomness of the encryption. Information entropy can be calculated as follows.

$$IE = -\sum_{i=0}^{Q} p(x_i) \log_2 p(x_i) \tag{34}$$

where Q denotes the number of gray levels, $p(x_i)$ denotes the occurrence probability of pixel x_i. For an 8-bit image, the maximum information entropy is 8. Higher information entropy indicates higher security. The information entropy of each plaintext image and ciphertext image is given in Table 3.

Table 3. Information entropy of plaintext image and ciphertext image.

Image	Size	PI	CI			Image	Size	PI	CI		
			Level 1	Level 2	Level 3				Level 1	Level 2	Level 3
5.1.11	256 × 256 × 1	6.4523	7.9971	7.9975	7.9978	5.1.12	256 × 256 × 1	6.8981	7.9977	7.9981	7.9979
1.1.01	512 × 512 × 1	6.7057	7.9993	7.9992	7.9994	5.2.10	512 × 512 × 1	7.0686	7.9993	7.9992	7.9993
1.3.01	1024 × 1024 × 1	7.6702	7.9998	7.9998	7.9998	5.3.01	1024 × 1024 × 1	7.2428	7.9999	7.9998	7.9998
4.1.01	256 × 256 × 3	5.7056	7.9990	7.9992	7.9992	4.1.05	256 × 256 × 3	6.6639	7.9998	7.9998	7.9998
4.2.01	512 × 512 × 3	7.4404	7.9997	7.9998	7.9998	4.2.05	512 × 512 × 3	7.6133	7.9999	7.9999	7.9999
2.2.01	1024 × 1024 × 3	7.5237	7.9999	7.9999	7.9999	6.2.02	1024 × 1024 × 3	6.5786	7.9999	7.9999	7.9999
Pure black	512 × 512 × 3	0	7.9997	7.9998	7.9998	Pure white	512 × 512 × 3	0	7.9998	7.9998	7.9998

5.8. Robustness Analysis

Robustness tests can detect whether the receiver is still able to obtain useful information after the image has been subjected to various levels of noise contamination or cropping attacks. We used 1% and 5% salt and pepper noise to attack the ciphertext image and show the decrypted image after suffering from noise in Figure 18. In addition, we cropped 5%, 10% of the ciphertext image and show the decrypted image after suffering the cropping attack in Figure 19. In the proposed algorithm, there are two factors that will affect the robustness. The first factor is the newly defined LL operation. Unlike addition and XOR operations, the result of encrypted pixel values can be completely different even if one bit is changed. The second factor is diffusion, because intra-block diffusion and inter-block diffusion lead to the propagation of erroneous pixel values multiple times, resulting in multiple decryption errors of the pixel values.

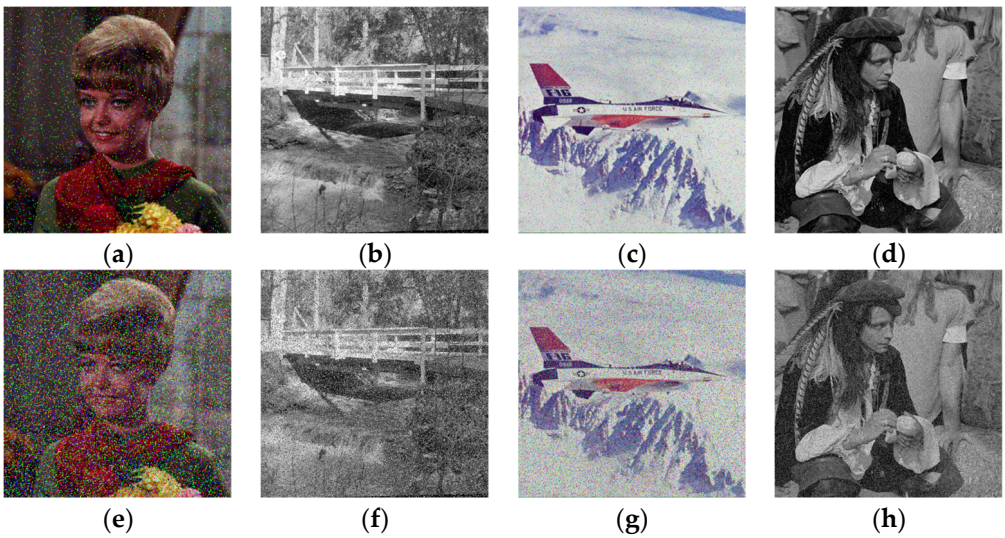

Figure 18. (**a**–**d**) Decryption image after adding 1% salt and pepper noise and (**e**–**h**) decryption image after adding 5% salt and pepper noise.

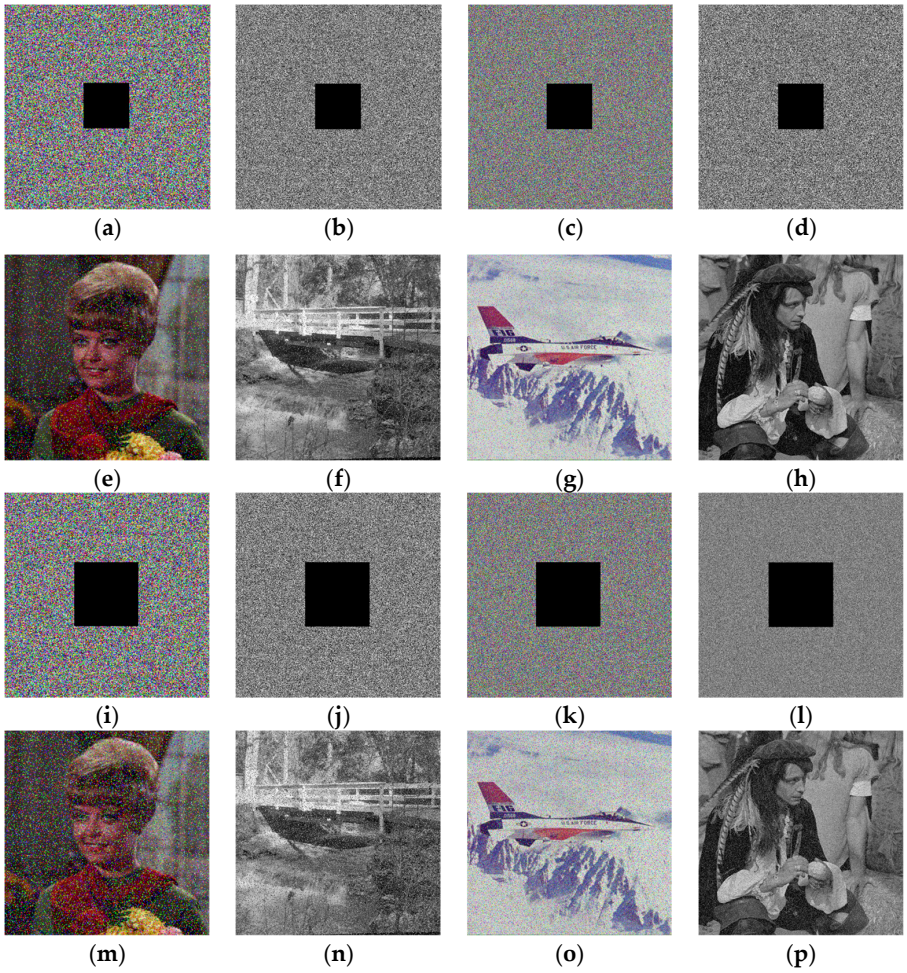

Figure 19. The 5% (**a**–**d**) cropped ciphertext images; (**e**–**h**) decryption images of (**a**–**d**); (**i**–**l**) 10% cropped ciphertext images; (**m**–**p**) decryption images of (**i**–**l**).

5.9. Speed Analysis

Encryption speed is an important factor in the performance of image encryption. In this paper, several parameters affecting the encryption time are designed so that users can choose different levels of encryption speed according to their needs. Since the size of the encrypted image varies, we use 256 × 256 as the image unit to compare the time required to encrypt each unit. The encryption times for different sizes are shown in Table 4.

Table 4. Encryption time.

Size	Level 1		Level 2		Level 3	
	Time (s)	Time (s/Unit)	Time (s)	Time (s/Unit)	Time (s)	Time (s/Unit)
256 × 256 × 1	0.064	0.064	0.121	0.121	0.229	0.229
512 × 512 × 1	0.226	0.057	0.337	0.084	0.749	0.187
1024 × 1024 × 1	0.807	0.050	1.161	0.073	2.502	0.156
256 × 256 × 3	0.171	0.057	0.264	0.088	0.579	0.193
512 × 512 × 3	0.624	0.052	0.886	0.074	1.944	0.162
1024 × 1024 × 3	2.221	0.046	3.153	0.066	6.626	0.138
Mean	-	0.054	-	0.084	-	0.178

5.10. Comparison with Other Schemes

The proposed multiple-image encryption scheme is compared with some recent related schemes. Table 5 compares NPCR, UACI, correlation coefficient, information entropy, and encryption speed. The table exhibits that the proposed algorithm has better performance along with good efficiency. The encryption performance is passable even for the relatively

low encryption Level 1. Adjusting the parameters $B_{\min}$, $B_{\max}$, and L_{num} to obtain a higher encryption level actually makes the computational complexity of the encryption increase. However, since the data are all close to the ideal values, it is difficult to see the exact difference.

Table 5. Comparison with other schemes.

Scheme	NPCR (%)	UACI (%)	Correlation			IE	Time (s/Unit)
			Horizontal	Vertical	Diagonal		
Ref. [18]	99.6229	33.4809	0.0019	0.0012	0.0020	7.9994	-
Ref. [43]	99.6060	33.5126	−0.0003	0.0011	0.0013	7.9998	0.107
Ref. [44]	-	-	0.0044	−0.0050	−0.0002	-	0.515
Ref. [45]	99.6167	33.4772	−0.0036	−0.0049	−0.0023	7.9993	-
Ref. [46]	99.6200	33.4600	0.0013	0.0009	−0.0018	7.9999	0.540
Ref. [47]	99.6085	33.4634	0.0011	0.0008	0.0015	7.9993	0.320
Ref. [48]	99.6012	33.4418	0.0015	−0.0002	−0.0004	7.9996	0.503
Ref. [49]	99.6077	33.4399	−0.0016	0.0057	−0.0189	7.9996	1.711
Ref. [50]	99.6367	33.3733	0.0116	0.0057	0.0039	7.9994	-
Ref. [51]	99.6174	33.4657	−0.0164	0.0056	0.0289	7.9993	0.123
Level 1	99.6107	33.4630	0.0022	0.0066	0.0027	7.9993	0.054
Level 2	99.6113	33.4627	−0.0044	0.0036	−0.0011	7.9994	0.084
Level 3	99.6175	33.4824	−0.0015	0.0009	−0.0014	7.9994	0.178

6. Conclusions

We propose ND-CMCM to overcome the drawbacks of the current existing chaotic systems. The chaotic model can generate chaotic maps with different dimensions to satisfy different scenarios. Simulation experiments show that the chaotic systems generated by ND-CMCM have an infinite range of chaotic parameters with positive Lyapunov exponents equal to the dimension. Our proposed newly defined operations can be used in image encryption algorithms to replace the previously used operations such as modular sum, XOR, and finite field with higher complexity. The proposed overlapping block-based permutation algorithm not only scrambles the position and value of pixels efficiently but also controls the number of pixels being encrypted by adjusting the length of the overlapping block. Our proposed image encryption algorithm gives a lot of freedom to the user: the chaotic system can be customized, the secret key can be set freely, there can be no restriction on the image type, number, and size, and different encryption levels can be set. The algorithm can be universally used in various scenarios and under different requirements to fulfill a wide range of needs. In addition, the algorithm has high efficiency while the encryption performance reaches the ideal standard and can cope with many different attacks. In future research, we will work on image encryption schemes with stronger robustness and explore ways to apply these proposed encryption schemes to daily life.

Author Contributions: Conceptualization, Z.Z.; methodology, Z.Z.; software, Z.Z. and K.S.; validation, K.S.; investigation, X.X.; data curation, X.X.; writing—original draft, Z.Z.; supervision, Z.J.; project administration, Z.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (grant numbers 61927803, 61071025, 61502538, and 61501525) and the Natural Science Foundation of Hunan Province of China (grant number 2018JJ3680).

Data Availability Statement: Not applicable.

Acknowledgments: The authors are thankful to the anonymous referees for their helpful comments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhou, N.-R.; Tong, L.-J.; Zou, W.-P. Multi-image encryption scheme with quaternion discrete fractional Tchebyshev moment transform and cross-coupling operation. *Signal Process.* **2023**, *211*, 109107. [CrossRef]
2. Xian, Y.; Wang, X. Fractal sorting matrix and its application on chaotic image encryption. *Inf. Sci.* **2021**, *547*, 1154–1169. [CrossRef]
3. Duan, C.-F.; Zhou, J.; Gong, L.-H.; Wu, J.-Y.; Zhou, N.-R. New color image encryption scheme based on multi-parameter fractional discrete Tchebyshev moments and nonlinear fractal permutation method. *Opt. Lasers Eng.* **2022**, *150*, 106881. [CrossRef]
4. Zhang, J.; Huang, Z.; Li, X.; Wu, M.; Wang, X.; Dong, Y. Quantum Image Encryption Based on Quantum Image Decomposition. *Int. J. Theor. Phys.* **2021**, *60*, 2930–2942. [CrossRef]
5. Guo, L.; Du, H.; Huang, D. A quantum image encryption algorithm based on the Feistel structure. *Quantum Inf. Process.* **2022**, *21*, 20. [CrossRef]
6. Li, J.; Zhang, Q. Compressed phase coding-based optical image encryption. *Optik* **2023**, *278*, 170744. [CrossRef]
7. Zhou, Q.; Wang, X.; Jin, M.; Zhang, L.; Xu, B. Optical image encryption based on two-channel detection and deep learning. *Opt. Lasers Eng.* **2023**, *162*, 107415. [CrossRef]
8. Kumar, K.; Roy, S.; Rawat, U.; Malhotra, S. IEHC: An efficient image encryption technique using hybrid chaotic map. *Chaos Solitons Fractals* **2022**, *158*, 111994. [CrossRef]
9. Chen, J.; Chen, L.; Zhang, L.Y.; Zhu, Z.-l. Medical image cipher using hierarchical diffusion and non-sequential encryption. *Nonlinear Dyn.* **2019**, *96*, 301–322. [CrossRef]
10. Hua, Z.; Zhou, B.; Zhou, Y. Sine Chaotification Model for Enhancing Chaos and Its Hardware Implementation. *IEEE Trans. Ind. Electron.* **2019**, *66*, 1273–1284. [CrossRef]
11. Wang, R.; Li, M.-Y.; Luo, H.-J. Exponential sine chaotification model for enhancing chaos and its hardware implementation. *Chin. Phys. B* **2022**, *31*, 080508. [CrossRef]
12. Hua, Z.; Zhou, B.; Zhang, Y.; Zhou, Y. Modular chaotification model with FPGA implementation. *Sci. China-Technol. Sci.* **2021**, *64*, 1472–1484. [CrossRef]
13. Hua, Z.; Zhang, Y.; Zhou, Y. Two-Dimensional Modular Chaotification System for Improving Chaos Complexity. *IEEE Trans. Signal Process.* **2020**, *68*, 1937–1949. [CrossRef]
14. Zhou, Z.; Xu, X.; Yao, Y.; Jiang, Z.; Sun, K. Novel multiple-image encryption algorithm based on a two-dimensional hyperchaotic modular model. *Chaos Solitons Fractals* **2023**, *173*, 113630. [CrossRef]
15. Teng, L.; Wang, X.; Yang, F.; Xian, Y. Color image encryption based on cross 2D hyperchaotic map using combined cycle shift scrambling and selecting diffusion. *Nonlinear Dyn.* **2021**, *105*, 1859–1876. [CrossRef]
16. Sun, J. 2D-SCMCI Hyperchaotic Map for Image Encryption Algorithm. *IEEE Access* **2021**, *9*, 59313–59327. [CrossRef]
17. Gao, X. Image encryption algorithm based on 2D hyperchaotic map. *Opt. Laser Technol.* **2021**, *142*, 107252. [CrossRef]
18. Patro, K.A.K.; Acharya, B. An efficient dual-layer cross-coupled chaotic map security-based multi-image encryption system. *Nonlinear Dyn.* **2021**, *104*, 2759–2805. [CrossRef]
19. Li, C.; Zhang, Y.; Li, H.; Zhou, Y. Visual image encryption scheme based on inter-intra-block scrambling and weighted diffusion. *Vis. Comput.* **2023**. [CrossRef]
20. 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]
21. Wang, X.; Xu, X.; Sun, K.; Jiang, Z.; Li, M.; Wen, J. A color image encryption and hiding algorithm based on hyperchaotic system and discrete cosine transform. *Nonlinear Dyn.* **2023**, *111*, 14513–14536. [CrossRef]
22. Li, T.; Zhang, D. Hyperchaotic Image Encryption Based on Multiple Bit Permutation and Diffusion. *Entropy* **2021**, *23*, 510. [CrossRef] [PubMed]
23. Qiu, H.; Xu, X.; Jiang, Z.; Sun, K.; Xiao, C. A color image encryption algorithm based on hyperchaotic map and Rubik's Cube scrambling. *Nonlinear Dyn.* **2022**, *110*, 2869–2887. [CrossRef]
24. Zhu, S.; Deng, X.; Zhang, W.; Zhu, C. Image Encryption Scheme Based on Newly Designed Chaotic Map and Parallel DNA Coding. *Mathematics* **2023**, *11*, 103340. [CrossRef]
25. Wang, Q.; Zhang, X.; Zhao, X. Image encryption algorithm based on improved Zigzag transformation and quaternary DNA coding. *J. Inf. Secur. Appl.* **2022**, *70*. [CrossRef]
26. Wei, D.; Jiang, M.; Deng, Y. A secure image encryption algorithm based on hyper-chaotic and bit-level permutation. *Expert Syst. Appl.* **2023**, *213*, 119074. [CrossRef]
27. Hua, Z.; Zhou, Y. Design of image cipher using block-based scrambling and image filtering. *Inf. Sci.* **2017**, *396*, 97–113. [CrossRef]
28. Wang, R.; Deng, G.-Q.; Duan, X.-F. An image encryption scheme based on double chaotic cyclic shift and Josephus problem. *J. Inf. Secur. Appl.* **2021**, *58*, 102699. [CrossRef]
29. Wang, X.; Liu, P. A New Image Encryption Scheme Based on a Novel One-Dimensional Chaotic System. *IEEE Access* **2020**, *8*, 174463–174479. [CrossRef]
30. Xu, Q.; Sun, K.; He, S.; Zhu, C. An effective image encryption algorithm based on compressive sensing and 2D-SLIM. *Opt. Lasers Eng.* **2020**, *134*, 106178. [CrossRef]
31. May, R.M. Simple mathematical models with very complicated dynamics. *Nature* **1976**, *261*, 459–467. [CrossRef] [PubMed]
32. Li, S.J.; Chen, G.R.; Mou, X.Q. On the dynamical degradation of digital piecewise linear chaotic maps. *Int. J. Bifurc. Chaos* **2005**, *15*, 3119–3151. [CrossRef]

33. Yuan, X.; Zhao, J.; Yang, Y.; Wang, Y. Hybrid parallel chaos optimization algorithm with harmony search algorithm. *Appl. Soft Comput.* **2014**, *17*, 12–22. [CrossRef]
34. Chang, L.; Lu, J.A.; Deng, X.M. A new two-dimensional discrete chaotic system with rational fraction and its tracking and synchronization. *Chaos Solitons Fractals* **2005**, *24*, 1135–1143. [CrossRef]
35. Belazi, A.; Kharbech, S.; Aslam, M.N.; Talha, M.; Xiang, W.; Iliyasu, A.M.; Abd El-Latif, A.A. Improved Sine-Tangent chaotic map with application in medical images encryption. *J. Inf. Secur. Appl.* **2022**, *66*, 103131. [CrossRef]
36. Chen, C.; Sun, K.; He, S. A class of higher-dimensional hyperchaotic maps. *Eur. Phys. J. Plus* **2019**, *134*, 410. [CrossRef]
37. Mansouri, A.; Wang, X. A novel one-dimensional sine powered chaotic map and its application in a new image encryption scheme. *Inf. Sci.* **2020**, *520*, 46–62. [CrossRef]
38. Talhaoui, M.Z.; Wang, X.; Midoun, M.A. A new one-dimensional cosine polynomial chaotic map and its use in image encryption. *Vis. Comput.* **2021**, *37*, 541–551. [CrossRef]
39. Wu, Y.; Zhou, Y.; Noonan, J.P.; Agaian, S. Design of image cipher using latin squares. *Inf. Sci.* **2014**, *264*, 317–339. [CrossRef]
40. Alvarez, G.; Li, S. Some basic cryptographic requirements for chaos-based cryptosystems. *Int. J. Bifurc. Chaos* **2006**, *16*, 2129–2151. [CrossRef]
41. Wu, Y.; Noonan, J.P.; Agaian, S. NPCR and UACI randomness tests for image encryption. *Cyber J. Multidiplinary J. Ence Technol. J. Sel. Areas Telecommun.* **2011**.
42. Liu, C.; Ding, Q. A Color Image Encryption Scheme Based on a Novel 3D Chaotic Mapping. *Complexity* **2020**, *2020*, 3837209. [CrossRef]
43. Zhang, X.; Hu, Y. Multiple-image encryption algorithm based on the 3D scrambling model and dynamic DNA coding. *Opt. Laser Technol.* **2021**, *141*, 107073. [CrossRef]
44. Sun, X.; Shao, Z.; Shang, Y.; Liang, M.; Yang, F. Multiple-image encryption based on cascaded gyrator transforms and high-dimensional chaotic system. *Multimed. Tools Appl.* **2021**, *80*, 15825–15848. [CrossRef]
45. Patro, K.A.K.; Soni, A.; Netam, P.K.; Acharya, B. Multiple grayscale image encryption using cross -coupled chaotic maps. *J. Inf. Secur. Appl.* **2020**, *52*, 102470. [CrossRef]
46. Gao, X.; Mou, J.; Banerjee, S.; Cao, Y.; Xiong, L.; Chen, X. An effective multiple-image encryption algorithm based on 3D cube and hyperchaotic map. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 1535–1551. [CrossRef]
47. Wang, X.; Liu, H. Cross-plane multi-image encryption using chaos and blurred pixels. *Chaos Solitons Fractals* **2022**, *164*, 112586. [CrossRef]
48. Gaffar, A.; Joshi, A.B.; Singh, S.; Mishra, V.N.; Rosales, H.G.; Zhou, L.; Dhaka, A.; Mishra, L.N. A Technique for Securing Multiple Digital Images Based on 2D Linear Congruential Generator, Silver Ratio, and Galois Field. *IEEE Access* **2021**, *9*, 96125–96150. [CrossRef]
49. Ye, H.-S.; Zhou, N.-R.; Gong, L.-H. Multi-image compression-encryption scheme based on quaternion discrete fractional Hartley transform and improved pixel adaptive diffusion. *Signal Process.* **2020**, *175*. [CrossRef]
50. Ul Haq, T.; Shah, T. Algebra-chaos amalgam and DNA transform based multiple digital image encryption. *J. Inf. Secur. Appl.* **2020**, *54*, 102592. [CrossRef]
51. Wu, J.; Zhang, J.; Liu, D.; Wang, X. A Multiple-Medical-Image Encryption Method Based on SHA-256 and DNA Encoding. *Entropy* **2023**, *25*, 898. [CrossRef] [PubMed]