*fractal and fractional*

MDPI

*Article*

# Research on Image Encryption Based on Fractional Seed Chaos Generator and Fractal Theory

**Haiping Chang, Erfu Wang * and Jia Liu**

School of Electronic Engineering, Heilongjiang University, Harbin 150080, China
* Correspondence: wangerfu@hlju.edu.cn

**Abstract:** In this paper, a new fractional-order seed chaotic generator is designed to solve the problem of the complex operations of single low-dimensional systems and simple high-dimensional systems. The fractional-order chaotic system generated is proven to have better chaotic performance using Lyapunov exponential differential calculus, approximate entropy, 0–1 test and other indicators. On this basis, the "multiple squares nested body scrambling (MSNBS)" model is extended from fractal theory to complete the image scrambling process, and a new algorithm is proposed to be applied to the encryption field in combination with the fractional-order coupled chaotic system. Experimental simulations show that the algorithm can resist common differential attacks and noise attacks and improve the security of the algorithm.

**Keywords:** chaos; image encryption; fractional-order

check for updates

**Citation:** Chang, H.; Wang, E.; Liu, J. Research on Image Encryption Based on Fractional Seed Chaos Generator and Fractal Theory. *Fractal Fract.* **2023**, 7, 221. https://doi.org/10.3390/fractalfract7030221

Received: 29 January 2023
Revised: 12 February 2023
Accepted: 22 February 2023
Published: 1 March 2023

**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

## 1. Introduction

With the popularity of 5G technology and the rise of 6G technology, people are increasingly inclined to use intuitive transmission methods such as pictures and videos for information interaction on the network, rather than text. Image encryption technology is the main means to protect information security and personal privacy from external threats. However, traditional image encryption algorithms are not suitable for images with miscellaneous data. The introduction of chaos theory provides a new idea and method for image encryption. For various beginning values, a chaotic system can produce unique pseudo-random sequences. The sequences traverse uniformly in the interval. Good key sensitivity can be obtained when encrypting images using the characteristics of chaos theory. However, a chaotic system is not uniformly chaotic in the whole parameter plane and will be controlled by the range of system parameters. This drawback has a negative impact on the subsequent ciphertext images to resist phase space attacks and powerful attacks. At present, most image encryption algorithms use a combination of chaotic systems and the "scrambling diffusion" framework. In response to the drawbacks of simple chaotic performance, insufficient scrambling and weak diffusion randomness of a single low-dimensional system, the three parts mentioned above are improved to enhance the encryption algorithm's ability to resist external attacks.

At present, many scholars are studying how to improve the disadvantages of low-dimensional chaotic systems, such as improving the existing one-dimensional chaotic systems and using linear methods to couple different systems. The magnitude of variables is dynamically set in [1], and the Logistic map and Sine map are cascaded to generate an I1DS chaotic sequence to expand the value range of parameters. However, this chaotic system still has the problem of blank window. Mohamed Amine Midoun performed fractional operation on cosine function and sine function, and 1DLSE [2] system generated is complex and difficult to predict. Considering the nonlinearity and high complexity of the fractional operation, the system still needs to be improved to enhance the efficiency of the algorithm. In reference [3], the 1DLSE chaotic system obtained by nesting the Logistic map

and Sine map in the way of multiplication traverses the entire parameter plane uniformly. At the same time, the introduction of control parameters expands the key space of the encryption algorithm. However, only a single 1DLSE system is used to complete the entire encryption process, making it easy for intruders to crack the original image. In the process of image encryption, it is sometimes necessary for multiple chaotic systems to generate different chaotic sequences. From the above analysis, we need to couple a wide range of chaotic systems with good chaos randomness, and we need to couple at least two new chaotic systems to realize the encryption process. Therefore, many scholars summarize the coupling methods of various literatures, take one-dimensional chaotic maps as the seed, build a general internal structure of integrated chaotic systems, create a seed chaotic generator and output multiple chaotic systems with a wide range of parameters. Zhou proposed a one-dimensional chaotic system seed generator for the first time. The integrated system is constructed by summation, and three chaotic sequences uniformly distributed in [0,1] are obtained. The Lyapunov exponent [4] of the integrated chaotic system is greater than the Lyapunov value of the seed mapping. Reference [5] proposed a coupled chaotic system based on unit transformation. As a novel chaotic model, UT-CCS has stronger seed universality than Zhou's integrated system, but the unit change function needs to be set manually. Hu proposed an ICS-I system [6] integrating three chaotic seed maps. In order to improve the unpredictability of this model, a circuit switch was designed to build the ICS-II system on its basis, which improved its chaotic performance to a certain extent. However, the output of the chaotic integrated system still does not have a continuous chaotic range. The problem of blank window also makes the output chaotic system unable to achieve the desired effect, and the structure of the system is complex. It is inconsistent with the original intention of constructing a seed generator to address the complex operation of a high-dimensional chaotic system. Based on this, this paper proposes a method based on cascading, modular and exponential operation to build the model of seed chaotic generator and completes the derivation of the Lyapunov exponent concerning reference [6] (see Section 2.2 for details). The model proposed in this paper takes the classical Logistic map, Sine map and Tent map as the seeds and arbitrarily selects two seeds as the input of the generator. Under the control parameters, the sensitivity of the output chaotic system is enhanced, and the key space is increased, which can play an important role in the subsequent encryption algorithms.

The correlation between plaintext images may become a breakthrough for criminals to attack ciphertext images. Therefore, we need to use scrambling methods to break the relationship between images and improve the security of ciphertext images. Scrambling achieves the best effect by changing the position of pixels and reducing the correlation between adjacent pixels. Existing scrambling algorithms include DNA scrambling [7,8], bit scrambling [9,10], pixel scrambling [11–13] and chaotic mapping scrambling [14,15]. Reference [16] proposed a spiral scanning scrambling method based on chaotic sub-blocks to overcome the shortcoming that the general spiral scanning algorithm cannot change the position of each pixel. Although the improved algorithm changes the position of each pixel, the relative position of the elements in the block matrix has not changed. In order to further hide plaintext information, reference [17] generates scrambled ciphertext through two rounds of dynamic L-type scrambling and Arnold scrambling. Although the ciphertext changes the position of plaintext pixels, the correlation between adjacent pixels of the ciphertext image has not been broken, and the algorithm requires two rounds of the scrambling process to achieve the desired scrambling effect, which not only increases the complexity of the algorithm but also reduces the encryption efficiency. For human organ images, medical images and military images containing sensitive information, it is necessary to completely hide the size and position of plaintext pixels. In order to prevent attackers from stealing and attacking images from ciphertext image lines, it is necessary to ensure that the pixels are fully scrambled in the scrambling process. Therefore, this paper proposes a scrambling method based on the fractal theory of multi-square nested volume (MSNBS). The algorithm applies the improved Josephus scrambling to the MSNBS

algorithm, realizes the complete and sufficient scrambling of plaintext images and improves the security of the encryption algorithm.

The structure and contents of this paper are as follows: Section 2 introduces the internal architectures of the seed chaotic generator in detail. Section 3 explains the principle of the MSNBS algorithm. Section 4 presents the algorithm flow of the entire image encryption. Section 5 analyzes the security performance of the algorithm. Finally, Section 6 summarizes the contents of this paper.

## 2. New Fractional-Order Seed Chaotic Generator

### *2.1. Internal Integration Structure of Seed Generator*

In this paper, the Logistic map, Sine map and Tent map are used as the inputs of the seed generator. Any two of them are used as the system inputs to build an internal integration framework of the seed generator. The characteristics of the seed chaotic mapping are as follows:

1. Logistic map:

$$x_{n+1} = 4\mu x_n(1 - x_n), \tag{1}$$

2. Sine map:

$$x_{n+1} = \mu \sin(\pi x_n), \tag{2}$$

3. Tent map:

$$x_{n+1} = \begin{cases} 2\mu x_n, x_n < 0.5 \\ 2\mu(1 - x_n), x_n \geq 0.5 \end{cases}, \tag{3}$$

where $\mu \in [0, 1]$. For Logistic system, the system enters into chaos when $\mu \in [0.9, 1]$; Sine map has good chaotic behavior when $\mu \in [0.87, 1]$; The chaotic range of Tent map is $\mu \in [0.5, 1]$. One noteworthy point about the Tent mapping is that it can fall into periodic behavior due to floating-point operations. In order to avoid this problem, we should select an appropriate initial value, such as 0.4999 instead of 0.5, to avoid the occurrence of periodic behavior. In this paper, the Tent map is used to combination with other chaotic maps, and the problems caused by rounding errors can also be solved. In conclusion, the above three classical chaotic maps are controlled by a single parameter, and the parameters must be within a certain range for the system to enter the chaotic state.

In this paper, the chaotic systems mentioned above are used as inputs to the fractional-order seed generator to complete the coupling and then the output. The internal integration structure of the fractional-order seed chaos generator is shown in Figure 1, and the formulae for the digitization of this model are defined as shown in Equation (4).

$$x_{n+1} = \Gamma(x_n) = \left(e^{\lambda(R(r,x)+S(1-r,x))} + R(r, S(1 - r, x))\right) \mathrm{mod} 1, \tag{4}$$

where $x_{n+1}$ is the output of the seed chaos generator, $R(r, x)$ and $S(1 - r, x)$ represent the seed chaotic map with parameters $r$ and $1 - r$, respectively. $\lambda$ is a constant parameter for controlling a chaotic system. To ensure good chaotic behavior, the value range of $\lambda$ is $[5, +\infty)$.

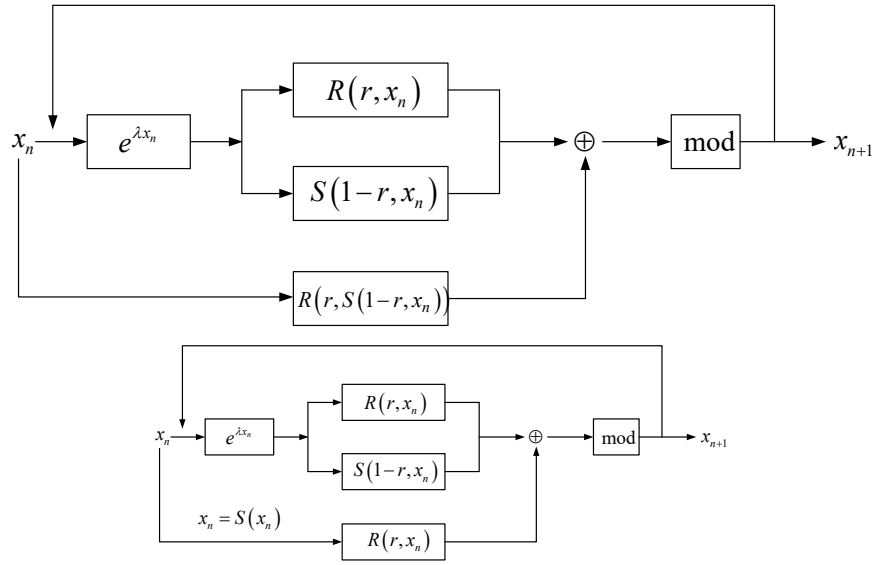



**Figure 1.** Internal integration structure block diagram of seed generator.

Based on the pairwise combination of three seed maps, six new chaotic systems can be generated. Using Logistic and Sine map as inputs can generate two new chaotic systems due to different cascade positions and parameters: LCSCM (Logistic cascade Sine chaotic map) and SCLCM (Sine cascade Logistic chaotic map). Similarly, when Logistic and Tent map, Sine and Tent map are used as inputs, the following chaotic systems can be generated: LCTCM (Logistic cascade Tent chaotic map), SCTCM (Sine cascade Tent chaotic map) and TCSCM (Tent cascade Sine chaotic map), respectively. The expressions of seed mapping types and generated chaotic systems are shown in Table 1. See Table 2 for the corresponding seed mapping types of the output chaotic system.

**Table 1.** The output of seed chaos generator.

| Name | New Chaotic Systems Excited by Seed Generator |
|---|---|
| LCSCM | $x_{n+1} = \left(e^{\lambda[4\mu x_n(1-x_n)+(1-\mu)\sin(\pi x_n)]} + 4\mu(1-\mu)\sin(\pi x_n)(1-(1-\mu)\sin(\pi x_n))\right)\bmod 1$ |
| SCLCM | $x_{n+1} = \left(e^{\lambda[4(1-\mu)x_n(1-x_n)+\mu\sin(\pi x_n)]} + \mu\sin(4\pi(1-\mu)x_n(1-x_n))\right)\bmod 1$ |
| LCTCM | $x_{n+1} = \begin{cases} \left(e^{\lambda[4\mu x_n(1-x_n)+2(1-\mu)x_n]} + 8\mu(1-\mu)x_n(1-2x_n)\right)\bmod 1, x_n < 0.5 \\ \left(e^{\lambda[4\mu x_n(1-x_n)+2(1-\mu)(1-x_n)]} + 8\mu(1-\mu)(1-x_n)(1-2(1-\mu)(1-x_n))\right)\bmod 1, x_n \geq 0.5 \end{cases}$ |
| TCLCM | $x_{n+1} = \begin{cases} \left(e^{\lambda[2\mu x_n+4(1-\mu)x_n(1-x_n)]} + 8\mu(1-\mu)x_n(1-x_n)\right)\bmod 1, x_n < 0.5 \\ \left(e^{\lambda[2\mu(1-x_n)+4(1-\mu)x_n(1-x_n)]} + 2\mu(1-4(1-\mu)x_n(1-x_n))\right)\bmod 1, x_n \geq 0.5 \end{cases}$ |
| SCTCM | $x_{n+1} = \begin{cases} \left(e^{\lambda[\mu\sin(\pi x_n)+2(1-\mu)x_n]} + \mu\sin(2\pi(1-\mu)x_n)\right)\bmod 1, x_n < 0.5 \\ \left(e^{\lambda[\mu\sin(\pi x_n)+2(1-\mu)(1-x_n)]} + \mu\sin(2\pi(1-\mu)(1-x_n))\right)\bmod 1, x_n \geq 0.5 \end{cases}$ |
| TCSCM | $x_{n+1} = \begin{cases} \left(e^{\lambda[(1-\mu)\sin(\pi x_n)+2\mu x_n]} + 2\mu(1-\mu)\sin(\pi x_n)\right)\bmod 1, x_n < 0.5 \\ \left(e^{\lambda[(1-\mu)\sin(\pi x_n)+2\mu(1-x_n)]} + 2\mu(1-(1-\mu)\sin(\pi x_n))\right)\bmod 1, x_n \geq 0.5 \end{cases}$ |

**Table 2.** Corresponding seed mapping type of output chaotic system.

| Name | R(x) | S(x) |
|---|---|---|
| LCSCM | Logistic | Sine |
| SCLCM | Sine | Logistic |
| LCTCM | Logistic | Tent |
| TCLCM | Tent | Logistic |
| SCTCM | Sine | Tent |
| TCSCM | Tent | Sine |

### *2.2. Theoretical Derivation and Analysis of Lyapunov Exponent*

LE (Lyapunov exponent) is a measure of the average convergence or divergence of similar orbits in phase space, and it is also an important indicator to determine whether there is dynamic chaos in the system [18]. The system enters a chaotic state when LE is greater than 0, and the larger the LE, the more sensitive the chaotic system is to the initial value [19]. The formula for the Lyapunov exponent is shown in Equation (5):

$$\varepsilon = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \ln\left|f'(x_i)\right|, \tag{5}$$

In the following section, the definition of the Lyapunov exponent will be combined with the principle of differential derivatives and the newly generated fractional-order seed chaos generator Lyapunov exponent values will be compared with those reported in the literature [6]. It is worth mentioning that the control parameters added can adjust the LE value of the generated chaotic system. However, the effect of these parameters is not considered in the inference process. First, two very close initial values are taken, and one iteration is performed using Equation (4). Then, the difference between the two values after iterating is calculated using the following formula:

$$|x_1 - y_1| = \left| e^{\lambda(R(x_0)+S(x_0))} - e^{\lambda(R(y_0)+S(y_0))} + R(S(x_0)) - R(S(y_0)) \right|, \tag{6}$$

After arranging Formula (6), we obtain:

$$|x_1 - y_1| = \left| \begin{array}{l} \frac{e^{\lambda(R(x_0)+S(x_0))} - e^{\lambda(R(y_0)+S(y_0))}}{\lambda(R(x_0)+S(x_0)-R(y_0)-S(y_0))} \times \\ \frac{\lambda(R(x_0)+S(x_0)-R(y_0)-S(y_0))}{x_0-y_0} + \\ \frac{R(S(x_0))-R(S(y_0))}{S(x_0)-S(y_0)} \times \frac{S(x_0)-S(y_0)}{x_0-y_0} \end{array} \right| |x_0 - y_0|, \tag{7}$$

Since $x_0 \to y_0$, then correspondingly $R(x_0) \to R(y_0)$ and $R(x_0) + S(x_0) \to R(y_0) + S(y_0)$. According to the definition of derivative, Formulas (8)–(10) can be obtained:

$$\left|\frac{de^{\lambda x}}{dx}|_{R(x_0)+S(x_0)}\right| \approx \lim_{R(x_0)+S(x_0)\to R(y_0)+S(y_0)} \left( \begin{array}{l} \left|\frac{e^{\lambda(R(x_0)+S(x_0))} - e^{\lambda(R(y_0)+S(y_0))}}{\lambda(R(x_0)+S(x_0)-R(y_0)-S(y_0))}\right| \times |\lambda| \\ \times \left( \left| \frac{R(x_0)-R(y_0)}{x_0-y_0} + \frac{S(x_0)-S(y_0)}{x_0-y_0} \right| \right) \end{array} \right) \tag{8}$$

$$\left|\frac{dR}{dx}|_{S(x_0)}\right| \approx \lim_{S(x_0)\to S(y_0)} \left| \frac{R(S(x_0)) - R(S(y_0))}{S(x_0) - S(y_0)} \times \frac{S(x_0) - S(y_0)}{x_0 - y_0} \right| \tag{9}$$

$$\left|\frac{dS}{dx}|_{x_0}\right| \approx \lim_{x_0\to y_0} \left| \frac{S(x_0) - S(y_0)}{x_0 - y_0} \right| \tag{10}$$

Therefore, Formula (7) can be expressed:

$$|x_1 - y_1| \approx \left( \left|\frac{de^{\lambda x}}{dx}|_{R(x_0)+S(x_0)}\right| + \left( \left|\frac{dR}{dx}|_{S(x_0)}\right| \times \left|\frac{dS}{dx}|_{x_0}\right| \right) \right) |x_0 - y_0| \tag{11}$$

The difference of absolute values generated after n iterations of initial values $x_0$ and $y_0$ is shown in Formula (12):

$$|x_n - y_n| \approx \left( \prod_{i=0}^{n} \left| \frac{de^{\lambda x}}{dx}|_{R(x_i)+S(x_i)} \right| + \prod_{i=0}^{n} \left| \frac{dR}{dx}|_{S(x_i)} \right| \times \prod_{i=0}^{n} \left| \frac{dS}{dx}|_{x_i} \right| \right) |x_0 - y_0| \tag{12}$$

We denote the rate of change of $\Gamma(x)$ at the position $x_n$ and $y_n$ as $\Delta\Gamma(x)$:

$$\Delta\Gamma(x) \approx \left( \prod_{i=0}^{n} \left| \frac{de^{\lambda x}}{dx}|_{R(x_i)+S(x_i)} \right| + \prod_{i=0}^{n} \left| \frac{dR}{dx}|_{S(x_i)} \right| \times \prod_{i=0}^{n} \left| \frac{dS}{dx}|_{x_i} \right| \right)^{\frac{1}{n}} \tag{13}$$

The Lyapunov expression obtained from the definition in Formula (5) is:

$$\varepsilon = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \ln \left( \left| \frac{de^{\lambda x}}{dx}|_{R(x_i)+S(x_i)} \right| + \left| \frac{dR}{dx}|_{S(x_i)} \right| \times \left| \frac{dS}{dx}|_{x_i} \right| \right) \tag{14}$$

The LE of ICS model in literature [6] is shown in Formula (15):

$$\varepsilon = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \ln \left( \left| \frac{dH}{dx}|_{x_i} \right| + \left| \frac{dF}{dx}|_{x_i} \right| \times \left| \frac{dG}{dx}|_{x_i} \right| \right), \tag{15}$$

The $\left| \frac{dR}{dx}|_{S(x_i)} \right| \times \left| \frac{dS}{dx}|_{x_i} \right|$ part from Lyapunov exponent of the chaotic system generated in this paper represents two chaotic systems in cascade, which has the same mathematical meaning as $\left| \frac{dF}{dx}|_{x_i} \right| \times \left| \frac{dG}{dx}|_{x_i} \right|$ in Formula (15). Therefore, the results of the comparison between the LE of [6] and this paper depend on $\left| \frac{dH}{dx}|_{x_i} \right|$ and $\left| \frac{de^{\lambda x}}{dx}|_{R(x_0)+S(x_0)} \right|$, where $\left| \frac{de^{\lambda x}}{dx}|_{R(x_0)+S(x_0)} \right|$ is the derivative of two chaotic systems at $e^{\lambda x}$.

For the convenience of comparison, x in formula 14 represents a function of x. In other words, it represents $x_{n+1}$ in Formula (4), so we use $H(x)$ to express $x$. We set the derivation object of both as $H(x)$, then the value of reference [6] is $\left| \frac{dH}{dx}|_{x_i} \right|$, and the value of this paper is $|\lambda| \cdot \left| e^{\lambda H(x)} \right| \cdot \left| \frac{dH}{dx}|_{x_i} \right|$. According to the monotonicity of the exponential function with e as the base, when $x \in [0, +\infty)$, and then $e^x \in [1, +\infty)$. Therefore, we can conclude that $\left| e^{\lambda H(x)} \right| \geq 1$. The value range of $\lambda$ mentioned earlier is $[5, +\infty)$. To sum up, the value of $|\lambda| \cdot \left| e^{\lambda H(x)} \right| \cdot \left| \frac{dH}{dx}|_{x_i} \right|$ is greater than $\left| \frac{dH}{dx}|_{x_i} \right|$. Finally, it is proved that the LE of this paper is greater than that of the ICs model. Therefore, the chaos generated in this paper has better characteristics and is more sensitive to the initial value.

As can be seen from Figure 2, except for the LCSCM chaotic system that is greater than 0 in the range of $(0.191, 1]$, The remaining chaotic systems are greater than 0 in the whole range. Compared with the original single chaotic map, the range of parameters is expanded. When the minimum value of $\lambda$ is taken as 5, the maximum Lyapunov exponent generated is greater than the value in reference [6], and the value of other systems is in the range of $[1, 2]$. However, the maximum LE value of the generated six chaotic systems can almost reach about 5. Therefore, the chaotic performance and sensitivity to the initial value of the seed generator of the chaotic system in this paper are better than the output of the ICS model. The generator in this paper only needs to complete "two out of three" to generate an output better than that in the literature [6], which reduces the computational complexity, simplifies the generator model, and obtains a more chaotic integrated system.

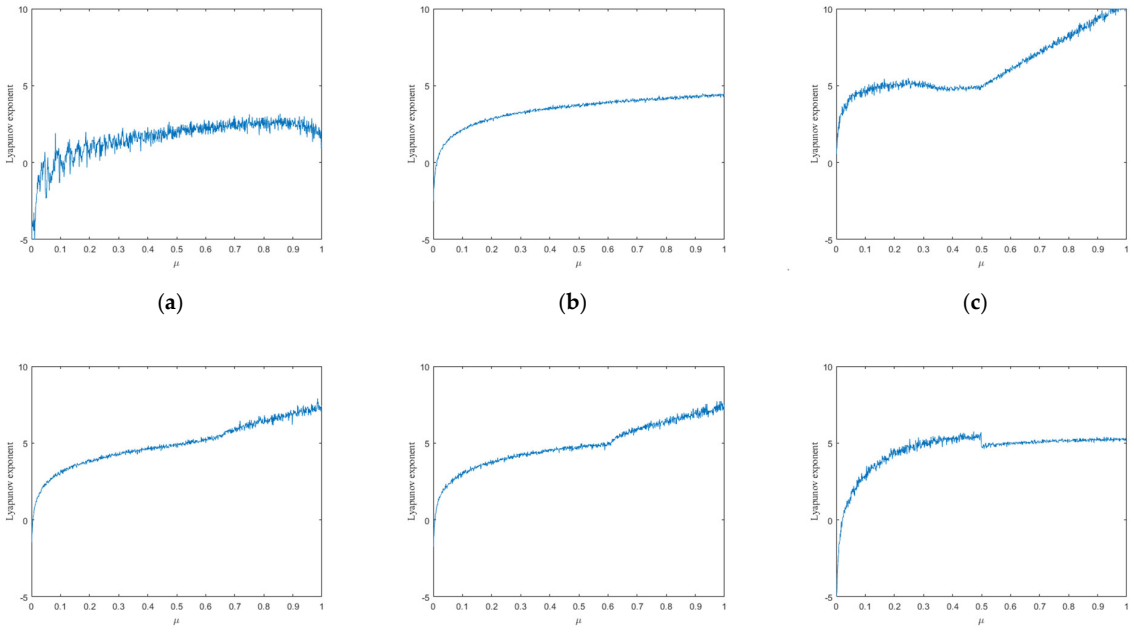



**Figure 2.** Lyapunov exponent graph of (**a**) LCSCM, (**b**) SCLCM, (**c**) LCTCM, (**d**) TCLCM, (**e**) TCSCM and (**f**) SCTCM.

### *2.3. Bifurcation Diagram*

The bifurcation diagram is used to depict how the dynamic system's trajectory changes as a result of the control parameters. A more uniform traversal of the bifurcation diagram indicates better chaotic performance of the chaotic system. The distribution of the system within the parameter range is visually shown in the bifurcation diagram. Figure 3 describes the bifurcation diagrams of the six created chaotic systems. The system moves consistently over the parameter range, as can be observed. The ideal chaotic effect can be attained, except for a narrow blank window issue in Figure 3a, provided the other chaotic system parameters are arbitrarily chosen to fall between [0,1].

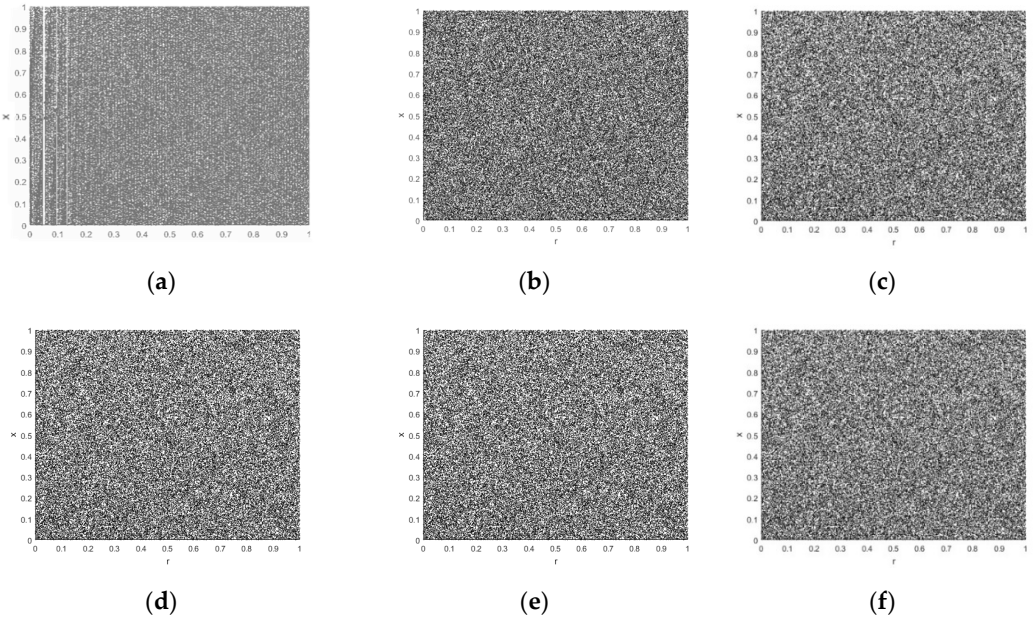



**Figure 3.** Bifurcation diagram of (**a**) LCSCM, (**b**) SCLCM, (**c**) LCTCM, (**d**) TCLCM, (**e**) TCSCM and (**f**) SCTCM.

### *2.4. 0–1 Test*

The 0–1 test is an algorithm to test whether the system has chaotic characteristics according to the output. For the output sequence $\phi(n)$, where $n = 1, 2, \ldots, N$, the calculation formula is shown in Equation (16):

$$K = corr(\xi, \Delta) = \frac{\mathrm{cov}(\xi, \Delta)}{\sqrt{\mathrm{var}(\xi)\mathrm{var}(\Delta)}}, \tag{16}$$

where $\xi = 1, 2, \ldots, n$, $\Delta = M(1), M(2), \ldots, M(n)$. $M(n)$ is the mean square displacement and their calculation expressions are as follows:

$$\mathrm{cov}(x, y) = \frac{1}{q}\sum_{i=1}^{q}(x(i) - \overline{x})(y(i) - \overline{y}), \mathrm{var}(x) = \mathrm{cov}(x, x) \tag{17}$$

$$M(n) = \lim_{N\to\infty}\frac{1}{N}\sum_{i-1}^{N}\left([p(i+n) - p(i)]^2 + [q(i+n) - q(i)]^2\right) - (E\phi)^2\frac{1 - \cos(nr)}{1 - \cos r} \tag{18}$$

$$p(n) = \sum_{i=1}^{n}\phi(i)\cos(ir), q(n) = \sum_{i=1}^{n}\phi(i)\sin(ir), E\phi = \lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N}\phi(i) \tag{19}$$

When the value of $K$ is close to one, it indicates that the system is chaotic. We test the six chaotic systems generated by the chaotic seed generator for the 0–1 test. As observed in Figure 4, the value of the new chaotic system is close to one, indicating that it performs more chaotic behavior than the previous chaotic systems.

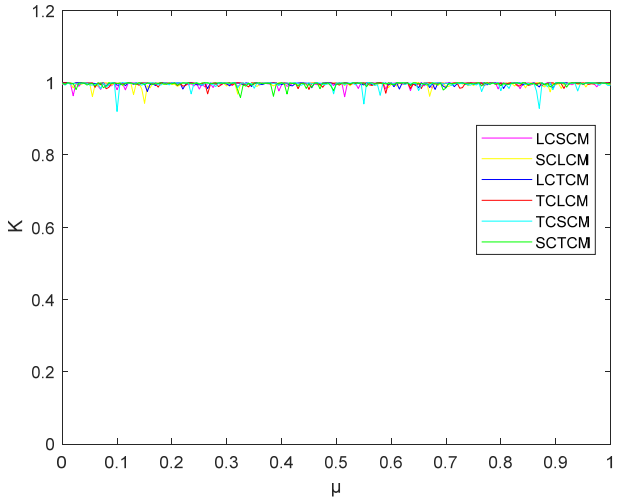



**Figure 4.** The 0–1 Test.

### *2.5. Approximate Entropy*

Approximate entropy (ApEn) is an important parameter used to characterize the complexity and irregularity of chaotic systems. The larger the value of ApEn, the higher the complexity of the system and the better its chaotic characteristics. Figure 5 shows a comparison of ApEn values for six newly generated chaotic systems and seed chaotic maps. From the figure, it can be observed that the approximate entropy of the new chaotic system is greater than the value of seed chaos, indicating that the chaotic characteristics of the new systems generated by the seed chaotic generator are better than those of a single seed chaotic map.

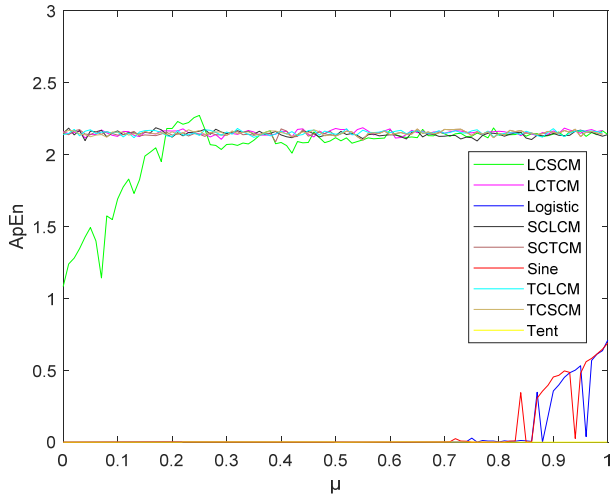



**Figure 5.** Approximate entropy.

## 3. MSNBS Scrambling Algorithm

### *3.1. Improved Josephus Loop*

The Josephus problem is summarized into a mathematical model where individuals are numbered in turn to form a circle. Each person counts off in sequence, and the person who counts to $l$ leaves the circle. Then the counting resumes from the next person who leaves until everyone leaves. The output is the position number of the person who leaves, generating the Josephus sequence [14]. The Josephus function expression, defined in reference [20], is $f(S,l,r)$. $S$ is the number of elements, $l$ is the cycle step and $r$ is the starting position. If the input sequence is $\{1,2,3,4,5,6,7,8,9,10\}$, the Josephus traversal is completed under the premise of $l = 5$ and $r = 3$, and the output sequence is $\{7,2,8,4,1,10,3,6,9,5\}$. The specific internal process is shown in Figure 6. The elements marked in orange indicate that the input and output sequences have the same elements at the same position.

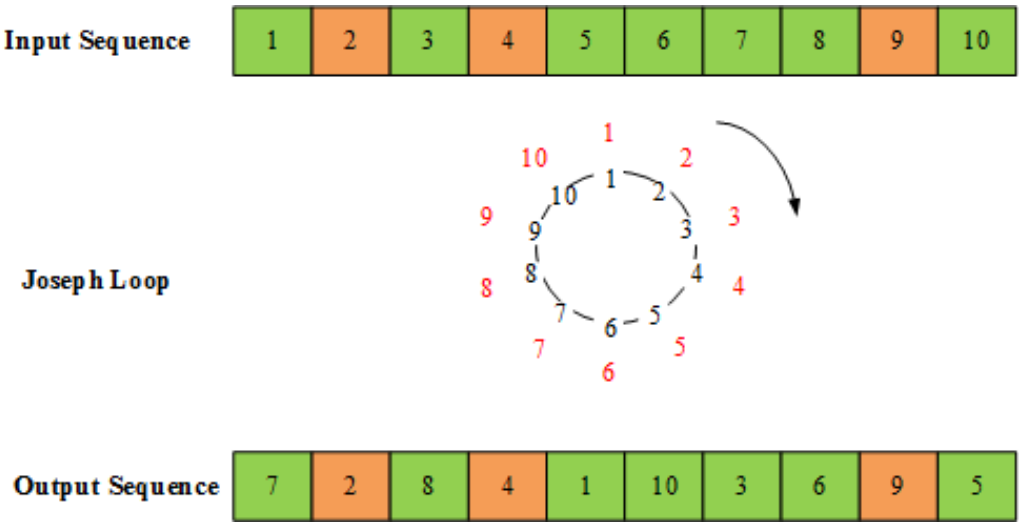



**Figure 6.** Josephus loop.

Wang applied the Josephus loop to image encryption. However, to enhance the connection between Josephus scrambling and plaintext, he changed the step size from a fixed value to the mean value of plaintext pixels, but the improved Josephus scrambling could not achieve sufficient scrambling [21]. To ensure that the position of each pixel changes when the image is dislocated, we improve the Josephus loop. Once an element completes the

Josephus loop, it is determined whether the input elements at the corresponding positions of the output elements are the same. If they are the same, the element is skipped, and the output starts at the next position until all elements have completed the Josephus loop.

### *3.2. Multi Square Nested Body Based on Fractal Theory*

Similarity is a linear transformation that contracts or expands geometry at the same rate in all directions, whereas affine is a linear transformation that contracts or expands at different rates in different directions. Affine is a non-uniform linear transformation, while similarity is a uniform linear transformation and is a special case of affine. Fractals under the action of self-similarity are called linear fractals and are classified as strictly linear fractals, statistical linear fractals, and random linear fractals. The strict linear fractal is strictly self-similar, i.e., there is infinite nesting in mathematics. Therefore, based on the strict linear fractal theory, this paper divides the image matrix into multiple square nested bodies of different specifications, named MSNB-I and MSNB-II. As shown in Figure 7, different square nested bodies are identified in red in the two models, which contain four layers of squares with different side lengths. The pixels on the square layer participate in the process of intra-block scrambling and inter-block scrambling.

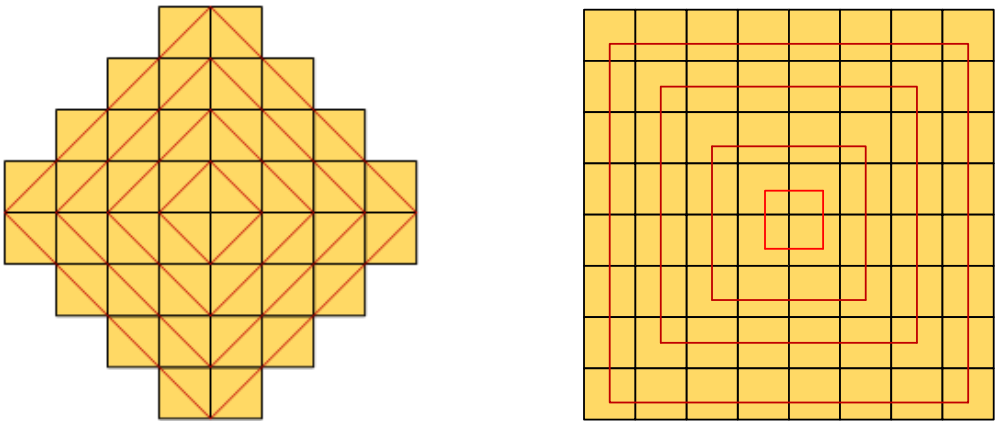



**Figure 7.** Multi square nested body model. (**a**) MSNBS-I; (**b**) MSNB-II.

### *3.3. Description of Intra-Block Scrambling and Inter-Block Scrambling*

In this paper, the processes of intra-block scrambling and inter-block scrambling are collectively referred to as the MSNBS algorithm. Firstly, intra-block scrambling is completed using the MSNB-I model and a modified Josephus loop to achieve a change in the position of each pixel in the plaintext. Secondly, inter-block scrambling is completed using the MSNB-II model and the index matrix to achieve full scrambling of the plaintext pixel correlation. The following is a detailed explanation of intra-block scrambling and inter-block scrambling.

Block-internal scrambling: Taking the following $8 \times 8$ square matrix as an example, when as many elements as possible are on the "square layer", more and wider position choices are obtained when scrambling. Therefore, for an $8 \times 8$ matrix, according to Section 2.2, the MSNB-I model generates a nested body containing four levels of squares. Each layer can form a Josephus ring. In order to ensure that the position of each element in the $8 \times 8$ square matrix changes, we nest the square elements outside the body and reconstruct them into a new Josephus ring, as shown in Figure 8. The operation steps are as follows:

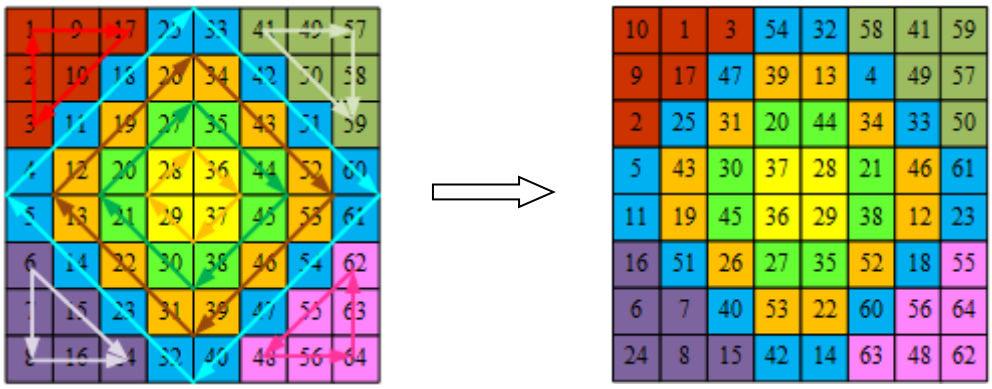



**Figure 8.** Pixels before and after scrambling.

Step 1: Set the initial position $r$ and Josephus distance $l$ of the Josephus ring.

Step 2: Pass the input sequence through the improved Josephus cycle, and the resulting output sequence is shown in Table 3.

**Table 3.** Input and output sequence of Josephus ring.

| Parameter | Input Sequence | Output Sequence |
|---|---|---|
| r = 6<br>l = 4 | B = 33→42→51→60→61→54→47→<br>40→32→23→14→5→4→11→18→25<br>O = 34→43→52→53→46→39→31→<br>22→13→12→19→26<br>G = 35→44→45→38→30→21→20→27 | B = 32→4→33→61→23→18→60→<br>14→42→40→51→11→5→25→47→54<br>O = 13→34→46→12→52→22→53→<br>26→19→43→31→39<br>G = 44→21→38→35→27→45→30→20 |
| r = 2<br>l = 3 | Y = 36→37→29→28<br>R = 1→9→17→10→3→2<br>P = 41→49→57→58→59→50<br>H = 6→7→8→16→24→15<br>F = 48→56→64→63→62→55 | Y = 28→29→36→37<br>R = 10→1→3→17→2→9<br>P = 58→41→59→57→50→49<br>H = 16→6→24→8→15→7<br>F = 63→48→62→64→55→56 |

Step 3: Put the output sequence of Step 2 into the scrambled matrix according to the position index of the Josephus ring.

Inter-block scrambling: The scrambled intermediate ciphertext in the completed block is re-divided into multiple square nested bodies according to the MSNB-II model. The square nested body is divided into four layers from the inside to the outside. First, set a position index matrix $A_1 = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$ for the innermost square. To break the correlation of adjacent elements between each layer of the square nested body, the index matrix with four layers scrambled should ensure that there are no duplicate elements at the same position. Therefore, this paper uses the method of matrix element rotation to generate index matrices with different elements at the same position. Rotate the elements 90° clockwise to generate the position index matrix $A_2 = \begin{pmatrix} 2 & 1 \\ 4 & 3 \end{pmatrix}$ of the second inner layer, and similarly, generate the position index matrices $A_3 = \begin{pmatrix} 4 & 2 \\ 3 & 1 \end{pmatrix}$ and $A_4 = \begin{pmatrix} 3 & 4 \\ 1 & 2 \end{pmatrix}$ of the other two layers. It can be seen from Figure 9 that the elements of the same level of the four sub-blocks complete the inter-block scrambling of each layer under the control of different index matrices.

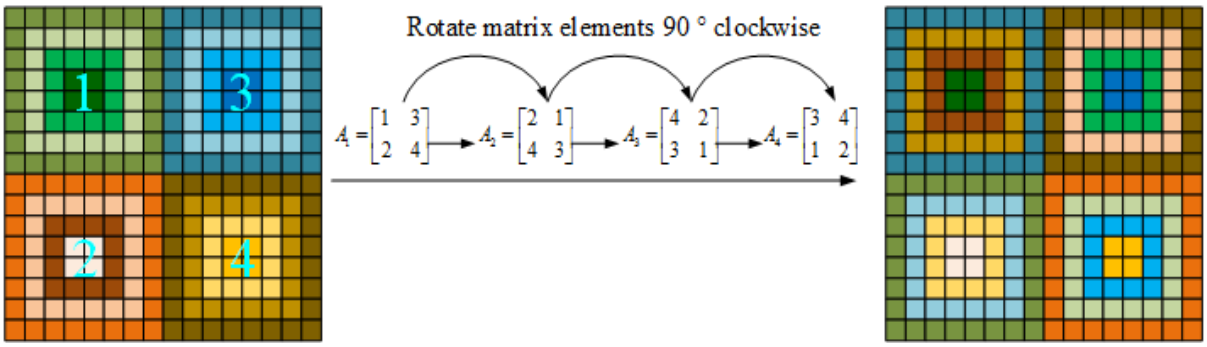



**Figure 9.** Inter block scrambling.

## 4. MSNBS Image Encryption Algorithm Based on LCSCM and SCLCM

This study offers an MSNBS image encryption algorithm based on the LCSCM and SCLCM chaotic systems formed by the new seed chaotic generator in Section 1 and the multi-square nested body model in Section 2. Image preprocessing and blocking, key creation, multi-square scrambling, and random diffusion are the primary stages of the procedure. Figure 10 shows the flowchart of the encryption algorithm in this paper. This work uses symmetric encryption, which reverses the encryption and decryption processes. This paper will not provide extensive details about the decryption method due to space constraints.

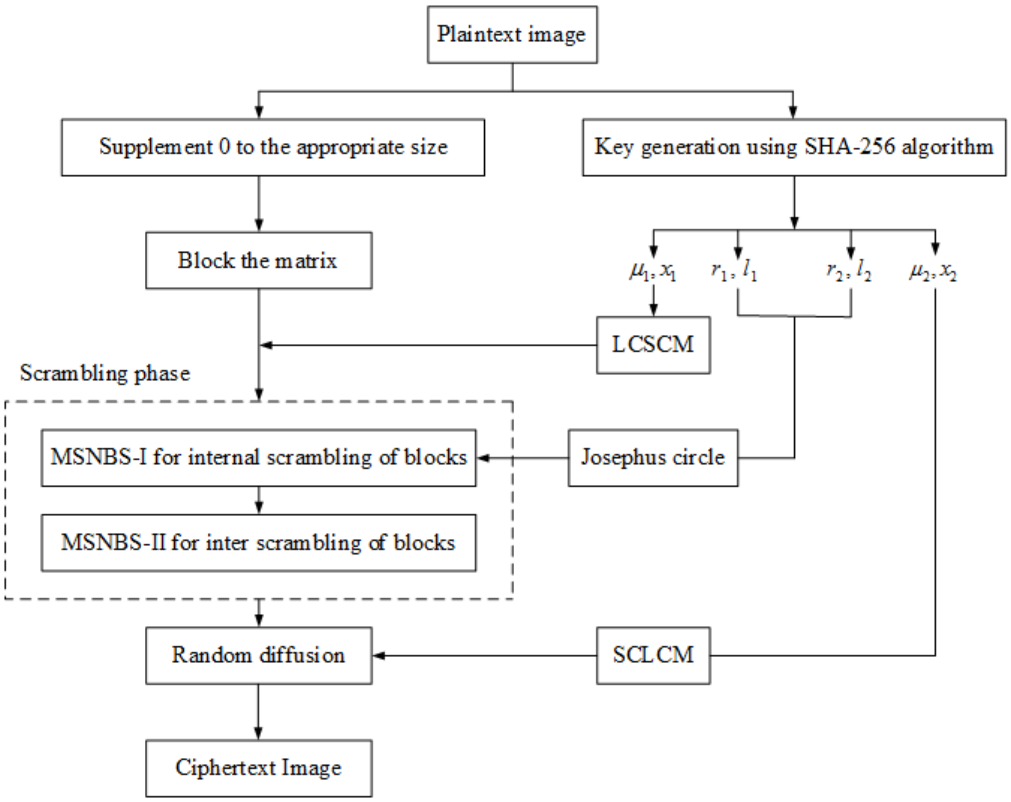



**Figure 10.** Algorithm encryption flowchart.

### *4.1. Image Preprocessing and Blocking*

Step 1: In order to ensure the smooth implementation of the subsequent encryption algorithm, we use the zero filling method to preprocess the plaintext image with the size of $M \times N$, and convert the original image into an image with a size of $M' \times N'$, where $M'$ and $N'$ are multiples of 16, respectively.

Step 2: Block the image matrix. This article uses a 256 × 256 gray-scale image as an example to illustrate the blocking process. First, the plaintext image is divided into 256 sub-blocks with a size of 16 × 16. Using the definition of the cell() function in MATLAB, the sub-blocks are stored in 1 × 256 cell arrays. The 16 × 16 sub-blocks in the cell array are further divided to obtain four 8 × 8 image matrices. Therefore, a 1 × 4 cell array is nested in each cell array, and an 8 × 8 matrix is stored in the array.

### *4.2. Key Generation*

The hash function can generate a binary output sequence based on the plaintext image. In this paper, the SHA-256 algorithm and plaintext pixel mean are used to generate the key. The sensitivity of the hash function to the initial image and the correlation between the key and plaintext improve the ability of the algorithm to resist plaintext attacks [22]. We set 256 × 256 images as the input of the SHA-256 function to generate a 256-bit sequence. The sequence is divided into groups every 16 bits to generate 16 groups, which are recorded as:

$$K = K_1 K_2 K_3 \cdots K_{16}, \tag{20}$$

The sum of the pixels of the plaintext image is recorded as $sum$, and the average value of the pixels is:

$$avg = \frac{sum}{256 \times 256}, \tag{21}$$

Formula (22) is used to generate the initial value and control parameters of the chaotic system, as well as the starting position and cycle distance of the Josephus cycle:

$$\begin{cases} \mu_1 = \mathrm{mod}\big((avg + bin2dec(K_1 \oplus K_3 \oplus K_5))/2^{16}, 1\big) \\ \mu_2 = \mathrm{mod}\big((avg + bin2dec(K_2 \oplus K_4 \oplus K_6))/2^{16}, 1\big) \\ x_1 = (bin2dec(K_7) + avg + M' \times N' \times 16)/sum \\ x_2 = (bin2dec(K_8) + avg + M' \times N' \times 16)/sum \\ r_1 = \mathrm{mod}(bin2dec(K_9 \oplus K_{10}), 8) + 1 \\ r_2 = \mathrm{mod}(bin2dec(K_{11} \oplus K_{12}), 8) + 1 \\ l_1 = \mathrm{mod}(bin2dec(K_{13} \oplus K_{14}), 4) + 1 \\ l_2 = \mathrm{mod}(bin2dec(K_{15} \oplus K_{16}), 4) + 1 \end{cases}, \tag{22}$$

### *4.3. Image Scrambling*

In the second section of this paper, the MSNBS algorithm is explained. The scrambling process changes all the positions of the pixels of the plaintext image and breaks the correlation of the plaintext image to generate the intermediate ciphertext image, which paves the way for the subsequent image diffusion to generate the ciphertext image. The main steps of applying MSNBS to the image encryption algorithm are as follows:

Step 1: LCSCM chaotic system uses $\mu_1$ and $x_1$ generated in Section 3.2 to iterate 500 times to eliminate the transient effect, and then generates a one-dimensional chaotic sequence $Q$ containing 1024 elements. After sorting $Q$ from small to large, it generates a position sequence $W$.

Step 2: Inter-block scrambling. The inter-block scrambling is performed in four matrices under the same cell array. First, the matrix elements corresponding to the MSNBS-II model are extracted. The position matrix of the innermost layer is $\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$. The rotation of the matrix elements is completed by using *circshift()* to generate the derived position matrix of the other three layers. The pixel values of the same level are scrambled between blocks according to the position sequence to generate the intermediate ciphertext. This process is described in detail in Section 2.3.

### *4.4. Random Diffusion*

Diffusion is a method of dispersing plaintext redundancy into ciphertext. A good diffusion mechanism extends the change of a single pixel in the image to the whole. That

is, the influence of a single plaintext or key bit is expanded to more ciphertext as much as possible, which not only hides the statistical relationship but also makes it more difficult for attackers to seek plaintext redundancy. At present, the diffusion algorithm often adopts a combination of forward diffusion and reverse diffusion. However, this method first arranges the image matrix to be diffused in order, and then XOR the adjacent position elements and random values to change the pixel size. Therefore, this diffusion method using sequential positions has low security and is easy to be cracked by attackers. Based on the disadvantages of the above methods, previous studies [23–25] have improved this problem. Reference [26] proposed a random diffusion algorithm for the first time, using the LDMLNCML system to generate two index chains, and the diffusion process is carried out randomly according to the value of the index chain. Based on this, this paper proposes a new random diffusion strategy that divides the plaintext image into two parts. The two parts are mutually diffused through the SCLCM chaotic system and position matrix. The diffused pixels are determined by two non-adjacent pixels. This random diffusion method improves the security of diffusion. The diffusion process is shown in Figure 11, using $2 \times 8$ images as an example to illustrate the diffusion process:

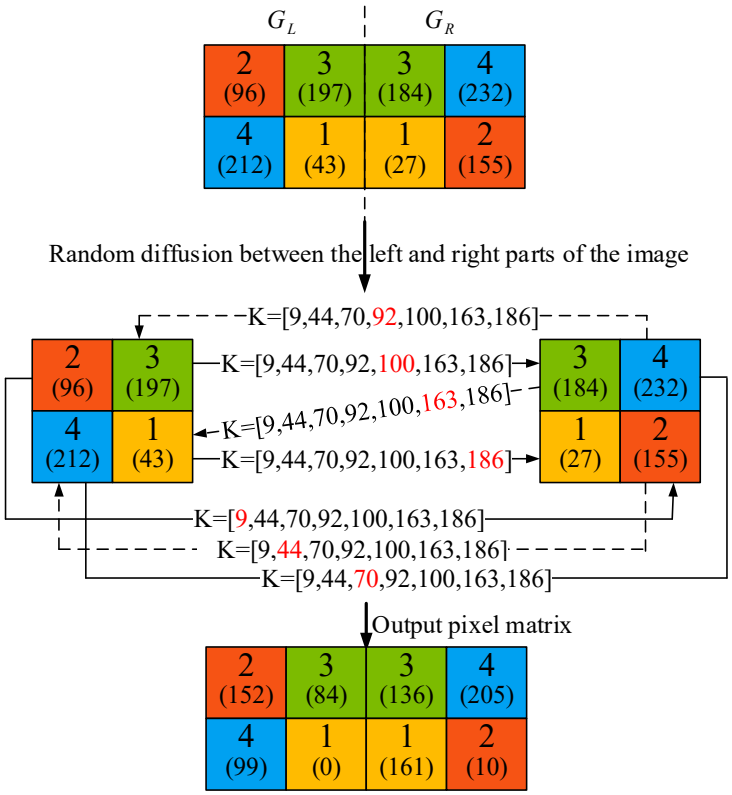



**Figure 11.** Random diffusion process.

Step 1: We divide the image matrix into two parts: $G_L$ and $G_R$. The two parts are sorted from small to large to generate their respective position matrices. We record the position matrix obtained after completing the sorting of the matrix as $F$. The numbers in the brackets represent the pixel values, and the numbers above the brackets are the elements in the corresponding position matrix. $G_L$ and $G_R$ represent the same position matrix elements with the same color.

Step 2: We bring the keys $x_2$ and $\mu_2$ into the SCLCM system and iterate 500 times to eliminate the transient reaction to obtain the chaotic sequence $K = [9, 44, 70, 92, 100, 163, 186]$. Each element in the chaotic sequence K will participate in the diffusion process in order, and we will mark the chaotic elements used in each round of diffusion with red.

Step 3: We explain the elements involved in the two directions of diffusion. When diffusing from $G_L$ to $G_R$, the elements involved in diffusion come from the pixel values and

chaotic sequence values with the same color on the left and right; while when diffusing from $G_R$ to $G_L$, the value obtained from the previous round of diffusion is different from the next element of G1 and the random value in the chaotic sequence $K$, or the diffusion is completed. We take Figure 11 as an example to illustrate the diffusion process, using ${G_L}'(1) = (sum + avg) \bmod 2^8$ to generate the diffusion value of the first element of $G_L$ and record it as ${G_L}'(1)$. Then ${G_L}'(1)$ XOR with the element with the same color in $G_R$ which is 155, and continue XOR with the first element in $K$. The result of the operation is 10. The value 99 after diffusion is obtained by XOR 10 with the second element 212 of $G_L$ and $K(2)$. The output pixel matrix can be obtained by successively diffusing according to the above law. Formula (23) is the expression of two-way diffusion.

### *4.5. Decryption Process*

This paper uses a symmetric encryption algorithm, where decryption is the reverse process of encryption. During decryption, input the same key as the encryption process, and then complete the diffusion, inter-block scrambling, and inter-block scrambling in order to obtain the initial plaintext image. The specific decryption steps are as follows:

Step 1: Enter the ciphertext image and the key.

Step 2: Use the key to generate the chaotic system parameters and the chaotic sequence of SCLCM map after the iteration. The ciphertext image is obtained by XOR operation according to the following formula.

$$\begin{cases} G_R(F(i)) = {G_R}'(i) \oplus K(i) \oplus G_L(i) \\ G_L(i+1) = {G_R}'(i) \oplus {G_L}'(i+1) \end{cases} i \geq 2$$

Step 3: Extract the corresponding position elements of each layer according to the known MSNBS-II model and complete the inverse process of inter-block scrambling with the index matrix $A_1 = \begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix} \rightarrow A_2 = \begin{bmatrix} 4 & 2 \\ 3 & 1 \end{bmatrix} \rightarrow A_3 = \begin{bmatrix} 2 & 1 \\ 4 & 3 \end{bmatrix} \rightarrow A_4 = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$.

Step 4: According to the MSNBS-I model mentioned above and the inverse process of Joseph cycle, complete the corresponding position inverse scrambling process.

Step 5: Use the key to generate the LCSCM chaotic system parameters, generate the corresponding position matrix, and complete the final scrambling process to obtain the final plaintext image.

## 5. Security Analysis

In this paper, the simulation test is carried out on the 64-bit windows 10 operating system with Intel (R) Core (TM) i5-6200u CPU @ 2.30ghz and 4 GB memory. The algorithm is written in the MATLAB R2019b program. On the premise that the key has been obtained, the encrypted image and decrypted image of the plaintext image obtained by the encryption algorithm are shown in Figure 12. The encrypted image is a disorganized snowflake image, as shown by the simulation results, and the plaintext image contour and other information cannot be extracted from it. The information from the plaintext image can be recovered by the decoded image. The plaintext image and the decrypted image are visually identical.

### *5.1. Histogram*

The distribution of image pixels is shown in the histogram. The attacker of the image can gain the necessary data about the plaintext image by statistically analyzing the histogram of the ciphertext image. To ensure the unity and consistency of the pixel distribution, it should be ensured that the histogram of the ciphertext image is as flat and uniform as possible. Histograms of the "Lena," "plane," "boat," "cameraman" and the encrypted photos are shown in Figure 13.

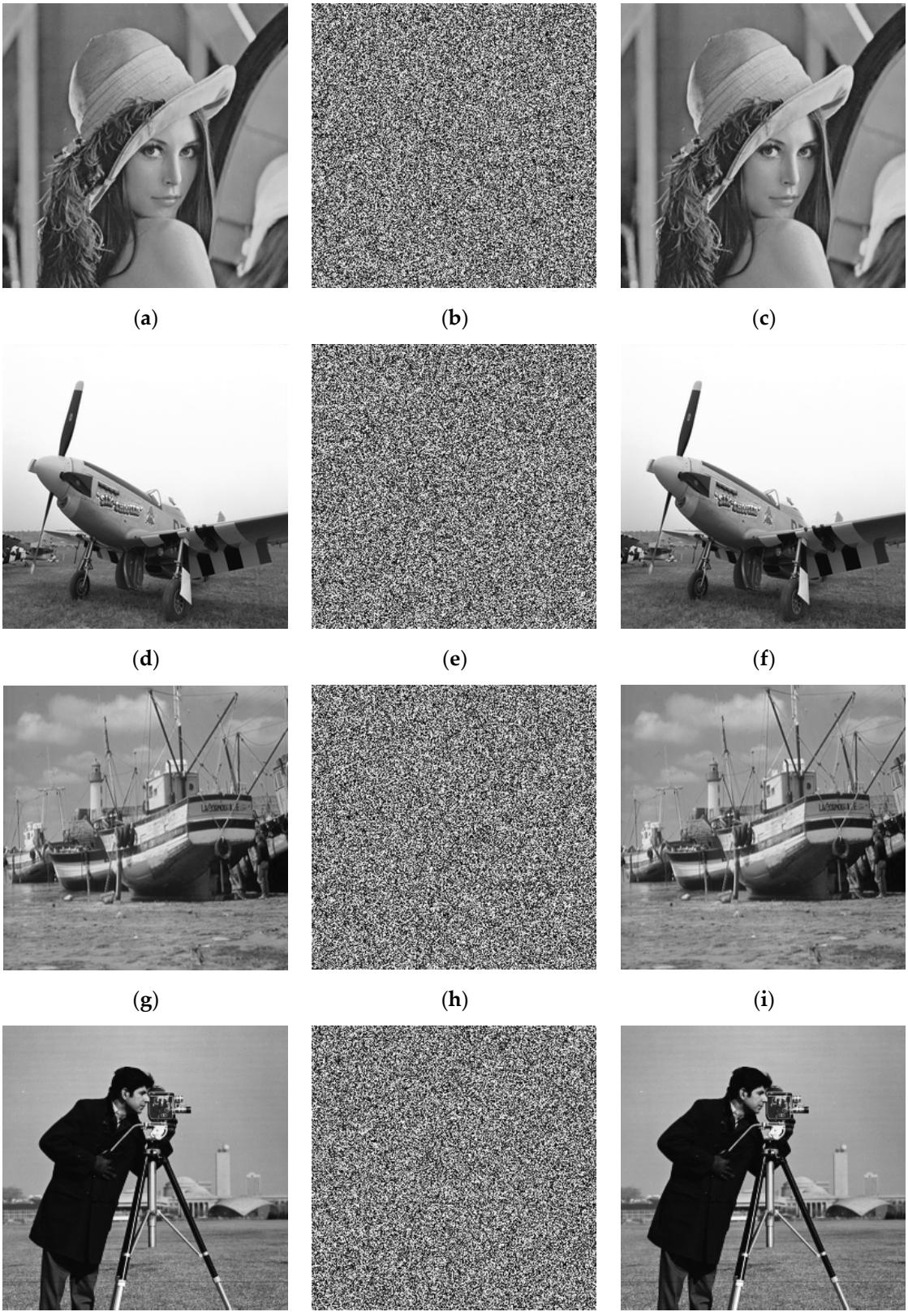



**Figure 12.** Encryption and decryption effect. (**a**–**c**) Encryption and decryption effect of Lena image. (**d**–**f**) Encryption and decryption effect of Plane image. (**g**–**i**) Encryption and decryption effect of Boat image. (**j**–**l**) Encryption and decryption effect of Cameraman image.

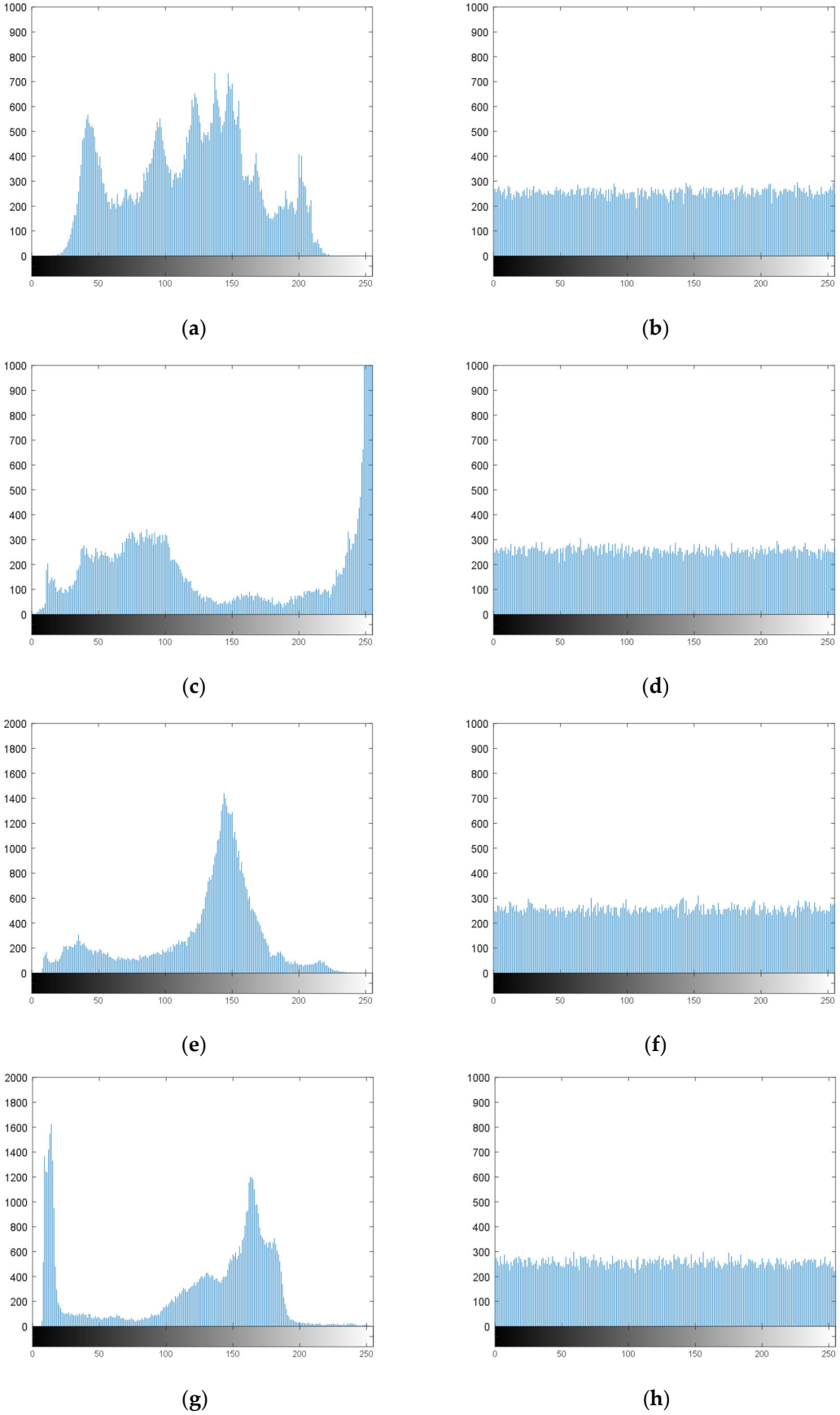


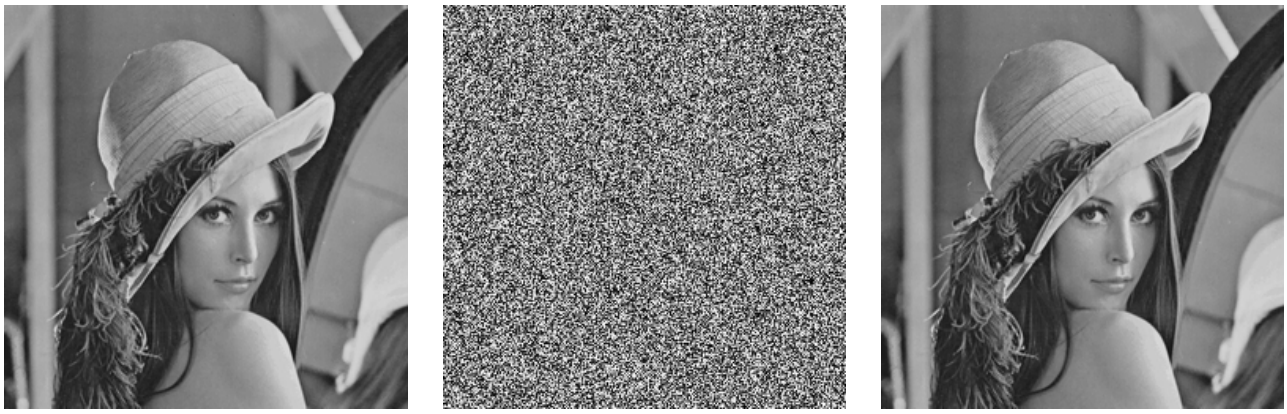

**Figure 13.** Comparison between plaintext image and ciphertext image of (**a**,**b**) Lena image, (**c**,**d**) plane image, (**e**,**f**) boat image and (**g**,**h**) cameraman image.

### 5.2. Key Space

A crucial determinant of how successfully encryption can withstand brute force is whether or not the key space is greater than $2^{100}$. The ciphertext image's resistance to exhaustive attacks increases with the size of the key space. The key space is $2^{256}$ significantly larger than $2^{100}$ in this study since the SHA-256 technique is used to create a 256-bit key stream. Consequently, this algorithm has the capacity to fend against violent assaults.

### 5.3. Key Sensitivity

The accurate key is crucial for decryption. Key sensitivity means that even a small inaccuracy in the key will prevent the decryption algorithm from restoring the proper original image. Let the correct initial key be $KEY$. If the calculation accuracy of the computer can reach $10^{-14}$, the changed key is $KEY' = KEY + 10^{-14}$. The decrypted image obtained by decrypting Lena's image with different keys is shown in Figure 14.



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

**Figure 14.** (**a**) Original image. (**b**) Decrypted image with $KEY'$. (**c**) Decrypted image with $KEY$.

### 5.4. Correlation of Adjacent Pixels

Strong correlations may be seen in the horizontal, vertical, and diagonal axes of the plaintext image. The three-directional correlation should be close to zero with the encrypted image. The formula illustrates the correlation coefficient $\rho_{xy}$ calculation process in Equation (23).

$$\rho_{xy} = \frac{\mathrm{cov}(x,y)}{\sqrt{DXDY}}, \tag{23}$$

where $\mathrm{cov}(x,y)$ and $DX$ represent the covariance and variance of pixel values, respectively:

$$\mathrm{cov}(x,y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - EX)(y_i - EY), \tag{24}$$

$$DX = \frac{1}{N}\sum_{i=1}^{N}(x_i - EX)^2, \tag{25}$$

To complete the experimental simulation in this article, 5000 pairs of pixels from the Lena image and the ciphertext image are randomly chosen. The correlation distribution plot is shown in Figure 15. The correlation of the ciphertext image is almost zero, as shown in Figure 15. Table 4 compares the correlation coefficients of Lena's pictures between this paper and previous papers, while Table 5 displays the test results for the correlation coefficients.

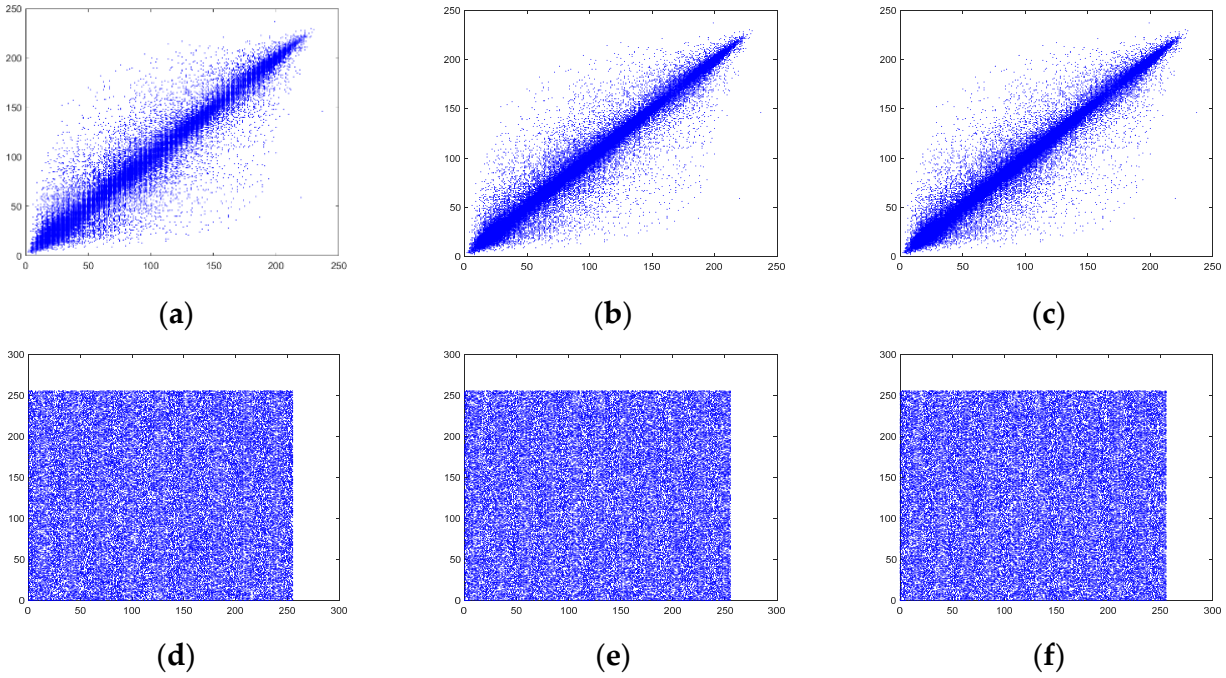


**Figure 15.** Correlation distribution of plaintext image and ciphertext image. (**a**,**d**) Horizontal direction (**b**,**e**) Vertical direction (**c**,**f**) Diagonal direction.

**Table 4.** Comparison of Lena image correlation coefficient with other literatures.

| Direction | Proposed | Ref. [27] | Ref. [28] | Ref. [29] | Ref. [30] | Ref. [16] |
|---|---|---|---|---|---|---|
| Horizontal | −0.0003 | 0.0010 | 0.0104 | −0.0028 | 0.0040 | 0.0080 |
| Vertical | 0.0021 | 0.0054 | 0.0062 | 0.0035 | −0.0012 | −0.0055 |
| Diagonal | −0.0030 | 0.0056 | −0.0056 | −0.0015 | −0.0021 | 0.0040 |

**Table 5.** Correlation coefficient.

| Image | Horizontal | | Vertical | | Diagonal | |
|---|---|---|---|---|---|---|
| | Plaintext | Ciphertext | Plaintext | Ciphertext | Plaintext | Ciphertext |
| Lena | 0.9390 | −0.0003 | 0.9681 | 0.0021 | 0.9135 | −0.0030 |
| Plane | 0.9823 | −0.0002 | 0.9869 | −0.0043 | 0.9758 | −0.0007 |
| Boat | 0.9172 | 0.0018 | 0.9391 | 0.0050 | 0.8708 | −0.0028 |
| Cameraman | 0.9333 | 0.0041 | 0.9590 | −0.0005 | 0.9085 | −0.0009 |

### *5.5. Information Entropy*

Information entropy measures how randomly distributed an image's pixels are. For 256-level grayscale images, the algorithm's capacity to fend off statistical attacks increases when the ciphertext image's information entropy value approaches eight. The computation formula can be seen in Equation (26).

$$H(u_i) = \sum_{i=1}^{n} p(u_i) \log_2 \frac{1}{p(u_i)}, \tag{26}$$

where $p(u_i)$ is the probability of occurrence, and $n$ represents the pixel level. Table 6 shows the information entropy, and Table 7 shows the comparison between Lena and other algorithms.

**Table 6.** Information entropy.

| Image | Plaintext | Ciphertext |
|---|---|---|
| Lena | 7.4204 | 7.9972 |
| Plane | 7.5411 | 7.9973 |
| Boat | 7.1612 | 7.9970 |
| Cameraman | 7.0193 | 7.9971 |

**Table 7.** Comparison of information entropy with other algorithms.

| Algorithm | Proposed | Ref. [12] | Ref. [31] | Ref. [32] |
|---|---|---|---|---|
| Entropy | 7.9972 | 7.9971 | 7.9972 | 7.9961 |

### *5.6. Differential Attack*

The resistance of the algorithm to differential attack is mainly reflected in that when the plaintext information changes slightly, the ciphertext image changes greatly. The degree of change is measured by NPCR and UACI. For 256-level gray-scale images, the ideal values of NPCR and UACI are 99.6094% and 33.4635%, respectively. The closer the degree of change of the algorithm is to the ideal value, the stronger its ability to resist differential attack. The NPCR and UACI of the experimental simulation pictures in this paper are shown in Table 8, and Table 9 is the comparison results with other literatures.

$$NPCR = \frac{\sum_{i=1}^{W}\sum_{j=1}^{H} D(i,j)}{W \times H} \times 100\%, \tag{27}$$

$$UACI = \frac{\sum_{i=1}^{W}\sum_{j=1}^{N} |C_1(i,j) - C_2(i,j)|}{W \times H \times 255} \times 100\%, \tag{28}$$

$$D(i,j) = \begin{cases} 1, C_1(i,j) \neq C_2(i,j) \\ 0, C_1(i,j) = C_2(i,j) \end{cases}, \tag{29}$$

**Table 8.** NPCR and UACI.

| Parameter | Lena | Plane | Boat | Cameraman | Mean |
|---|---|---|---|---|---|
| NPCR (%) | 99.6079 | 99.6078 | 99.5834 | 99.6017 | 99.6002 |
| UACI (%) | 33.5046 | 33.5476 | 33.4535 | 33.3855 | 33.4728 |

**Table 9.** Comparison.

| Parameter | Ideal Value | Proposed | Ref. [33] | Ref. [34] | Ref. [17] |
|---|---|---|---|---|---|
| NPCR (%) | 99.6094 | 99.6079 | 99.6300 | 99.6207 | 99.6058 |
| UACI (%) | 33.4635 | 33.5046 | 33.4300 | 33.4125 | 33.4580 |

### *5.7. Shearing Attack*

If the encryption algorithm can resist cropping attacks in the process of image transmission, the receiver can still recover the original image according to the remaining data. We change some pixel values to 0 to simulate the ciphertext image after some pixels are lost. Figure 16 shows the ciphertext image and the corresponding decrypted image after cutting 1/8, 1/4 and 1/2. It can be seen from the figure that even when half of the data of the ciphertext image are lost, the decrypted image can still reflect the information of the original image.

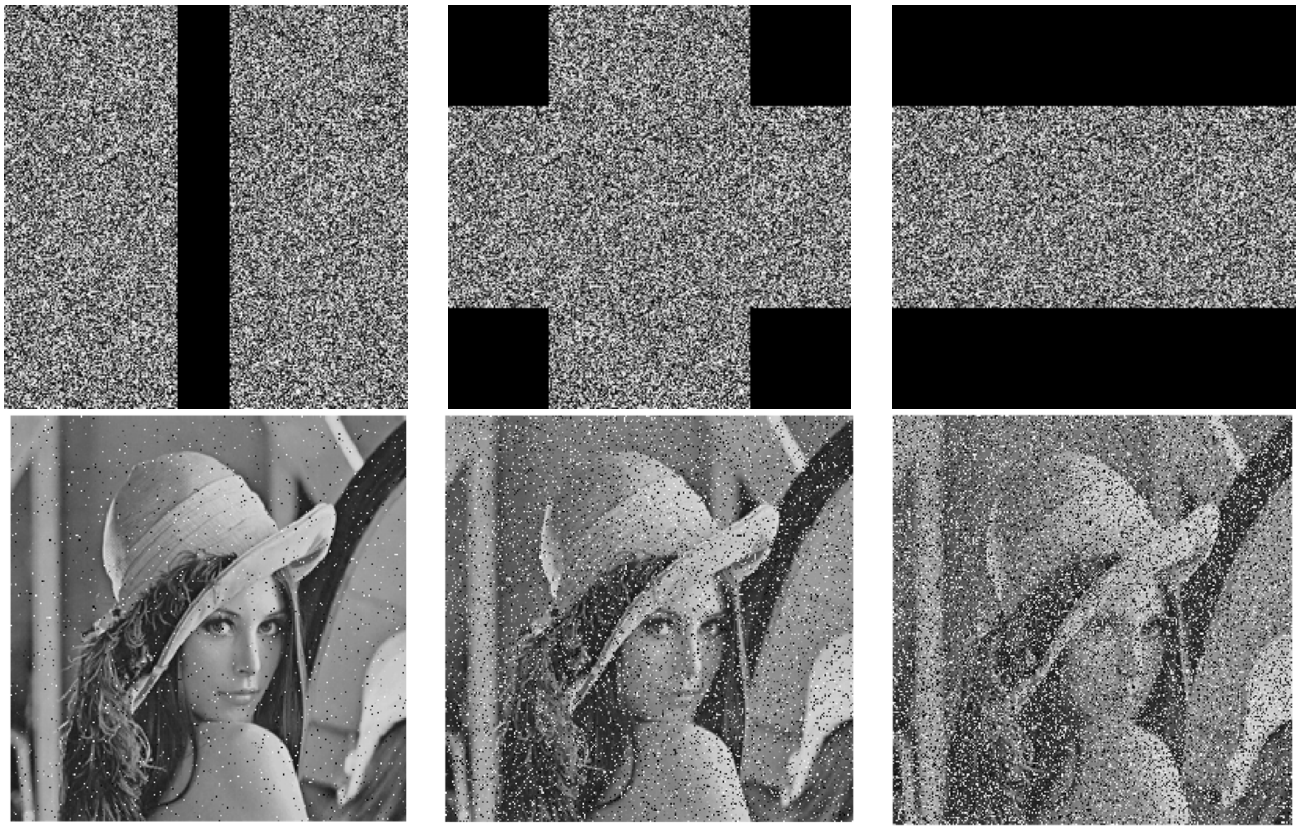


**Figure 16.** Clipping attack. (**a**) 1/8 occlusion, (**b**) 1/4 occlusion, (**c**) 1/2 occlusion.

### *5.8. Noise Attack*

Noise during the channel transmission procedure will have an impact on the image. To represent external attacks, we simulate them by adding varying levels of Gaussian and Salt-and-pepper noise to the ciphertext image. The decrypted image can be obtained while being subjected to various levels of noise, and its peak signal-to-noise ratio (PSNR) can be determined. The higher the peak signal-to-noise ratio is, the better the anti-noise effect of this method.

The size of *MSE* has a significant role in the *PSNR* calculating procedure (mean square error). The method of calculation is as follows:

$$PSNR = 20 \times \lg \frac{255}{\sqrt{MSE}}, \tag{30}$$

$$MSE = \frac{\sum_{i=1}^{W} \sum_{j=1}^{H} (P(i,j) - C(i,j))^2}{W \times H}, \tag{31}$$

where $W \times H$ is the size of the plaintext image. $P(i,j)$ and $C(i,j)$ are the pixel size before and after encryption, respectively. The test results of this algorithm against noise attack are shown in Figure 17, and Table 10 shows the PSNR values of ciphertext images.

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

(**d**) (**e**) (**f**)

**Figure 17.** Gaussian noise attack with (**a**) $\sigma^2 = 0.005$, (**b**) $\sigma^2 = 0.05$ and (**c**) $\sigma^2 = 0.2$; Salt-and-pepper noise attack with (**d**) $\sigma^2 = 0.005$, (**e**) $\sigma^2 = 0.05$ and (**f**) $\sigma^2 = 0.2$.

**Table 10.** PSNR.

| Type | Intensity | PSNR (dB) |
|---|---|---|
| Gaussian noise | 0.005 | 14.0834 |
| | 0.05 | 11.7679 |
| | 0.2 | 10.5901 |
| Salt-and-pepper noise | 0.005 | 32.2771 |
| | 0.05 | 22.4233 |
| | 0.2 | 16.1928 |

### *5.9. Analysis of Algorithm Efficiency*

When applying the encryption algorithm, we need to consider the efficiency of the algorithm, which is mainly reflected in the encryption time and the complexity of the algorithm. The algorithm in this paper is simulated by MATLAB software. The encryption time of Lena, plane, boat and cameraman are 0.456, 0.572, 0.511 and 0.421 s, respectively. Table 11 shows the comparison between Lena image encryption time and other algorithms. The image encryption process of the algorithm can be divided into the following three stages: chaotic sequence generation, scrambling and diffusion. Assuming that the size of the image we process is M × N, the complexity of LCSCM mapping and SCLCM mapping generated in this paper is O(2 × M × N). The scrambling process is divided into built-in scrambling and inter-block scrambling, and pixel-level position scrambling is used, so the time complexity at this stage is O(2 × M × N). In the third stage, the diffusion process is completed by XOR with the elements in the chaotic sequence, and the complexity is O (M × N). To sum up, the total complexity of the algorithm encryption process in this paper is O (5 × M × N).

**Table 11.** Comparison and analysis of encryption time.

| Algorithm | Our | Ref. [35] | Ref. [36] | Ref. [14] |
|---|---|---|---|---|
| Encrypted time | 0.456 | 0.926 | 0.707 | 0.563 |

## 6. Conclusions

In this paper, a chaotic system seed generator based on the exponential operation is proposed to generate six composite low-dimensional chaotic systems. After simulation analysis of LE, bifurcation, 0–1 test, and approximate entropy, it is shown that the chaotic performance and randomness of the new system are better than those of a single seed map. The coupled chaotic system is used to complete the MSNBS scrambling and random diffusion process, which makes the plaintext image fully scrambled and breaks the correlation of plaintext pixels. The ciphertext image undergoes relevant simulation tests, proving that the algorithm can resist differential attack and selective plaintext attacks. Therefore, the algorithm in this paper has a good encryption effect. Considering the application of real-value chaos in cryptography, many scholars [37] have proposed that the consistency of encryption and decryption should be taken into account in the application. We will explore the floating-point compatibility in subsequent research.

**Author Contributions:** Conceptualization, H.C., J.L. and E.W.; methodology, H.C.; software, H.C.; validation, J.L., E.W. and H.C.; investigation, J.L. and E.W.; writing—original draft preparation, H.C.; writing—review and editing, H.C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by Heilongjiang University Postgraduate Innovative Research Project, grant number no. YJSCX2022-207HLJU and Natural Science Foundation of Heilongjiang Province (LH2019F048).

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

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

## References

1. Zhou, W.; Wang, X.; Wang, M.; Li, D. A new combination chaotic system and its application in a new Bit-level image encryption scheme. *Opt. Lasers Eng.* **2022**, *149*, 106782. [CrossRef]
2. Midoun, M.A.; Wang, X.; Talhaoui, M.Z. A sensitive dynamic mutual encryption system based on a new 1D chaotic map. *Opt. Lasers Eng.* **2021**, *139*, 106485. [CrossRef]
3. Wang, X.; Guan, N.; Yang, J. Image encryption algorithm with random scrambling based on one-dimensional logistic self-embedding chaotic map. *Chaos Solitons Fractals* **2021**, *150*, 111117. [CrossRef]
4. Zhou, Y.; Long, B.; Chen, C. A new 1D chaotic system for image encryption. *Signal Process.* **2014**, *97*, 172–182. [CrossRef]
5. Hu, G.; Li, B. Coupling chaotic system based on unit transform and its applications in image encryption. *Signal Process.* **2020**, *178*, 107790. [CrossRef]
6. Lan, R.; He, J.; Wang, S.; Gu, T.; Luo, X. Integrated Chaotic Systems for Image Encryption. *Signal Process.* **2018**, *147*, 133–145. [CrossRef]
7. Alghafis, A.; Firdousi, F.; Khan, M.; Batool, S.I.; Amin, M. An efficient image encryption scheme based on chaotic and Deoxyribonucleic acid sequencing. *Math. Comput. Simul.* **2020**, *177*, 441–466. [CrossRef]
8. Kang, X.; Guo, Z. A new color image encryption scheme based on DNA encoding and spatiotemporal chaotic system. *Signal Process. Image Commun.* **2019**, *80*, 115670.
9. Shahna, K.U.; Mohamed, A. A novel image encryption scheme using both pixel level and bit level permutation with chaotic map. *Appl. Soft Comput.* **2020**, *90*, 106162.
10. Hasheminejad, A.; Rostami, M.J. A novel bit level multiphase algorithm for image encryption based on PWLCM chaotic map. *Optik* **2019**, *184*, 205–213. [CrossRef]
11. Liu, M.; Ye, G.D. A new DNA coding and hyperchaotic system based asymmetric image encryption algorithm. *Math. Biosci. Eng.* **2021**, *18*, 3887–3906. [CrossRef] [PubMed]
12. Xw, A.; Sc, A.; Yz, B. A chaotic image encryption algorithm based on random dynamic mixing. *Opt. Laser Technol.* **2021**, *138*, 106837.

13. Wang, X.; Guan, N. Chaotic image encryption algorithm based on block theory and reversible mixed cellular automata—ScienceDirect. *Opt. Laser Technol.* **2020**, *132*, 106501. [CrossRef]
14. 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]
15. Zhao, F.; Liu, M.; Wang, K.; Zhang, H. Color image encryption via Hénon-zigzag map and chaotic restricted Boltzmann machine over Blockchain. *Opt. Laser Technol.* **2021**, *135*, 106610.
16. Xian, Y.; Wang, X.; Yan, X.; Li, Q.; Wang, X. Image Encryption Based on Chaotic Sub-Block Scrambling and Chaotic Digit Selection Diffusion. *Opt. Lasers Eng.* **2020**, *134*, 106202. [CrossRef]
17. Wang, X.; Si, R. A new chaotic image encryption scheme based on dynamic L-shaped scrambling and combined map diffusion. *Opt. Int. J. Light Electron Opt.* **2021**, *245*, 167658. [CrossRef]
18. Jain, A.; Rajpal, N. A robust image encryption algorithm resistant to attacks using DNA and chaotic logistic maps. *Multimed. Tools Appl.* **2016**, *75*, 5455–5472. [CrossRef]
19. Boyland, P.; De Carvalho, A.; Hall, T. Itineraries for inverse limits of tent maps: A backward view. *Topol. Its Appl.* **2017**, *232*, 1–12. [CrossRef]
20. Niu, Y.; Zhang, X.C. Image encryption algorithm based on variable step Joseph traversal and DNA dynamic coding. *J. Electron. Inf.* **2020**, *42*, 1383–1391.
21. Xingyuanwang, X.; Zhu, X.; Zhang, Y. An image encryption algorithm based on Josephus traversing and mixed chaotic map. *IEEE Access* **2018**, *6*, 23733–23746.
22. Tutueva, A.V.; Karimov, A.I.; Moysis, L.; Volos, C.; Butusov, D.N. Construction of one-way hash functions with increased key space using adaptive chaotic maps. *Chaos Solitons Fractals* **2020**, *141*, 110344. [CrossRef]
23. Wang, X.; Du, X. Pixel-level and bit-level image encryption method based on Logistic-Chebyshev dynamic coupled map lattices. *Chaos Solitons Fractals* **2022**, *155*, 111629. [CrossRef]
24. Wang, X.; Yang, J.; Guan, N. High-sensitivity image encryption algorithm with random cross diffusion based on dynamically random coupled map lattice model. *Chaos Solitons Fractals* **2021**, *143*, 110582. [CrossRef]
25. Li, M.; Guo, Y.; Huang, J.; Li, Y. Cryptanalysis of a chaotic image encryption scheme based on permutation-diffusion structure. *Signal Process. Image Commun.* **2018**, *62*, 164–172. [CrossRef]
26. Wang, X.; Zhao, H.; Feng, L.; Ye, X.; Zhang, H. High-sensitivity image encryption algorithm with random diffusion based on dynamic-coupled map lattices. *Opt. Lasers Eng.* **2019**, *122*, 225–238. [CrossRef]
27. Souyah, A.; Faraoun, K.M. An image encryption scheme combining chaos-memory cellular automata and weighted histogram. *Nonlinear Dyn.* **2016**, *86*, 1–15. [CrossRef]
28. Li, C.; Yang, X. An image encryption algorithm based on discrete fractional wavelet transform and quantum chaos. *Optik* **2022**, *260*, 169042.
29. Xu, S.; Wang, X.; Ye, X. A new fractional-order chaos system of Hopfield neural network and its application in image encryption. *Chaos Solitons Fractals* **2022**, *157*, 111889. [CrossRef]
30. Zhang, S.; Liu, L. A novel image encryption algorithm based on SPWLCM and DNA coding. *Math. Comput. Simul.* **2021**, *190*, 723–744. [CrossRef]
31. Li, Y.; Wang, C.; Chen, H. A hyper-chaos-based image encryption algorithm using pixel-level permutation and bit-level permutation. *Opt. Lasers Eng.* **2017**, *90*, 238–246. [CrossRef]
32. Belazi, A.; El-Latif, A.; Belghith, S. A novel image encryption scheme based on substitution-permutation network and chaos. *Signal Process.* **2016**, *128*, 155–170. [CrossRef]
33. Bezerra, J.; Molter, A.; Camargo, V. A new efficient permutation-diffusion encryption algorithm based on a chaotic map. *Chaos Solitons Fractals X* **2021**, *151*, 111235. [CrossRef]
34. Wang, X.; Li, Y. Chaotic image encryption algorithm based on hybrid multi-objective particle swarm optimization and DNA sequence. *Opt. Lasers Eng.* **2021**, *137*, 106393. [CrossRef]
35. Zhu, H.; Zhao, Y.; Song, Y. 2D Logistic-Modulated-Sine-Coupling-Logistic Chaotic Map for Image Encryption. *IEEE Access* **2019**, *7*, 14081–14098. [CrossRef]
36. Hossam, D. An Efficient Chaotic Image Cryptosystem Based on Simultaneous Permutation and Diffusion Operations. *IEEE Access* **2018**, *6*, 42227–42244.
37. Lopez, G.A.; Taufer, M.; Teller, P.J. Evaluation of IEEE 754 floating-point arithmetic compliance across a wide range of heterogeneous computers. In Proceedings of the 2007 Conference on Diversity in Computing, Orlando, FL, USA, 14–17 October 2007.


**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.