entropy

MDPI

Article

An Image Encryption Algorithm Using Cascade Chaotic Map and S-Box

Jiming Zheng [1,2,*] and Tianyu Bao [1]

1 College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
2 Key Laboratory of Intelligent Analysis and Decision on Complex Systems, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
* Correspondence: zhengjm@cqupt.edu.cn

Abstract: This paper proposed an image algorithm based on a cascaded chaotic system to improve the performance of the encryption algorithm. Firstly, this paper proposed an improved cascaded two-dimensional map 2D-Cosine-Logistic-Sine map (2D-CLSM). Cascade chaotic system offers good advantages in terms of key space, complexity and sensitivity to initial conditions. By using the control parameters and initial values associated with the plaintext, the system generates two chaotic sequences associated with the plaintext image. Then, an S-box construction method is proposed, and an encryption method is designed based on the S-box. Encryption is divided into bit-level encryption and pixel-level encryption, and a diffusion method was devised to improve security and efficiency in bit-level encryption. Performance analysis shows that the encryption algorithm has good security and is easily resistant to various attacks.

Keywords: image encryption; cascaded chaotic system; S-box; diffusion

check for updates

Citation: Zheng, J.; Bao, T. An Image Encryption Algorithm Using Cascade Chaotic Map and S-Box. *Entropy* **2022**, *24*, 1827. https://doi.org/10.3390/e24121827

Academic Editors: Xiaowei Li, Jian-Zhong Li and Yu Zhao

Received: 8 November 2022
Accepted: 12 December 2022
Published: 14 December 2022

1. Introduction

After decades of development, digital information has grown extensively in capacity, and various electronic devices are changing gradually, accompanied by a large amount of data for transmission and communication [1], which brings various kinds of security risks in this background, especially image data and video. While traditional encryption techniques such as DES and AES have achieved good results in text encryption, they are not ideal for encrypting large amounts of modern image data [2,3].

Due to chaotic sensitivity to initial state and control value, good pseudo-randomness, ergodicity and unpredictable trajectory [4], the combination of chaos and encryption technology has produced many different chaos and applications: The chaotic system that has the property crosses a pre-define cylinder repeatedly and proposes the XOR approach for diffusion encryption of images. Despite receiving good encryption performance, this scheme lacks the appropriate scrambling operations, and the encryption scheme is independent of the plaintext image, making it difficult to resist chosen-plaintext attacks [5]. The encryption algorithm uses 3Dchua's system with a combination of DWT transform and compressed sensing [6], and its experiments have shown its good encryption effect. The method NCCS, which generates new maps by combining methods, is able to overcome the shortcomings of a traditional one-dimensional chaotic map and gives a bit-level confusion and diffusion scheme while incorporating plaintexts in the keys used, which are shown to be well secured. However, when operating on the bit level, a large number of chaotic sequences are required, which creates some time consumption [7]. In addition, a cascaded chaotic system, as a form of chaos, is also a research hotspot; different chaos and encryption schemes are proposed: Zheng et al. [8] used an encryption scheme based on cascaded chaos, which generates new chaos through a cascaded one-dimensional chaotic map, so as to improve the performance of chaos and be used in encryption. Encryption methods based on DNA and dual chaotic systems are also proposed, but the overall encryption is not associated with the plaintext and lacks an efficient scrambling process. Lan et al. [9] proposed a composite integrated chaotic system, which integrated cascading, nonlinear combination and other operations, and then proposed an encryption scheme ICST with bit-level substitution and transform, but the overall lack of an effective diffusion process. Wang et al. [10] proposed a cascaded chaos model for CMCS to generate new chaos maps by using two-dimensional chaos mappings and one-dimensional chaos mappings, and demonstrated the good performance of the new chaos map through relevant experiments, an encryption method for chunking images and performing different operations on different blocks is also given, including bit-level transformations, shuffling algorithms, DNA encoding and V-shaped diffusion, etc. At the same time, in order to improve the security of the algorithm, people started to introduce some other techniques into chaotic encryption, and S-boxes is one of them.

Two types of cryptosystems can be classified: stream ciphers, which are converted bit-by-bit, and packet ciphers, which convert n-bit inputs into m-bit outputs. At the core of this conversion is the static S-box, which gives the cryptosystem the obfuscation properties described by Shannon [11]. An S-box is considered to be a well-performing S-box if it satisfies some of the following conditions [12]: bijection, nonlinearity, strict avalanche criterion (SAC) and output bit independence criterion (BIC). According to the way the S-box is generated, we can also classify it into static S-boxes and dynamic S-boxes. The security of the ciphertext is not guaranteed [13]. Dynamic S-boxes are based on key generation, and different keys can generate different S-boxes; all of these can improve the security of the encryption system [14]. Currently, S-boxes are the only nonlinear component of many packet ciphers, and the performance of S-boxes largely determines the security strength of encryption algorithms. In [15], Zhou et al. proposed a randomized approach to S-box generation by using DNA encoding and showed that the S-boxes generated by this method are resistant to different types of attacks. In addition, many researchers have applied knowledge from other fields to the generation of S-boxes, but some of these methods are still not efficient enough to meet today's cryptographic efficiency requirements, so most of them cannot be practically applied in the encryption process. With the development of chaotic applications, many research results have been achieved in chaos-based S-box generation. Wang et al. [16] used LDCML to construct new S-boxes by dividing the interval of [0, 1] into 256 equal intervals, iterating LDCML to generate chaotic sequences, and generating non-repeating numbers between 0 and 255 according to the interval in which the generated values fall, and verifying that this method can generate S-boxes that satisfy the S-box criteria. Belazi et al. [17] used a map method to generate values between 0 and 255 and randomly place the mapped values through the sinusoidal map. In [18], Beg et al. finished S-box construction by expanding the remaining random values, generating chaotic sequences by iterating a chaotic map and expanding the remainder to continuously generate non-repeating values between 0 and 255 to add to the array.

As the security strength of the entire encryption algorithm is determined by the cryptographic strength of the S-box, the researcher has proposed many methods to enhance the S-box's performance, such as using the high-dimensional chaotic system, and multiple chaotic systems to generate S-boxes have become solutions. Liu et al. [19] used a high-performance S-box construction using 3D chaotic systems and gave the corresponding encryption algorithm, iterating the high-dimensional chaotic systems to complete the S-box construction; however, the high-dimensional chaotic system then takes much time in computation, and although the security is improved, it becomes less efficient in terms of efficiency. Zheng et al. [20] proposed a multi-chaotic method for constructing dynamic S-boxes with improved efficiency and security and gave a corresponding encryption algorithm, which was shown to be feasible in experimental results. Özkaynak [21] proposed to use two S-boxes from the Henon map and Chen system, which are chosen at random to increase the encryption's security. Wang et al. proposed to generate three S-boxes by the 3D chaotic map and perform one round of permutation for encryption [22], but the generation process generates a large number of useless chaotic sequences, which perform slightly worse in terms of real-time performance, and the key of the encryption algorithm is fixed, which means that the S-boxes generated by each encryption iteration are the same. Wang et al. proposed a cascade chaotic map and used it to generate S-box [23], followed by a diffusion operation after substituting with S-boxes, and this algorithm has good advantages in terms of security and complexity.

Based on the above analysis, a cascade chaotic map 2D-Cosine-Logistic-Sine map (2D-CLSM) is proposed in this paper. The initial values and parameters of the chaotic map are combined with the original image to resist known plaintext attacks. Then, an S-box construction method is proposed, and an S-box-based encryption method is designed. The encryption method is divided into four stages: key generation, S-box generation, bit-level encryption and pixel-level encryption. In bit-level encryption, a bit-level operation is performed on the plaintext pixel values, converting the original decimal plaintext image pixels to eight-bit binary, then permuting the lower four bits and using a proposed new diagonal diffusion method for the higher four bits. In the pixel-level encryption part, one diffusion is completed by a three-number XOR (by using the chaotic sequence value, the current pixel value to be encrypted and the previously encrypted pixel value), and the resulting value calculates the row and column index of the S-box for pixel value replacement. Through experimental analysis, the encryption algorithm in this paper has good security performance.

The remainder of this essay is structured as follows: Section 2 introduces the chaotic map; Section 3 describes the design of the encryption algorithm; Section 4 gives the simulation experiments and analysis of the results of the method titled in this paper; Finally, Section 5 gives conclusion remarks and further research work.

2. Introduction of Chaotic Map

In nonlinear dynamic systems, chaos is a stochastic process that is frequently employed in cryptography research [24]. One-dimensional Logistic and Sine map is classical chaotic map with a simple structure. They are defined as follows.

$$t_{n+1} = 4\mu t_n(1 - t_n) \tag{1}$$

$$t_{n+1} = ksin(\pi t_n) \tag{2}$$

with the control parameters $\mu, k \in [0, 1]$.

The chaotic behavior of a chaotic system can be measured by the bifurcation diagram and the Lyapunov exponent. Figure 1a,b shows that the chaotic range of the one-dimensional Logistic chaotic map is restricted. From Figure 1b, we are able to find that μ is in the range of [0.89, 1], and its Lyapunov exponent is greater than 0. Figure 1c,d show that the chaotic range of the one-dimensional Sine map is similarly constrained, with k in Figure 1d in the range [0.87, 1] before its Lyapunov exponent is greater than 0 to be chaotic. Hence the classical Logistic and Sine maps are limited in terms of key space and are not resistant to brute force cracking.

2.1. Cascade Chaotic Map

A cascade map is a form of a chaotic map; the use of cascades can effectively improve the performance of chaotic systems. In order to solve the problem of small chaotic intervals and uneven distribution, we designed a two-dimensional cascade chaotic map, which has a more complex chaotic behavior than one-dimensional chaos and a faster iteration speed than two-dimensional chaos. The cascade system is shown in Figure 2, where $f_1(x_n), f_2(x_n)$ are two different subsystems.

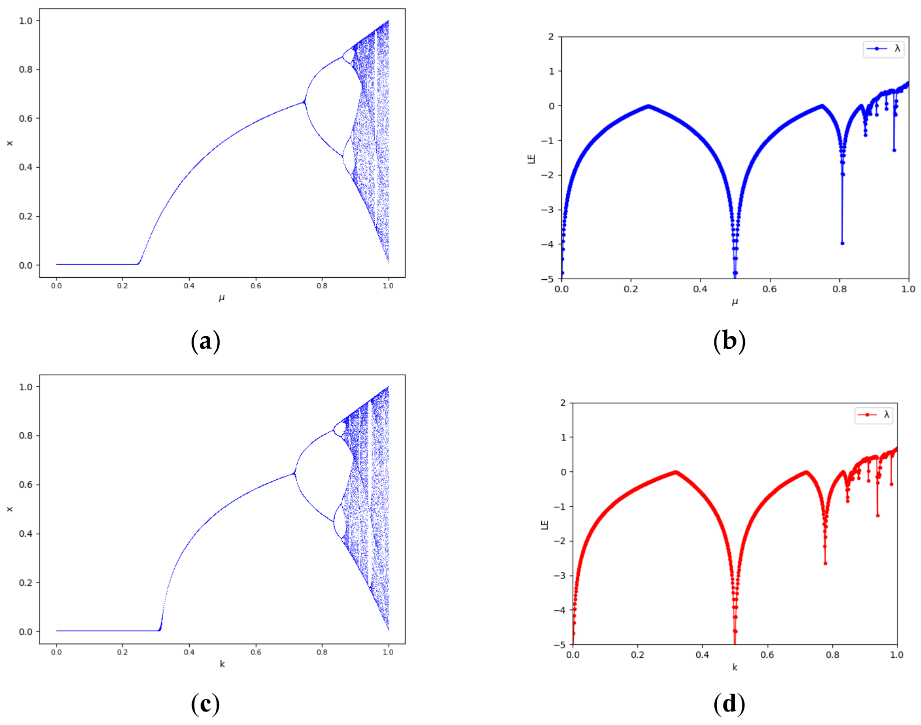


Figure 1. Chaotic bifurcation diagrams and Lyapunov exponents: (**a**) bifurcation diagram of Logistic map; (**b**) Lyapunov exponent of Logistic map; (**c**) bifurcation diagram of Sine map; (**d**) Lyapunov exponent of Sine map.

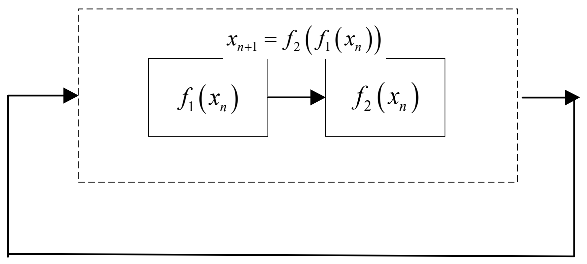


Figure 2. Cascade system.

The essence of cascading is that the output of a certain initial value after the iteration of system 1 is taken as the iterative output of system 2, and the iterative output of system 2 is taken as the iterative output of system 1, thus forming a circular iteration between two subsystems. Chaotic systems are generated by the cascade method, where the Lyapunov value of the system is the sum of the Lyapunov exponents of the cascaded subsystems, and the sequential trajectory of the system output deviates sharply as the number of iterations increases [25].

Using cosine functions is proposed and demonstrated in [26], where existing chaotic maps are cascaded by using cosine functions to improve their chaotic performance and extend the chaotic space, the expression as shown in Equation (3):

$$x_{i+1} = G\left(2^{h+F(x_i)}\right) \tag{3}$$

where $G(\cdot)$ is the subsystem 2 and $F(\cdot)$ is the subsystem 1. In order to extend it to two dimensions, the general form expression is shown in Equation (4):

$$S(x_i, y_i) = G\left(2^{h+F(x_i,y_i)}\right) \tag{4}$$

where x_i, y_i are two variables and G is the cosine that it can be represented as Equation (5):

$$\begin{cases} x_{i+1} = G\left(2^{h+A(x_i,y_i)}\right) = cos\left(2^{h+A(x_i,y_i)}\right) \\ y_{i+1} = G\left(2^{h+B(x_i,y_i)}\right) = cos\left(2^{h+B(x_i,y_i)}\right) \end{cases} \tag{5}$$

In Equation (3), to increase the chaotic complexity of the cosine function-based system, a new control parameter h is introduced as part of the exponential function. When the angle of the cosine function is large enough that even small differences can lead to large output differences such as $cos\left(2^{10}\right) = 0.0001557636$ and $cos\left(2^{10.00001}\right) = 0.987355203504$, also reducing the complexity of the calculation to make $h \in [10, 24]$.

$f_1(x_i) = 4k_1x_i(1-x_i)$ and $f_2(x_i) = k_2\sin(\pi x_i)$ are Logistic map and Sine map, respectively, and $A(x_i, y_i) = f_1(x_i) + f_2(y_i)$, $B(x_i, y_i) = f_1(y_i) + f_2(x_i)$. We made them as subsystem 1 of Equation (4) and used it as the argument of subsystem 2, and then we derived Equation (6):

$$\begin{cases} x_{i+1} = cos\left(2^{h+f_1(x_i)+f_2(y_i)}\right) = cos\left(2^{h+(4k_1x_i(1-x_i)+k_2\sin(\pi y_i))}\right) \\ y_{i+1} = cos\left(2^{h+f_1(y_i)+f_2(x_{i+1})}\right) = cos\left(2^{h+(4k_1y_i(1-y_i)+k_2\sin(\pi x_{i+1}))}\right) \end{cases} \tag{6}$$

Further, controlling the value of the parameter θ enables Logistic and Sine to be in chaos, which, according to Figure 1, makes $k_1 = k_2 = 1 - \frac{1}{9}\theta$ [27], and when using mod1 to control the iteration value between (0, 1), the chaotic map 2D-CLSM expression is shown in Equation (7):

$$\begin{cases} x_{i+1} = \left[cos\left(2^{h+(1-\frac{1}{9}\theta)(4x_i(1-x_i)+\sin(\pi y_i))}\right)\right]\text{mod}1 \\ y_{i+1} = \left[cos\left(2^{h+(1-\frac{1}{9}\theta)(4y_i(1-y_i)+\sin(\pi x_{i+1}))}\right)\right]\text{mod}1 \end{cases} \tag{7}$$

where $x_i \in (0, 1)$, $y_i \in (0, 1)$ are the control parameters $h \in [10, 24]$, $\theta \in [0, 1]$.

2.2. Performance Evaluation

In order to analyze the 2D-CLSM's performance, we compared it with another existing 2D chaotic map for image encryption, i.e., the 2D Logistic-Sine-Coupling Map (2D-LSCM) [28].

$$\begin{cases} x_{i+1} = \sin(\pi(\theta \cdot 4x_i(1-x_i) + (1-\theta)\sin(\pi y_i))) \\ y_{i+1} = \sin(\pi(\theta \cdot 4y_i(1-y_i) + (1-\theta)\sin(\pi x_{i+1}))) \end{cases} \tag{8}$$

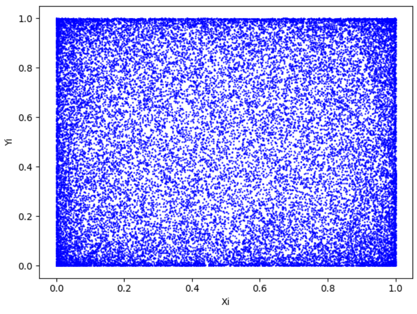
2.2.1. Chaotic Trajectory

The trajectory of a 2D-CLSM shows how motion increases over time from a specific initial state. In the case of periodic motion, the trajectory would be a closed curve, whereas the trajectory of chaotic behavior would theoretically never close or repeat. Therefore, chaotic trajectories usually occupy a part of the phase space and can reflect the randomness of the chaotic system output. If a chaotic trajectory can occupy a larger portion of the phase space, the chaotic system has a better stochastic output.

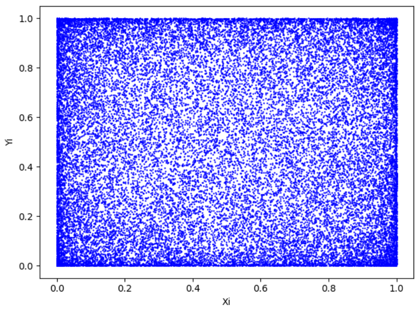
From Figure 3a–f, we can see that the 2D-CLSM is able to occupy the full phase plane for all trajectories within the parameter range, and the ability to occupy the full phase plane with both different θ and h indicates that the improved chaotic system has a better random output. On the contrary, 2D-LSCM is influenced by control parameters, which are not able to occupy full space.

2.2.2. Bifurcation Diagram

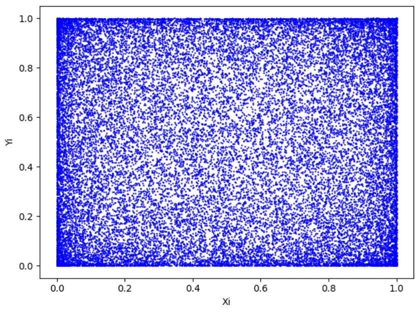
We set initial value as $x_0 = 0.4, y_0 = 0.3$ and the 2D-CLSM $h = 15$. As shown in Figure 4a–d, 2D-CLSM is in a chaotic state in the whole parameter domain, and it is more uniform. On the contrary, the 2D-LSCM does not occupy the entire plane, where the control parameters are at [0.3, 0.43], and values between [0, 0.1] cannot be generated iteratively.



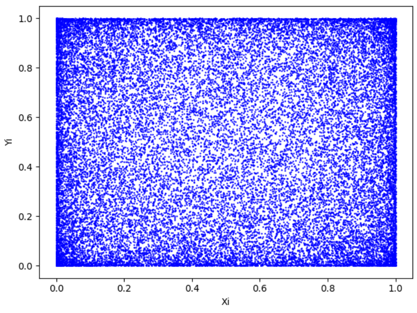
(**a**)



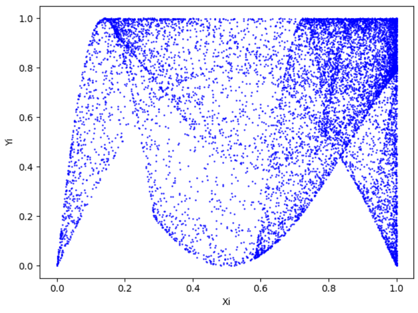
(**b**)



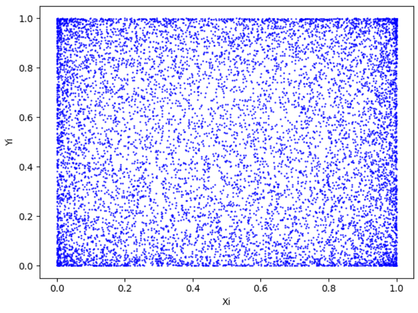
(**c**)



(**d**)

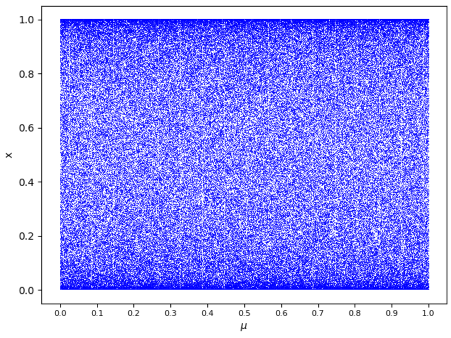


(**e**)

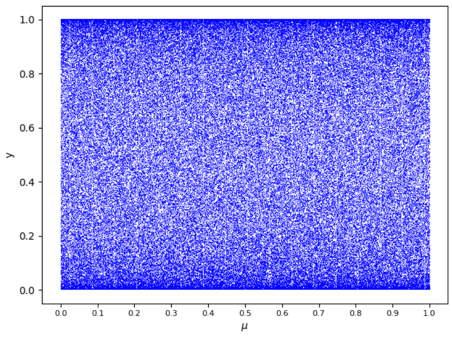


(**f**)

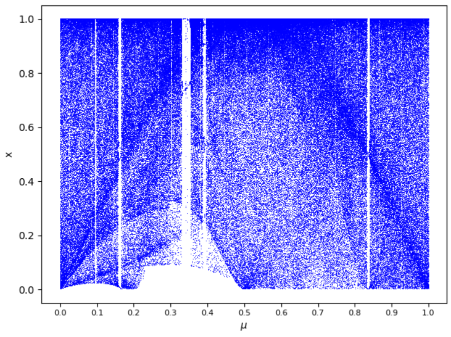
Figure 3. The trajectory of chaotic systems: (**a**) the trajectory of 2D-CLSM with $\theta = 0.71, h = 15$; (**b**) the trajectory of 2D-CLSM with $\theta = 0.3, h = 15$; (**c**) the trajectory of 2D-CLSM with $\theta = 0.71, h = 20$; (**d**) the trajectory of 2D-CLSM with $\theta = 0.3, h = 20$; (**e**) the trajectory of 2D-LSCM with $\theta = 0.3$; (**f**) 2D-LSCM with $\theta = 0.71$.



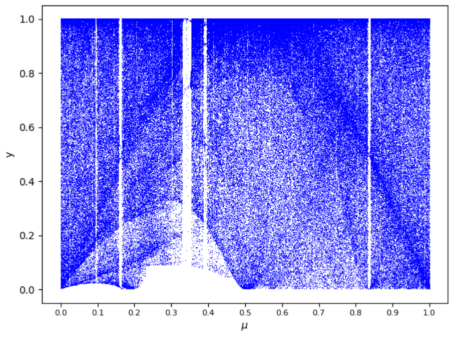
(**a**)



(**b**)



(**c**)



(**d**)

Figure 4. Bifurcation diagram of chaotic system: (**a**) bifurcation diagram of 2D-CLSM $\theta - x$; (**b**) bifurcation diagram of 2D-CLSM $\theta - y$; (**c**) bifurcation diagram of 2D-LSCM $\theta - x$; (**d**) bifurcation diagram of 2D-LSCM $\theta - y$.

2.2.3. Lyapunov Exponent

The LE (Lyapunov exponent) describes the sensitivity of a chaotic mapping to initial values. In general, a chaotic map is in a chaotic state when $\lambda > 0$ indicates that two adjacent phase points are about to separate and the chaotic map is in a chaotic state. For a two-dimensional map, the system of difference equations is assumed to be:

$$\begin{cases} x_{i+1} = f_1(x_i, y_i) \\ y_{i+1} = f_2(x_i, y_i) \end{cases} \tag{9}$$

Its Jacobian matrix at the point $x^{(i)} = (x_i, y_i)$ is as follows:

$$f'\left(x^{(i)}\right) = \left.\begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix}\right|_{(x_i, y_i)} \tag{10}$$

Let $J_i = f'\left(x^{(0)}\right) f'\left(x^{(1)}\right) \cdots f'\left(x^{(i-1)}\right)$, then the eigenvalue of J_i may be expressed as $\lambda_1^{(i)}, \lambda_2^{(i)}$. The Lyapunov exponents of system (9) can be expressed as Equation (11):

$$\lambda_k = \lim_{i \to \infty} \frac{1}{i} ln \left| \lambda_k^{(i)} \right| \tag{11}$$

The larger the value of λ, the faster the separation of point phase points in the phase space and the greater the sensitivity of chaos to initial values. Figure 5 shows the Lyapunov exponent of 2D-CLSM and 2D-LSCM; it is clear that the 2D-CLSM has a better chaotic behavior than 2D-LSCM.

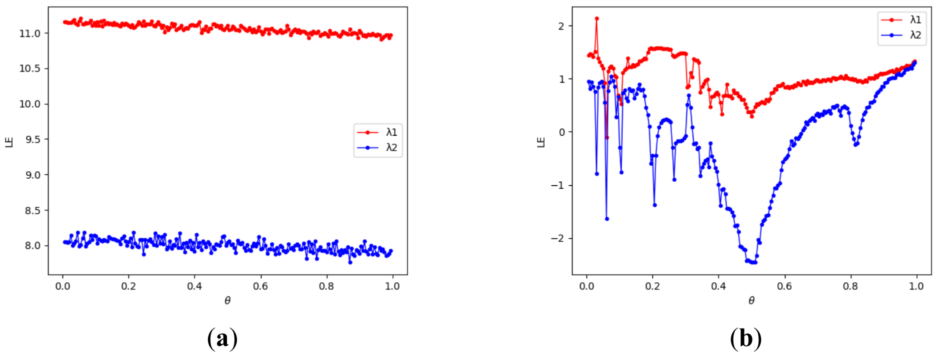


Figure 5. Lyapunov exponents for chaotic systems: (**a**) Lyapunov exponent of 2D-CLSM; (**b**) Lyapunov exponent of 2D-LSCM.

2.2.4. NIST Test

In this paper, 15 NIST tests were used to test the randomness of the generated sequences, and 15 correspond to p-values to show the test result. The sequence is considered random when the p-value is over 0.01. Table 1 contains the NIST test results for the 2D-LSCM map and the 2D-CLSM map. The NIST test results for the 2D-CLSM map are all "Success", as shown in Table 1, and 11 of the 2D-CLSM test results are higher than 2D-LSCM, demonstrating 2D-CLSM map can generate pseudo-random sequences with good random performance.

Table 1. NIST test results.

Serial Number	Test Items	2D-CLSM		2D-LSCM	
		p Value	Test Results	p Value	Test Results
1	Frequency	0.1422	Success	0.0767	Success
2	Block Frequency	0.7165	Success	0.9936	Success

Table 1. *Cont.*

Serial Number	Test Items	2D-CLSM		2D-LSCM	
		p Value	Test Results	p Value	Test Results
3	Cumulative Sums	0.2472	Success	0.0692	Success
4	Runs	0.8561	Success	0.7405	Success
5	Longest Run of Ones	0.7310	Success	0.4477	Success
6	Rank	0.1691	Success	0.1514	Success
7	Discrete Fourier Transform	0.8330	Success	0.1766	Success
8	Nonperiodic Template Matchings	0.6254	Success	0.4721	Success
9	Overlapping Template Matchings	0.6886	Success	0.9365	Success
10	Universal	0.9992	Success	0.9583	Success
11	Approximate Entropy	0.9309	Success	0.7040	Success
12	Random Excursions	0.1319	Success	0.2175	Success
13	Random Excursion Variant	0.1025	Success	0.4166	Success
14	Serial	0.9068	Success	0.1638	Success
15	Linear Complexity	0.9250	Success	0.5041	Success

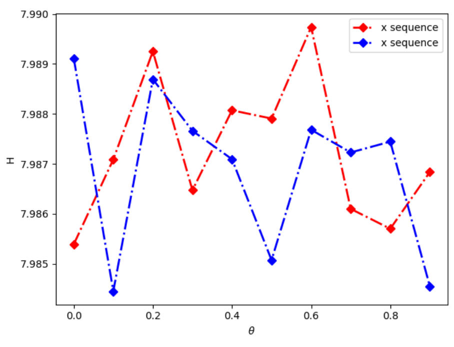
2.2.5. Information Entropy

In information theory, information entropy is used to quantify the uncertainty of the information content. It can be used to evaluate how random a set of data is. We transformed the chaotic sequence obtained by iteration into values between 0 and 255 and obtain the information entropy according to Equation (12):

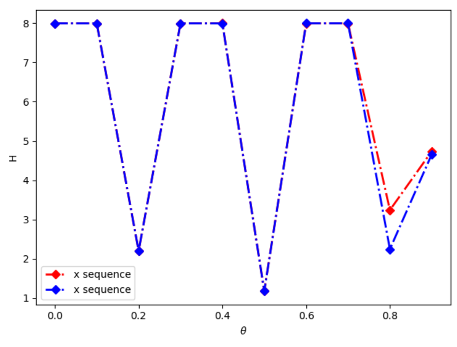
$$H(X) = -\sum_{i=0}^{2^{N-1}} Pr(x_i)\log_2 Pr(x_i) \tag{12}$$

where X is a data sequence, x_i is the ith possible value in X and $Pr(x_i)$ is the probability of x_i. A bigger information entropy value means better randomness; for a set with 256 states, its maximum expected value is $\log_2{}^{256} = 8$.

Better randomness is correlated with larger information entropy values, and Figure 6 shows the information entropy of the output sequences generated by 2D-CLSM for different parameter settings. From Figure 6, the mean information entropy value of 2D-CLSM is bigger than 2D-LSCM, and it is close to 8, which means good randomness.



(**a**)



(**b**)

Figure 6. The information entropy of the sequences: (**a**) information entropy of 2D-CLSM; (**b**) information entropy of 2D-LSCM.

3. New Encryption Algorithm Design

There are four main stages in our encryption algorithm: key generation, S-box generation, bit-level encryption and pixel-level encryption. The size of the plaintext image P is $M \times N$, and $t, \mu, \theta, x, y, \Delta m$ are the initial keys. The overall process is shown in Figure 7.

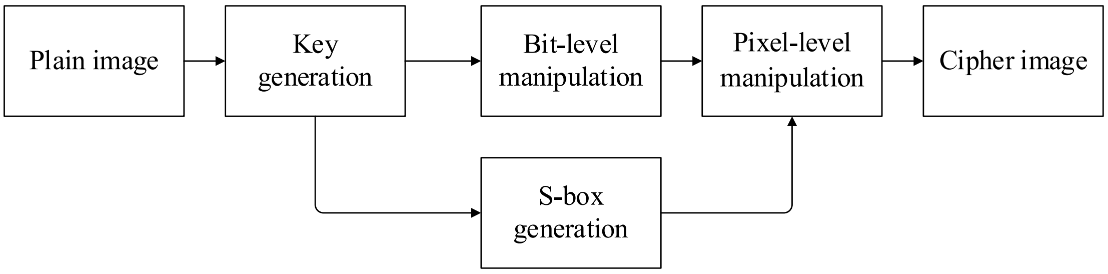


Figure 7. Overall process.

Two different chaotic maps are used in our algorithm: Logistic and 2D-CLSM, which are used in different stages of encryption. In the S-box generation stage, a Logistic map is used to generate the chaotic sequence needed to generate the S-box; a 2D-CLSM map is used in the bit-level operation and pixel-level operation phases. By using the different chaotic maps in different stages to expand the key space, a more complex encryption algorithm is generated.

3.1. Keys Generation

If the key stream used for encryption is only related to the key and not related to the plaintext image, the designed algorithm is not safe for chosen/known plaintext attack. In order to ensure the security of encryption, the encryption key is generated from the initial key and the plaintext image.

Initial encryption keys contain six decimal numbers, $t, \mu, \theta, x, y, \Delta m$. Six decimal encryption keys, $t_0, \hat{\mu}, \hat{\theta}, x_0, y_0, \hat{h}$, are generated by combining the plaintext images, which are used as initial values and control parameters of the chaotic system. Using $\Delta m > 0$ as a scrambling value prevents an attacker from attacking the key with a black or white image. The encryption key is generated by Algorithm 1, where $t_0, \hat{\mu}$ is used as the initial value and control parameter of the Logistic map; $x_0, y_0, \hat{h}, \hat{\theta}$ is set as the initial value, control parameter and variables h of 2D-CLSM.

Algorithm 1 Generation of the encryption keys

```
Input: Plain image $P$, initial keys $t, \mu, \theta, x, y, \Delta m$
Output: The encryption keys $t_0, \hat{\mu}, \hat{\theta}, x_0, y_0, \hat{h}$
    1: Read the size: $[M, N]$ = size $(P)$
    2: Obtain the sum of all pixels and add scramble number
    $sum = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} P(i,j) + \Delta m$
    3: Calculate the mean of the all pixels $\overline{m} = sum/(M \times N)$
    4: $t_0 = cos\left(\overline{m} \times 2^{t+15}\right)\mathrm{mod}1$
    5: $\hat{\mu} = 1 - cos\left(\overline{m} \times 2^{\mu+15}\right)\mathrm{mod}0.1$
    6: $\hat{\theta} = cos\left(\overline{m} \times 2^{\theta+20}\right)\mathrm{mod}1$
    7: $x_0 = cos\left(\overline{m} \times 2^{x+20}\right)\mathrm{mod}1$
    8: $y_0 = cos\left(\overline{m} \times 2^{y+20}\right)\mathrm{mod}1$
    9: $\hat{h} = \left(cos\left(\overline{m} \times 2^{x_0 \times y_0 + 20}\right)\mathrm{mod}1\right) \times 14 + 10$
```

3.2. S-Box Generation

3.2.1. S-Box Generation Algorithm

In the construction method of a chaotic S-box, there is the problem of generating useless chaotic sequences that affect efficiency. Wang et al. proposed in [15] to use a threedimensional chaotic map to iterate chaotic sequences and expand the modulo operation to generate values between 0 and 255 to generate S-boxes, and the algorithm only ends when all 256 values are generated, this method generates a large number of useless chaotic sequences, and if a value cannot be generated all the time, the efficiency of the algorithm is affected, and the real-time performance becomes poor. In some methods, there are immobile points in the generated S-boxes [29], which can become attackable points. Table 2 gives the generation times of existing S-box generation methods, the presence of fixed points and the number of fixed points before and after the Fisher–Yates shuffling algorithm is applied to the S-box.

Table 2. Generation time and fix point.

S-Box	Generate Time	Fixed Point	After Fisher–Yates
Ref. [17]	0.9415	2, 17, C7, CB	None
Ref. [21]	0.0487	0D, 33, 77, 95	None
Ref. [22]	0.7593	None	None

The Fisher–Yates shuffling algorithm is an algorithm proposed by R. Fisher and F. Yates for generating random permutations of finite linear arrays [30]. The most important feature of this approach is that it generates an unbiased result so that the probability of each value being at any position is equally likely, essentially generating a finite set of random permutations. Thus, by using the Fisher–Yates random shuffle algorithm, we are able to make the encryption scheme more complex and secure by enabling us to quickly generate different S-boxes and reduce the presence of fixed points during each encryption. The steps are as follows:

Step 1: Obtain the length m of the sequence P that needs to be shuffled;
Step 2: Generate a random number n with a value between $[0, m-1]$;
Step 3: Shuffle the values of the two positions according to n and m, then exchange the values of $P(n)$ and $P(m)$;
Step 4: Subtract 1 from m to obtain the new position;
Step 5: Repeat step1~step4 until $m = 1$.

Algorithm 2 describes the generation process of the S-box. $t_0, \hat{\mu}$, generated by algorithm 1, is taken as the initial value and control parameter of Logistic, iteratively generating a chaotic sequence Q with a length of 512 bits, and then divided into two sequences $Q1, Q2$ with a length of 256 bits, respectively, calculating and obtaining the ascending index $q1 \in [0, 255]$, $q2 \in [0, 255]$ of $Q1$ and $Q2$. Then $q2$ is taken as the random number sequence of the shuffling algorithm, shuffling $q1$ and finally, $q1$ is converted into a 16 bits $\times$ 16 matrix to obtain the S-box s.

Algorithm 2 Generation of S-box

```
Input: encryption keys $t_0, \hat{\mu}$
Output: S-box $s$
    1: for $i$ from 1 to 1512:
           Substituting $t_0, \hat{\mu}$ into Equation (1)
           if $i$ >1000:
               obtain the 512-length chaotic sequences $Q$
    2: Divide $Q$ into two subsequences of length 256 $Q1$, $Q2$; obtain the ascending
       sort index of the $Q1$, $Q2$; and assign it to $q1 = \text{arg}sort(Q1)$ and $q2 = \text{arg}sort(Q2)$
    3: Read the size: $m = size(q1)$
    4: while $m > 1$:
           Obtain a random number $q2(m)$
           Swap the value of $q1(q2(m)), q1(m)$
           Set $m = m - 1$
    5: If $m = 1$ transform $q1$ into 16 matrix and assign it to $s$
    6: Obtain S-box $s$
```

3.2.2. Performance Test of the Proposed S-Box

In order to verify the performance and strength of the generated S-boxes, we used the general criteria for S-box performance evaluation to be able to test them [31]. In this thesis, the nonlinearity of the S-box, the strict avalanche criterion SAC, the output bit independence criterion BIC and the differential approximate probability DP are verified, respectively. Table 3 displays the S-box matrix that we produced.

Table 3. The generated S-box by proposed algorithm.

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

(I) Nonlinearity In the process of encryption, if the given S-box makes a linear mapping between the input (plaintext) and the output (ciphertext) [32], then a decipherer can easily deduce and break the ciphertext when the cryptographic strength of the S-box is very small, but if the S-box can map the input to the output in a nonlinear way, then it is considered a reliable S-box that can protect the plaintext data and can help us resist the attacks of linear cryptanalysis, and we can calculate the nonlinear value of the 8-bit Boolean function S by using Equation (13).

$$NL_f = 128 - \frac{1}{2}\max_{\omega \in GF(2^n)} \left| WH_f(\omega) \right| \tag{13}$$

where NL_f is the 8-bit Boolean function, $WH_f(\omega)$ is the Walsh–Hadamard transform of the eight-bit Boolean function S. The values of the S-box nonlinearity obtained according to the above are shown in Table 4, where the maximum value is 110, the minimum value is 106 and the average value is 107.5

Table 4. S-box nonlinear values.

	S1	S2	S3	S4	S5	S6	S7	S8
NL(s)	108	106	108	106	108	108	110	106

(II) Strict Avalanche Criterion (SAC) Webster and Tavares used the strict avalanche criterion as an important feature of the performance of an S-box [33]. The strict avalanche criterion ensures that if one bit is changed in the input, it causes at least 50% of the output to change, and an S-box is considered to be a strong S-box if the value of SAC is approximately equal to 0.5. We propose that the S box has a SAC value that satisfies the strict avalanche criterion. The SAC dependency matrix of our generated S-boxes is shown in Table 5. The table shows the average SAC of the generated S-boxes is 0.4996, which is very close to the desired value of 0.5, and shows that the generated S-boxes satisfy the SAC criteria.

Table 5. Generating S-box SAC values.

0.4196	0.4235	0.4196	0.5647	0.4823	0.5137	0.5450	0.5294
0.4549	0.5019	0.4509	0.5019	0.5490	0.4705	0.4352	0.5176
0.5960	0.4862	0.4980	0.5137	0.5607	0.5333	0.5607	0.5176
0.4980	0.5137	0.5490	0.4980	0.4392	0.4862	0.5450	0.5490
0.5019	0.5294	0.5294	0.4980	0.4392	0.4862	0.5450	0.5490
0.5019	0.4509	0.5647	0.5176	0.5137	0.4980	0.5137	0.4392
0.4666	0.5333	0.4823	0.4235	0.4549	0.5450	0.5294	0.4235
0.4705	0.5333	0.4823	0.4549	0.5490	0.5294	0.5019	0.4980

(III) Output Bits Independence Criterion (BIC) This criterion was introduced by Webster et al. as one of the important properties of S-box evaluation, a property that ensures that there is no dependence on the change of any two output bits when a single input bit is changed, a property that makes any Boolean function must be independent and highly nonlinear. For two output bit Boolean functions f_i and f_j in an S-box, if $f_i \oplus f_j$ is highly nonlinear and satisfies the SAC as close as possible, then it is guaranteed that when one input bit is inverted, the correlation coefficient of each output bit is close to 0, i.e., the BIC is satisfied.

Through experimental tests, the average BIC nonlinearity value of the proposed S-box is 104.5, and the BIC-SAC test result is 0.5009, which satisfies the BIC criterion.

(IV) Difference Approximation Probability (DP) differential cryptanalysis, introduced by Biham and Shamir [34], is able to obtain the input differential from the output differential while being able to attempt to obtain from it modifications to the plaintext and changes to the ciphertext data, combined with the difference between the two changes an attacker is able to use the resulting small differences to identify complete or partial plaintexts and keys, and in the process of designing the S-box, there is a need to minimize both changes The difference between the two needs to be minimized in the design of the S-box. The designer calculates the difference by differential uniformity, which is checked right by the differential approximation probability, as shown in Equation (14):

$$DP(f) = \max_{\Delta x \neq 0, \Delta y} \left(\frac{\#\{x \in GF(2^n) | f(x) \oplus f(x + \Delta x) = \Delta y\}}{2^n} \right) \tag{14}$$

where Δx and Δy are the input difference and output difference, and DP denotes the maximum probability that the output of each given difference Δx is equal to Δy. The maximum DP value of our proposed generated S-box is 10, indicating that our S-box can resist differential.

In comparison with the other four methods, the results show in Table 6 that the S-box nonlinearity produced by the algorithm in this paper is higher than the other four solutions. The SAC value of S box 0.4996 is closest to the ideal value of 0.5 in five methods, and the BIC value also meets the test requirements. The BIC of the S-box also meets the test requirements based on the DP value of the test and has a good ability to resist the differential password attack based on the DP value of the test. In terms of S-box generation time, the algorithm in this paper has a good advantage. Although our nonlinear value is still some distance from the ideal value, we believe that the performance of the generated S-box needs to meet the performance requirements, and the efficiency and space cost of S-box generation also need to be considered.

Table 6. S-box performance test results.

S-Box	Nonlinearity			SAC	BIC-SAC	BIC-Nonlinearity	DP	Generate Time
	Min	Max	Avg					
ours	106	110	107.5	0.4996	0.5009	104	10	0.0066
Ref. [18]	100	106	104	0.4988	0.5006	104	10	0.3071
Ref. [19]	102	108	104	0.4988	0.5052	104	10	0.0091
Ref. [20]	106	108	106	0.4916	0.5058	104.14	10	0.0160
Ref. [22]	99	106	103.5	0.5065	0.5013	103.357	12	0.7593

3.3. Bit-Level Encryption

Traditional image encryption generally falls into one of three categories: permutation-only, diffusion-only, or combined forms. Due to its simpler computational complexity, the permutation-only type of these is more efficient, but the security is not very strong. Due to the substantial computational load required for real arithmetic operations, the diffusion type is a time-consuming process. The pixel values and random numbers are used to perform an XOR operation, and the calculated numbers are used to determine the ranks of the S-boxes and substitute the original pixels with the values in the S-boxes, but the overall encryption efficiency is low due to the lack of permutation and diffusion operations on the plaintext image and the low efficiency of S-box generation [15].

In this stage, by separating the eight-bit pixel values into upper four bits and lower four bits, different operations are performed on the two parts, respectively, and a new diagonal diffusion method for irregular matrices is proposed to be applied to the high four-bit matrix. The bit-level encryption process is shown in Figure 8.

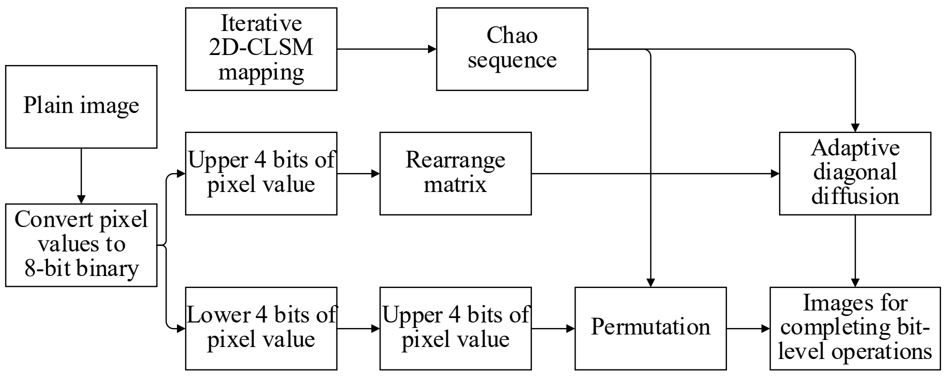

Iterative 2D-CLSM mapping
Chao sequence
Plain image
Upper 4 bits of pixel value
Rearrange matrix
Adaptive diagonal diffusion
Convert pixel values to 8-bit binary
Lower 4 bits of pixel value
Upper 4 bits of pixel value
Permutation
Images for completing bit-level operations

Figure 8. Bit-level encryption process.

3.3.1. Pixel Value Split

We converted the original pixel value of the plaintext image into a binary representation and extracted it into two parts for different operations. The upper four bits P1 of the pixel value and the lower four bits P2 of the pixel value are shown in Figure 9.

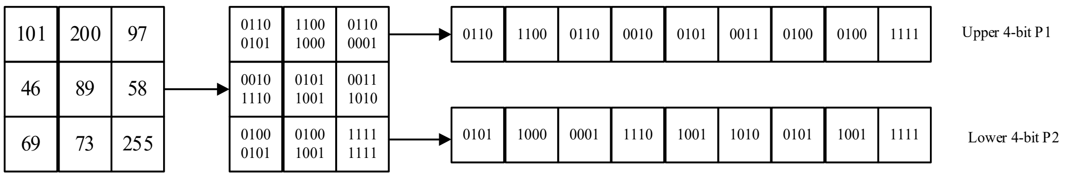


Figure 9. Plaintext image processing.

3.3.2. Improved Diagonal Diffusion

In order to make the information of pixel points have a good diffusion effect, many diffusion methods, such as V-diffusion, zigzag diffusion and chunk diffusion, were proposed. A zigzag algorithm was used in [35] and improved to rearrange the pixel values, but the transformation law of zigzag is relatively single, traversing from the upper left foot to the lower right corner in a Z-shaped order, as shown in Figure 10.

However, the security of the zigzag is not guaranteed due to the fact that it is extremely easy to be broken by a single variation.

In order to obtain faster diffusion speed and diffusion effect, this paper proposes a new diagonal diffusion method by randomly and irregularly rearranging the image matrix through a chaotic sequence. The number of pixel values in each row is controlled by the random sequence, there may be no pixel values at some positions of the diagonal, and the matrix presents an irregular arrangement. Through the proposed new diagonal diffusion, it can complete the diagonal of diffusion. Through the comparative analysis of the relevant experiments in Section 4, this scheme has good performance.

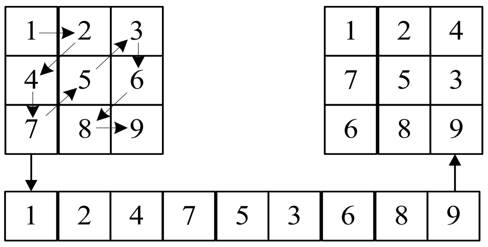


Figure 10. Zigzag transform.

In this paper, after obtaining the upper four-bit matrix, we changed its alignment through a random sequence to generate different irregular matrices, performed diagonal diffusion on the matrix according to the changed alignment and changed its alignment, as shown in Figure 11. Algorithm 3 describes the transformation process.

Algorithm 3 Matrix rearrange

```
Input: upper 4-bits sequence $P1$, random sequence $X'(i)$
Output: irregular matrices $P1'$
    1: Obtain the size of $P1$, $m$ = size $(P1)$
    2: Set an empty list $P1'$ and $i = 0, j = 0$
    3: While $(m - X'(i) > 0)$:
            $P1'(i) = P1(j : j + X'(i))$
            $i = i + 1$
          $j = j + X'(i)$
          $m = m - X'(i)$
    4: $P1'(j) = P1(j :)$
```

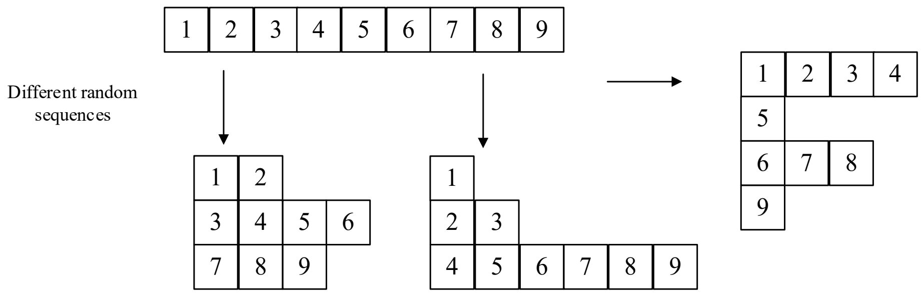


Figure 11. Matrix rearrangement.

Where $P1(i : j)$ denotes from the ith to the jth digit of $P1$, and $P1(i :)$ denotes from the ith to the last digit of $P1$.

Based on the rearranged matrices, we proposed a diagonal diffusion to perform diffusion operations on upper 4-bit matrices capable of new diagonal diffusion operations based on different irregular matrices, as shown in Figure 12, where the arrows represent the order and direction.

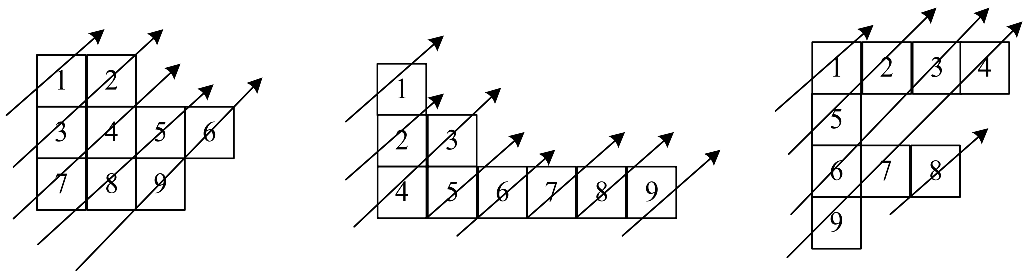


Figure 12. Diagonal diffusion of different matrices.

According to the irregular matrix obtained by Algorithm 3, we need to diffuse according to the diagonal, but the shape of the matrix is irregular, so given Algorithm 4 used to obtain the order of the pixel values of the diagonal of the irregular matrix, sequentially traversing the diagonal to obtain its diffusion order matrix for diffusion, according to different irregular matrices for the order of diagonal diffusions, such as the first: (1- > 3- > 2- > 7- > 4- > 8- > 5- > 9- > 6), the second: (1- > 2- > 4- > 3- > 5- > 6- > 7- > 8- > 9) and the third: (1- > 5- > 2- > 6- > 3- > 9- > 7- > 4- > 8), if we use the normal zigzag, then the order of each diffusion is fixed in a 3 × 3 matrix, and the result is (1- > 2- > 4- > 7- > 5- > 3- > 6- > 8- > 9) each time, which is not guaranteed in terms of security.

Algorithm 4 Diagonal diffusion order

```
Input: irregular matrices $P1'$
Output: order sequence $D$
    1: Obtain the size of $P1'$ $m$ = size $(P1')$
    2: Set tow empty list $D, sub$
    3: for $i$ from 1 to $m$:
            Obtain the size of $P1'$ $n = size(P1'(i))$
            for $i$ from 1 to $n$:
                $sub(i+j).insert(0, P1'(i,j))$
    4: Obtain the size of $sub$ $m = size(sub)$
    5: for $i$ from 1 to $m$:
            Obtain the size of $sub(i)$ $n$ = size $(sub(i))$
            for $i$ from 1 to $n$:
                $D.\ insert(sub(i,j))$
```

Where $sub(i+j).insert(0, P1'(i,j))$ means adding $P1'(i,j)$ to the head of the array $sub(i+j)$ and $D.\ insert(sub(i,j))$ means adding $sub(i,j)$ to the end of the array D.

3.3.3. Permutation and Diffusion Process

Step 1: Generate S-boxes s according to the S-box generation method proposed in Section 3.2.;

Step 2: Generate the encryption key $\hat{\theta}, x_0, y_0, \hat{h}$ and process the image to obtain $P1, P2$ according to Algorithm 1 and θ, x, y;

Step 3: Iterate over $\hat{\theta}, x_0, y_0, \hat{h}$ as the control parameter and initial value of the 2D-CLSM chaos map to generate two chaotic sequences $X = \{x_1, x_2, \ldots, x_{M\times N}\}$, $Y = \{y_1, y_2, \ldots, y_{M\times N}\}$ of length $M \times N$;

Step 4: Process the chaotic sequence to obtain the random numbers for array rearrangement;

$$X'(i) = \left(floor\left(X(i) \times 10^{13}\right) \mathrm{mod} N\right) + 1, i = 1, 2, \ldots, M \times N \tag{15}$$

Step 5: Based on the resulting upper 4-bit matrix $P1$ and the random sequence $X'(i)$, the upper 4-bit matrix is rearranged by Algorithm 3 to obtain a new irregular matrix $P1'$;

Step 6: Perform a diffusion operation on the irregular matrix $P1'$ obtained in step 4. For computational convenience, we first transform the irregular matrix into a one-dimensional matrix D by means of Algorithm 4;

Step 7: Diffusion operation based on the 1D matrix D obtained in step 6 to obtain D';

$$\begin{cases} D'(i) = (X'(1) \mathrm{mod} 16) \oplus D(i), i = 1 \\ D'(i) = D'(i-1) \oplus D(i), i = 2, 3, \ldots, M \times N \end{cases} \tag{16}$$

Step 8: Process the chaotic sequence to obtain chaotic values for the lower four positions;

$$Y'(i) = floor\left(Y(i) \times 10^{13}\right) \mathrm{mod} 65536\ i = 1, 2, \ldots, M \times N \tag{17}$$

Step 9: Based on the resulting $Y'(i)$ the lower four bits of the matrix are permutated.

$$\begin{cases} temp = P2(Y'(i)) \\ P2(Y'(i)) = P2(i) \quad , i = 1, 2, \ldots, M \times N \\ P2(i) = temp \end{cases} \tag{18}$$

3.3.4. Pixel-Level Encryption

At the end of the bit-level encryption, pixel-level encryption is performed by the resulting pixel values using the S-box. It mainly includes pixel value diffusion and substitution. The main process is shown in Figure 13.

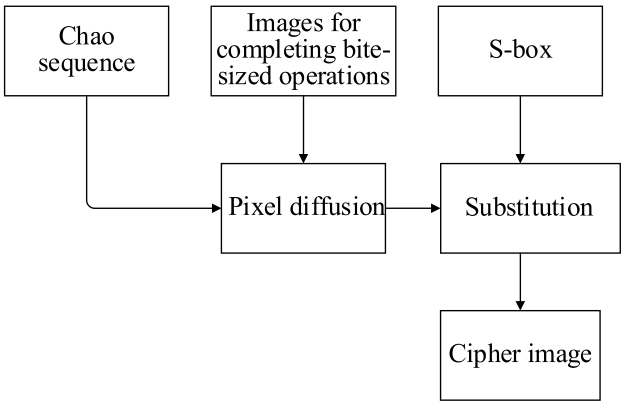


Figure 13. Pixel-level encryption process.

Generate an empty matrix C of size $M \times N$. Based on the matrix $D', P2$ after completing the diffusion and permutation, reorganize $D', P2$ by replacing the original upper four bits as lower four bits and the original lower four bits as upper four bits to obtain 8-bit values between 0 and 255, calculate the row index r and column index c of the S-box and complete the pixel value substitution using the S-box s generated in the second stage.

$$\begin{cases} temp = \mathrm{dec}(\lceil P2(i) + D'(i) \rceil) \oplus X'(i) \oplus C(i-1) \\ c = temp\%16 \\ r = (temp - c)/16 \\ C(i) = s(r,c) \end{cases}, \ i = 1, 2, \ldots, M \times N \tag{19}$$

where $i = 1$, $C(i-1) = 0$; $i = M \times N$, $C(i-1) = C(1)$. $\mathrm{dec}(\lceil a + b \rceil)$ is the reconfiguration of two four-bit binary numbers a, b into an eight-bit binary number, and $s(r,c)$ represent obtaining the value at row r and column c from the S-box s.

Finally, C is converted to an $M \times N$ ciphertext image.

4. Simulation Experiments

The simulation results of the proposed method are presented in this section by running PyCharm software under Windows 10 64-bit system. By using the method in this paper, 10 images were both encrypted and decrypted. The results of encrypting and decrypting several grey-scale images such as Baboon.png, House.png, Cameraman.png and Peppers.png pixels are shown here, respectively. The initial keys used for encryption/decryption are shown in Table 7, and generated keys are shown in Table 8, where $t_0, \hat{\mu}$ are used as initial values and control parameters for the Logistic. $x_0, y_0, \hat{h}, \hat{\theta}$ are used as control parameters, initial values and variables h for the 2D-CLSM.

Table 7. Initial key.

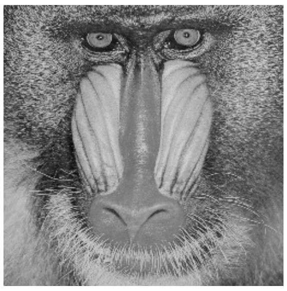
t	μ	θ	x	y	Δm
0.9	0.5	0.2	0.5	0.2	5001

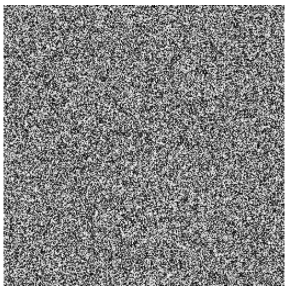
From Table 8, we can clearly see that the encryption keys generated by the algorithm in this paper have good plaintext correlation, can generate completely different encryption keys depending on the plaintext, have good plaintext sensitivity and can effectively resist selective plaintext/known plaintext attacks.

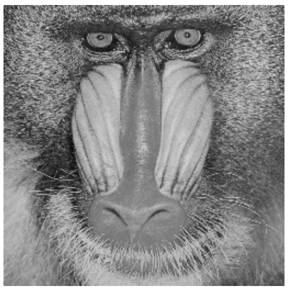
Table 8. Generated encryption keys.

Image	Baboon	House	Cameraman	Peppers
t_0	0.1202650498467574	0.9477909907361988	0.2160499006239973	0.0961937648047185
$\hat{\mu}$	0.9349678430636638	0.9505574749633823	0.9210123850695779	0.9032945073229159
$\hat{\theta}$	0.0664003944646814	0.6920508714375233	0.2309975836114158	0.977565451998436
x_0	0.0264462005880243	0.4490902428109265	0.1811279735456482	0.1440190683132808
y_0	0.0664003944646814	0.6920508714375233	0.2309975836114158	0.977565451998436
$\hat{h}$	20.512754134641057	16.49739814853646	22.592144200059774	11.400485742889497

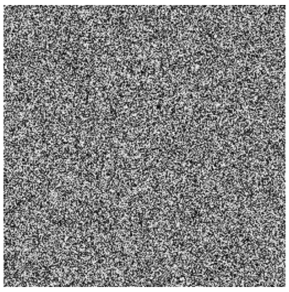
Figure 14a–c shows the experimental results of Baboon, Figure 14d–f shows the experimental results of House, Figure 14g–i shows the experimental results of Cameraman and Figure 14j–l shows the experimental results of Peppers. It is clear from the Figure 14 that the encrypted image can still be fully restored, which verifies the effectiveness of the algorithm in this paper, which can obtain good encryption and decryption results.


(**a**) Original Baboon


(**b**) Cipher Baboon

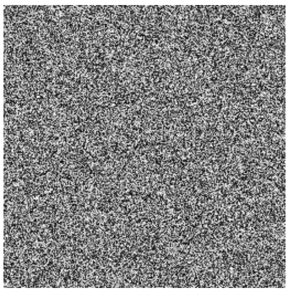

(**c**) Decrypted Baboon

(**d**) Original House


(**e**) Cipher House

(**f**) Decrypted House

(**g**) Original Cameraman


(**h**) Cipher Cameraman

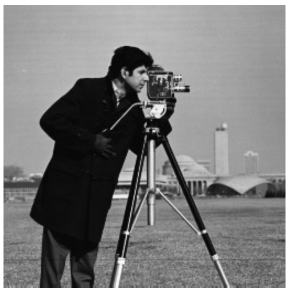
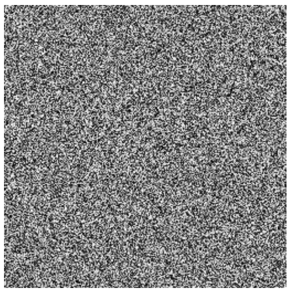

(**i**) Decrypted Cameraman

Figure 14. *Cont.*

(**j**) Original Peppers


(**k**) Cipher Peppers

(**l**) Decrypted Peppers

Figure 14. Encryption and decryption results: (**a**) plaintext image Baboon; (**b**) encrypted image Baboon; (**c**) decrypted image Baboon; (**d**) plaintext image House; (**e**) encrypted image House; (**f**) decrypted image House; (**g**) plaintext image Cameraman; (**h**) encrypted image Cameraman; (**i**) decrypted image Cameraman; (**j**) plaintext image Peppers; (**k**) encrypted image Peppers; (**l**) decrypted image Peppers.

4.1. Security Analysis

4.1.1. Key Space Analysis

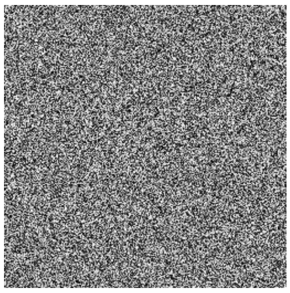
Assuming that the accuracy of the computer is 10^{-16}, in order to make the key sufficiently resistant to brute-force attacks, the key space of the encryption system must be more than 2^{100}, and the parameters of the encryption system are $t_0, \hat{\mu}, \hat{\theta}, x_0, y_0, \hat{h}$, $t_0 \in [0,1]$, $\hat{\mu} \in [0.89, 1], \hat{\theta} \in [0,1], x_0 \in [0,1], y_0 \in [0,1], \hat{h} \in [10.24]$, so the key space is calculated as in Equation (20). The key space is larger than 2^{100}, so the algorithm in this paper has sufficient key space to resist brute-force cracking attacks.

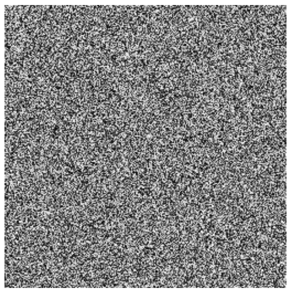
$$0.11 \times 10^{16} \times 10^{16} \times 10^{16} \times 10^{16} \times 10^{16} \times 14 \times 10^{16} > 2^{100} \tag{20}$$

4.1.2. Key Sensitivity Analysis

In a secure encryption system, a mirror change in the key can cause a complete change in the ciphertext obtained from the encryption. Through testing, the encryption key t, μ of the Peppers graph was increased by 10^{-16} to the original one, respectively, and the rest of the keys were left unchanged, and the Peppers were encrypted with the modified key and the original key, respectively, and the results obtained are shown in Figure 15a–f.

As shown in Figure 15, when the initial key is slightly changed, the resulting encrypted image is completely changed, which shows that the algorithm has good key sensitivity and only the correct key can decrypt the ciphertext.


(**a**) Original keys t


(**b**) Changed keys $t + 10^{-16}$

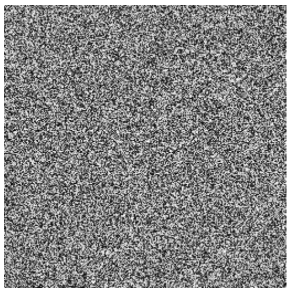
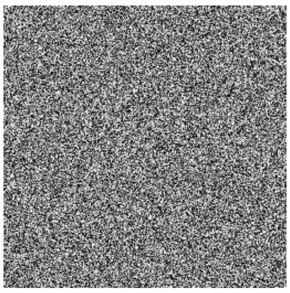
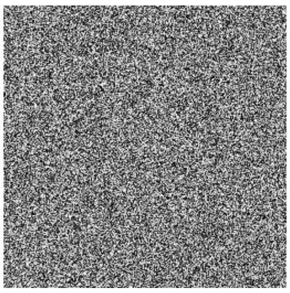

(**c**)Difference diagram

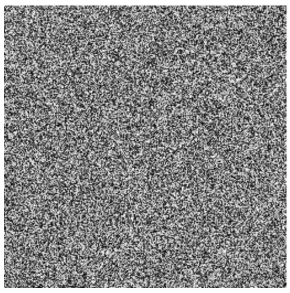
Figure 15. *Cont.*



(**d**) Original keys μ



(**e**) Changed keys $\mu + 10^{-16}$



(**f**) Difference diagram

Figure 15. Key sensitivity analysis: (**a**) image encrypted with original key t; (**b**) image encrypted with original key $t + 10^{-16}$; (**c**) the difference between (**a**,**b**); (**d**) image encrypted with original key μ; (**e**) image encrypted with original key $\mu + 10^{-16}$; (**f**) the difference between (**d**,**e**).

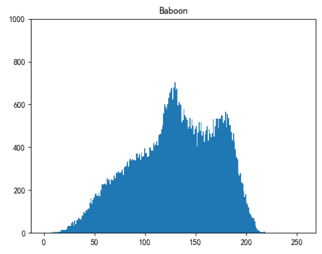
4.1.3. Histogram Analysis

The distribution of the image's pixel values can be directly described by the histogram. A secure encryption system should ensure that pixel values of the obtained ciphertext image are uniformly distributed in order to reduce readability, improve security and provide effective protection against statistical attacks.

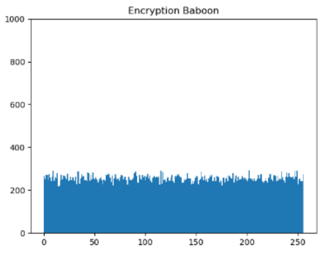
Figure 16 shows the histogram of plaintext and ciphertext pixel frequencies for Baboon, House, Cameraman and Peppers, from which we can see that the pixel of ciphertext images is uniformly distributed. The statistical properties of the images were altered so that they are well resistant to statistical analysis attacks.

The variance of the histogram can be used to specifically describe the distribution of pixel values. The calculation equation is shown in (21), and the result shows in Table 9.

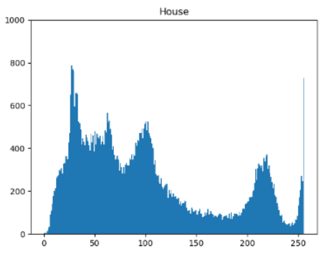
$$var(z) = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \frac{1}{z} (z_i - z_j)^2 \tag{21}$$



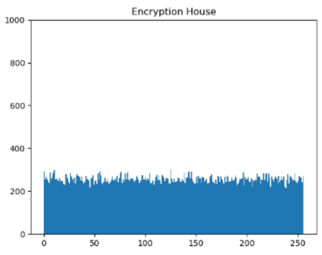
(**a**) Original Baboon



(**b**) Cipher Baboon



(**c**) Original House



(**d**) Cipher House

Figure 16. *Cont.*

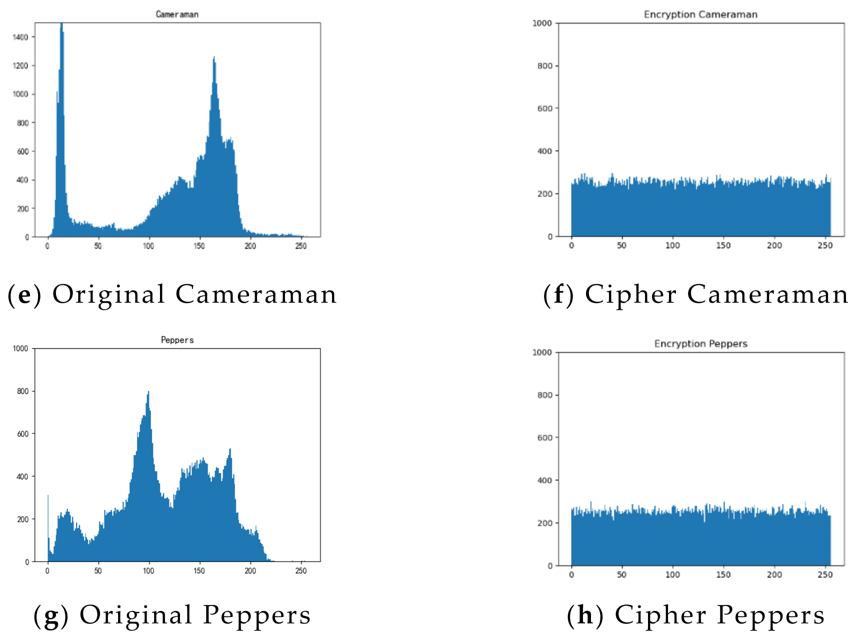


Figure 16. Plain text image histogram and Ciphertext image histogram analysis: (**a**) plaintext Baboon histogram; (**b**) Ciphertext Baboon histogram; (**c**) plaintext House histogram; (**d**) Ciphertext House histogram; (**e**) plaintext Cameraman histogram; (**f**) Ciphertext Cameraman histogram; (**g**) plaintext Peppers histogram; (**h**) Ciphertext Peppers histogram.

Table 9. Variance of histograms of encrypted images.

Image	Plain Image	Ours	Ref. [18]	Ref. [19]	Ref. [20]	Ref. [22]
Baboon	47,065.25	233.35	270.33	270.68	271.31	237.31
House	28,706.41	217.75	255.35	263.38	246.16	246.76
Cameraman	10,5149.27	247.92	269.91	249.93	265.32	259.79
Peppers	35,550.14	246.04	234.49	242.46	240.74	242.06
Average	54,513.19	236.26	257.52	256.61	255.88	246.48

Where z represents histogram values; N represents the total number of samples (N for an image with a grey level of 8 is 256); z_i, z_j are number of pixels with grey values at i, j. The smaller the histogram's variance, the more evenly distributed the histogram.

Table 9 compares the variance of histograms with methods in Refs. [18–22], from which we can see that our encrypted Baboon, House, and Cameraman have the lowest variance of all the schemes, and Peppers is slightly higher than the others. The average variance of the four images is 236.26, which is the lowest of the four methods, so it performs better in resisting statistical attacks.

4.1.4. Plaintext Sensitivity Analysis

A differential attack is a common form of attack in cryptography, in which an attacker usually selects to encrypt the plaintext and a slightly altered plaintext and analyses them to find a specific relationship. In order to be effective against differential attacks, the encryption algorithm should be sensitive enough to the plaintext that a diffusion operation during encryption can cause small changes in the plaintext to affect all pixels, and we usually evaluate the sensitivity of the algorithm using the pixel change rate NPCR and the normalized pixel average change intensity UACI, their ideal expectations are 99.6094% and 33.4635%, respectively. We obtained NPCR and UACI values by randomly modifying two pixel points of the plaintext image. In this paper, we choose to modify pixel points (2, 2) and (236, 207). In Table 10, we give the corresponding experimental results as well as the corresponding results for the other four scenarios. The calculation expressions are shown in Equations (22)–(24).

$$NPCR = \frac{1}{M \times N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} D(i,j) \times 100\% \tag{22}$$

$$D(i,j) = \begin{cases} 0, C(i,j) = C'(i,j) \\ 1, C(i,j) \neq C'(i,j) \end{cases} \tag{23}$$

$$UACI = \frac{1}{M \times N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \frac{|C_1(i,j) - C_2(i,j)|}{255} \times 100\% \tag{24}$$

where $i = 0, 1, \ldots, M-1, j = 0, 1, \ldots, N-1$, C, C' are original encrypted image and the encrypted image with the plaintext modified, respectively; M and N make the height and length of the image, respectively; and $C(i,j)$ denotes the pixel values at coordinates (i,j) of the encrypted image C.

Table 10. NPCR and UACI values.

Image		NPCR	UACI
Baboon	Ours	99.6246%	33.4949%
	Ref. [18]	99.4863%	32.1574%
	Ref. [19]	99.5616%	33.0145%
	Ref. [20]	99.6551%	33.4146%
	Ref. [22]	99.5256%	33.3324%
House	Ours	99.6292%	33.5274%
	Ref. [18]	99.4515%	33.1501%
	Ref. [19]	99.5849%	33.3829%
	Ref. [20]	99.6149%	33.5051%
	Ref. [22]	99.5951%	33.1216%
Cameraman	Ours	99.6231%	33.4488%
	Ref. [18]	99.5987%	33.2151%
	Ref. [19]	99.5739%	33.3015%
	Ref. [20]	99.6032%	33.5028%
	Ref. [22]	99.5897%	32.7981%
Peppers	Ours	99.5941%	33.5739 %
	Ref. [18]	99.4782%	33.4131%
	Ref. [19]	99.5801%	33.4466%
	Ref. [20]	99.6337%	33.5884%
	Ref. [22]	99.6111%	33.0147%

Table 10 compares the NPCR and UACI values of our scheme with the methods in Refs. [18–22]. In our method, the NPCR and UACI values of Baboon, House and Cameraman images exceed the expected values, and the NPCR values of Peppers are close to the expected values. Of the other methods, only the Cameraman plot of Ref. [20] achieved the desired values for both NPCR and UACI values. In contrast, our encryption algorithm has better sensitivity to plaintext.

4.1.5. Correlation Analysis

When the image is not encrypted, there is a strong correlation between the pixel values of the images; the encryption operation on the plaintext allows us to break this correlation. We calculated the correlation coefficient by randomly selecting n pair of adjacent pixels (a_i, b_i) from the image for which the correlation is to be calculated using the following Equation (25). Table 11 shows the experimental results for a random selection of 3000 pairs of pixel values.

$$r_{ab} = \frac{\sum_{i=1}^{n} (a_i - \overline{a})(b_i - \overline{b})}{\sqrt{\sum_{i=1}^{n} (a_i - \overline{a})^2} \sqrt{\sum_{i=1}^{n} (b_i - \overline{b})^2}} \tag{25}$$

where r_{ab} represents the correlation coefficients, $\overline{a}$, $\overline{b}$ is the mean value.

Table 11. Correlation coefficients.

Image	Directions	Ours	Ref. [18]	Ref. [19]	Ref. [20]	Ref. [22]
Baboon	Horizontal	0.0007	−0.0284	−0.0027	0.0039	0.0034
	Vertical	−0.0030	0.0147	0.0023	0.0103	−0.0019
	Diagonal	0.0080	0.0459	0.0088	−0.0070	0.0005
House	Horizontal	−0.0020	0.0497	0.0047	−0.0063	0.0098
	Vertical	−0.0147	0.0327	0.0030	0.0035	−0.0342
	Diagonal	0.0086	−0.0154	−0.0039	0.0103	0.0196
Cameraman	Horizontal	−0.0055	0.0120	−0.0027	0.0047	0.0023
	Vertical	−0.0008	0.0478	0.00025	0.0018	0.0044
	Diagonal	−0.0005	0.0354	0.0039	−0.0019	−0.0048
Peppers	Horizontal	−0.0029	−0.0654	−0.0008	0.0028	0.0051
	Vertical	0.0089	−0.0259	0.0083	−0.0017	−0.0049
	Diagonal	−0.0088	−0.0351	−0.0012	−0.0103	0.0078

The closer the calculated correlation coefficient is to 1, the stronger the correlation is, and the closer it is to 0, the weaker the correlation is. As shown in Table 11, take baboon as an example; we are able to see in Table 11 that it has a horizontal correlation of 0.0007, a vertical correlation of −0.003 and a diagonal correlation of 0.008, with a correlation index very close to 0 in three directions, indicating that the high correlation between pixels is broken.

In addition, from Figure 17, we can clearly see that for the plaintext image, most of the points are close to the diagonal of the axes, whereas, for the encrypted image, these points are randomly distributed throughout the space, with significantly lower inter-pixel correlation. Both illustrate the effectiveness of our algorithm in removing intra-pixel correlations.

4.1.6. Information Entropy Analysis

Information entropy is another metric for evaluating the security of a ciphertext and represents the strength of the uncertainty of the image information, which is expressed in Equation (12).

For a grey-scale image with 256 states, it has a maximum expectation value of 8, Table 12 shows the test results we obtained and the comparison of the other four schemes, where the result of the three encrypted images of House, Baboon and Peppers are higher than the other schemes, and the results of all four images are close to the ideal value of 8. It can be said that the image results processed by our encryption scheme are close to the random images, with good uncertainty, and can effectively resist statistical analysis attacks.

Table 12. Comparison of ciphertext information entropy.

Methods	Baboon	House	Cameraman	Peppers
Ours	7.9974	7.9976	7.9972	7.9972
Ref. [18]	7.9865	7.9907	7.9716	7.9930
Ref. [19]	7.9609	7.9611	7.9581	7.9592
Ref. [20]	7.9974	7.9969	7.9974	7.9967
Ref. [22]	7.9672	7.9858	7.9773	7.9668

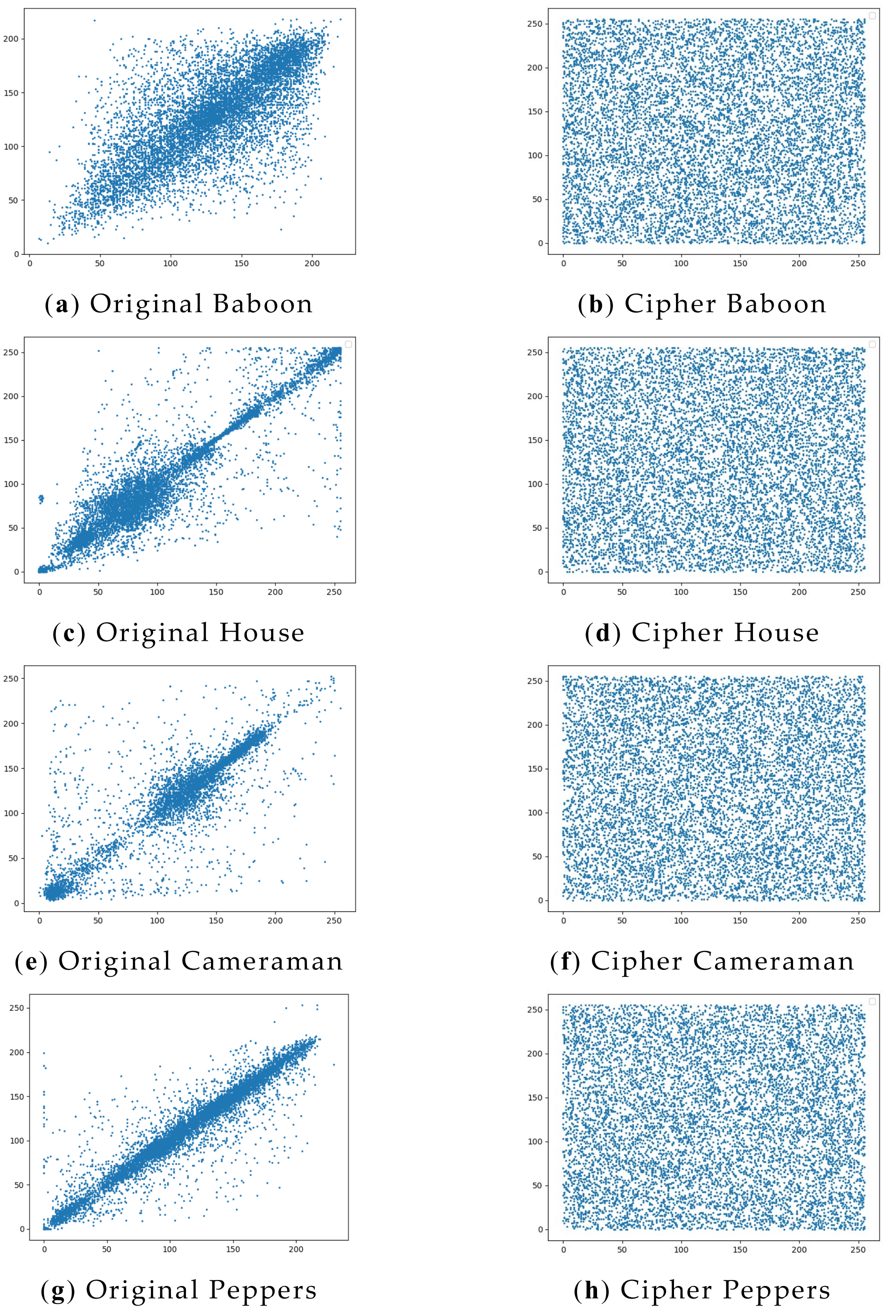
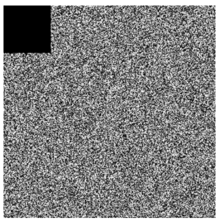


Figure 17. Plain text image correlation and Ciphertext image correlation analysis: (**a**) plaintext Baboon diagonal direction; (**b**) Ciphertext Baboon diagonal direction; (**c**) plaintext House diagonal direction; (**d**) Ciphertext House diagonal direction; (**e**) plaintext Cameraman diagonal direction; (**f**) Ciphertext Cameraman diagonal direction; (**g**) plaintext Peppers diagonal direction; (**h**) Ciphertext Peppers diagonal direction.

4.1.7. Anti-Cropping Attack Analysis

During the transmission of information, information may be lost due to humans or some uncontrollable factors. If the encryption algorithm is able to restore the plaintext image in this case, then the encryption algorithm is valid. In Figure 18, we add a 5% loss of pixel values to the Peppers. From the perspective of the decrypted image, even though the image has lost some data, it is still possible to decrypt the basic information and be able to recover it basically, so the algorithm is resistant to basic cropping attacks.

(**a**) Original Peppers


(**b**) 5% loss cipher Peppers

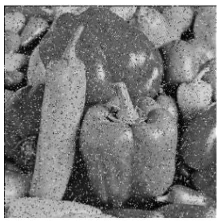

(**c**) Decrypted Peppers

Figure 18. The cropped encrypted image and the corresponding decrypted image. (**a**) The plain image of Peppers. (**b**) The cipher image of Peppers with 5% loss. (**c**) The decrypted image of (**b**).

4.1.8. Speed Analysis

Running speed is an important index to evaluate the encryption algorithm; we simulated it in Python 3.7 environment on a PC with a 2.2 GHz CPU and 8 G RAM. As can be seen from Table 13, this paper's encryption efficiency is superior to that of Refs. [11–13] while being slightly slower than that of Ref. [15]. This shows that our proposed encryption algorithm performs well in terms of efficiency and is capable of being applied in practical applications. Because we used a round of bitrate encryption and a round of pixel-level encryption, the efficiency is reduced, but the security is improved.

Table 13. Comparison of encryption algorithm runtimes (seconds).

Ref. [18]	Ref. [19]	Ref. [20]	Ref. [22]	Ours
1.84	1.69	1.91	0.83	1.12

5. Conclusions

In this paper, we propose an improved cascaded chaotic map 2D-CLSM and design a novel encryption scheme based on it and S-box. By generating a key associated with the plaintext, the whole encryption process is associated with the plaintext, effectively resisting the chosen/known plaintext attack. An S-box encryption scheme is designed and applied to our encryption, increasing the overall security of the algorithm. Dividing the encryption into bit-level encryption and pixel-level encryption improves the complexity and security of the encryption to a certain extent but reduces its efficiency. The encryption method in this paper can be applied to different image types, such as grey-scale images, grey-scale medical images, etc., while the algorithm is able to meet everyday encryption requirements, both in terms of efficiency and security. The experimental results show that the algorithm has sufficient key space, is resistant to brute force attacks and performs well against statistical analysis attacks, clipping attacks and differential attacks. However, although we improved the efficiency of S-box generation, we used two rounds of encryption, one at the bit level and one at the pixel level, which makes us less efficient overall, so the next step is to improve the efficiency of encryption in both rounds.

Author Contributions: Investigation, J.Z. and T.B.; Writing—original draft, T.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by the National Natural Science Foundation of China (Grant No. 62272077).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: The authors are thankful to the reviewers for their comments and suggestions to improve the quality of the manuscript.

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. Wang, M.; Liu, H.; Zhao, M. Bit-level image encryption algorithm based on random-time S-Box substitution. *Eur. Phys. J. Spec. Top.* **2022**, *24*, 3225–3237. [CrossRef]
2. Xiang, H.; Liu, L. An improved digital logistic map and its application in image encryption. *Multimed. Tools Appl.* **2020**, *79*, 30329–30355. [CrossRef]
3. Ratan, R.; Yadav, A. Security analysis of bit-plane level image encryption schemes. *Def. Sci. J.* **2021**, *71*, 209–221. [CrossRef]
4. Pareek, N.K.; Patidar, V.; Sud, K.K. Image encryption using chaotic logistic map. *Image Vis. Comput.* **2006**, *24*, 926–934. [CrossRef]
5. Farhan, A.K.; Al-Saidi, N.M.G.; Maolood, A.T. Entropy analysis and image encryption application based on a new chaotic system crossing a cylinder. *Entropy* **2019**, *21*, 958. [CrossRef]
6. Luo, Y.; Lin, J.; Liu, J. A robust image encryption algorithm based on Chua's circuit and compressive sensing. *Signal Process.* **2019**, *161*, 227–247. [CrossRef]
7. Zhou, W.; Wang, X.; Wang, M. A new combination chaotic system and its application in a new Bit-level image encryption scheme. *Opt. Lasers Eng.* **2022**, *149*, 106782. [CrossRef]
8. Zheng, J.; Liu, L.F. Novel image encryption by combining dynamic DNA sequence encryption and the improved 2D logistic sine map. *IET Image Process.* **2020**, *14*, 2310–2320. [CrossRef]
9. Lan, R.; He, J.; Wang, S. Integrated chaotic systems for image encryption. *Signal Process.* **2018**, *147*, 133–145. [CrossRef]
10. Wang, T.; Ge, B.; Xia, C. Multi-Image Encryption Algorithm Based on Cascaded Modulation Chaotic System and Block-Scrambling-Diffusion. *Entropy* **2022**, *24*, 1053. [CrossRef]
11. Adams, C.; Tavares, S. The structured design of cryptographically good s-boxes. *J. Cryptol.* **1990**, *3*, 27–41. [CrossRef]
12. Arslan, S.; Fawad, A. Image encryption using dynamic S-box substitution in the wavelet domain. *Wirel. Pers. Commun.* **2020**, *115*, 2243–2268.
13. Yang, C.; Wei, X.; Wang, C. S-Box design based on 2D multiple collapse chaotic map and their application in image encryption. *Entropy* **2021**, *23*, 1312. [CrossRef] [PubMed]
14. Li, Y.; Ge, G.; Xia, D. Chaotic hash function based on the dynamic s-box with variable parameters. *Nonlinear Dyn.* **2016**, *84*, 2387–2402. [CrossRef]
15. Zhou, S.; He, P.; Kasabov, N. A dynamic DNA color image encryption method based on SHA-512. *Entropy* **2020**, *22*, 1091. [CrossRef]
16. Wang, X.; Yang, J. A novel image encryption scheme of dynamic s-boxes and random blocks based on spatiotemporal chaotic system. *Optik* **2020**, *217*, 164884. [CrossRef]
17. Belazi, A.; El-Latif, A. A simple yet efficient S-box method based on chaotic sine map. *Optik* **2017**, *130*, 1438–1444. [CrossRef]
18. Beg, S.; Baig, F.; Hameed, Y. Thermal image encryption based on laser diode feedback and 2D logistic chaotic map. *Multimed. Tools Appl.* **2022**, *81*, 26403–26423. [CrossRef]
19. Liu, H.; Liu, J.; Ma, C. Constructing dynamic strong S-Box using 3D chaotic map and application to image encryption. *Multimed. Tools Appl.* **2022**, *81*, 12069. [CrossRef]
20. Zheng, J.M.; Zeng, Q.X. An image encryption algorithm using a dynamic s-box and chaotic maps. *Appl. Intell.* **2022**, *52*, 15703–15717. [CrossRef]
21. Özkaynak, F. Construction of robust substitution boxes based on chaotic systems. *Neural Comput. Appl.* **2019**, *31*, 3317–3326. [CrossRef]
22. Wang, X.; Çavuşoğlu, Ü.; Kacar, S. S-Box based image encryption application using a chaotic system without equilibrium. *Appl. Sci.* **2019**, *9*, 781. [CrossRef]
23. Wang, J.; Zhu, Y.; Zhou, C. Construction method and performance analysis of chaotic s-box based on a memorable simulated annealing algorithm. *Symmetry* **2020**, *12*, 2115. [CrossRef]
24. Zhou, G.; Zhang, D.; Liu, Y. A novel image encryption algorithm based on chaos and Line map. *Neurocomputing* **2015**, *169*, 150–157. [CrossRef]
25. Zhou, Y.; Hua, Z.; Pun, C.M. Cascade chaotic system with applications. *IEEE Trans. Cybern.* **2014**, *45*, 2001–2012. [CrossRef]
26. Alawida, M.; Samsudin, A.; Teh, J.S. Digital cosine chaotic map for cryptographic applications. *IEEE Access* **2019**, *7*, 150609–150622. [CrossRef]
27. Lu, Q.; Zhu, C.; Deng, X. An efficient image encryption scheme based on the LSS chaotic map and single s-box. *IEEE Access* **2020**, *8*, 25664–25678. [CrossRef]
28. Hua, Z.Y. 2D Logistic-Sine-Coupling map for image encryption. *Signal Process.* **2018**, *149*, 148–161. [CrossRef]
29. Zhang, Y. The unified image encryption algorithm based on chaos and cubic s-box. *Inf. Sci.* **2018**, *450*, 361–377. [CrossRef]
30. Wang, X.; Su, Y.; Liu, L. Color image encryption algorithm based on Fisher-Yates scrambling and DNA subsequence operation. *Vis. Comput.* **2021**, *37*, 2311. [CrossRef]
31. Liu, H.; Kadir, A.; Xu, C. Cryptanalysis and constructing S-box based on chaotic map and backtracking. *Appl. Math. Comput.* **2020**, *376*, 125153. [CrossRef]
32. Farwa, S.; Muhammad, N. A Novel Image Encryption Based on Algebraic s-box and Arnold Transform. *3D Res.* **2017**, *8*, 26. [CrossRef]

33. Lu, Q.; Zhu, C.; Wang, G. A novel s-box design algorithm based on a new compound chaotic system. *Entropy* **2019**, *21*, 1004. [CrossRef]
34. Biham, E.; Shamir, A. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptol.* **1991**, *4*, 3–72. [CrossRef]
35. Wang, X.; Du, X. Chaotic image encryption method based on improved zigzag permutation and DNA rules. *Multimed. Tools Appl.* **2022**, *81*, 13012. [CrossRef]