Hindawi
Computational Intelligence and Neuroscience
Volume 2021, Article ID 2091053, 13 pages
https://doi.org/10.1155/2021/2091053

Research Article

A Novel Chaotic Image Encryption Algorithm Based on Latin Square and Random Shift

Xuncai Zhang,[1] **Tao Wu**,[1] **Yanfeng Wang**,[1] **Liying Jiang**,[1] **and Ying Niu**[2]

[1]*College of Electrical and Information Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China*
[2]*College of Architecture Environment Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China*

Correspondence should be addressed to Ying Niu; niuying@zzuli.edu.cn

Received 27 May 2021; Accepted 13 August 2021; Published 7 September 2021

Academic Editor: Anastasios D. Doulamis

To realize the safe transmission of images, a chaotic image encryption algorithm based on Latin square and random shift is proposed. The algorithm consists of four parts: key generation, pixel scrambling, pixel replacement, and bit scrambling. Firstly, the key is generated from the plain image to improve the sensitivity of the encryption method. Secondly, each pixel in each row of the image matrix is moved cyclically to the right, in turn, to change the position of the image pixel and realize pixel position scrambling. Then, a 256-order Latin square matrix composed of a chaotic sequence is used as a lookup table, and the replacement coordinates are calculated based on the image pixel value and the chaotic sequence value, replacing the corresponding coordinate elements in the image matrix. Finally, decompose the bitplane of the image matrix and combine it into two-bit matrices, scramble the two bit matrices, respectively, with the Latin square matrix, recombine the scrambled two-bit matrices, and convert them into decimal to obtain the ciphertext image. In the proposed encryption method, all the Latin square matrices used are generated by chaotic sequences, further enhancing the complexity of the generated Latin square matrix and improving the algorithm's security. Experimental results and security analysis show that the proposed algorithm has good security performance and is suitable for image encryption.

1. Introduction

With the rapid development of modern communication technology, more and more digital images are transmitted on social networks. These images carry personal information. Therefore, how to protect this private information has become a research hotspot [1]. Due to the inherent characteristics of strong correlation and high redundancy between adjacent pixels of an image, some traditional encryption methods such as Data Encryption Standard (DES) or Advanced Encryption Standard (AES) are not suitable for performing digital images encryption because traditional encryption algorithms have the disadvantage of low efficiency when encrypting digital images [2].

Because some characteristics of the chaotic system are very suitable for the development of image encryption algorithms, such as sensitivity to initial conditions, unpredictability, and ergodicity, among the currently proposed image encryption methods, chaos systems have been widely used [3–5]. Wang [6] proposed an image encryption method based on a one-dimensional chaotic system, which improved the structure of control parameters and the sensitivity of one-dimensional chaotic mapping and improved its ability to resist differential attacks. Zhou [7] combined two existing one-dimensional chaotic maps and proposed a new one-dimensional chaotic map, which improved the performance against various attacks. However, some inherent shortcomings of one-dimensional chaotic mapping, such as relatively simple structure and small keyspace, are difficult to eliminate. In contrast, high-dimensional chaotic systems have more complex dynamics and high ergodicity, so people apply high-dimensional chaotic systems to image encryption [8–10]. Gan [11] proposed a color image encryption algorithm based on high-dimensional chaos and three-dimensional bit-plane arrangement, which can effectively resist known plaintext attacks and selected plaintext attacks. Zhang [12] proposed a new 3D bit matrix replacement algorithm, and the encryption method developed a new replacement method based on the Chen system to improve the randomness of the scrambling process. However, the encryption scheme is vulnerable to attack by the selected plaintext. Image encryption algorithms based on high-dimensional chaos often scramble the pixels of the image by obtaining the index vector of the chaotic sequence. The security of these scrambling methods only depends on the index vector because the attacker may analyze the difference between the cipher image and the plain image. The relationship to obtain the index vector causes the scrambling operation to be invalid.

In recent years, due to the excellent performance of the Latin square in image encryption methods, it has received extensive attention from researchers. The Latin square is a special square matrix with uniformity. Shannon first pointed out the relationship between the Latin square and cryptography [13]. Latin squares have some good features that are very suitable for image encryption: the number of Latin squares is huge, and the number of Latin squares of the 10th order is about 10^{37}, so its keyspace is large and can prevent brute force attacks. The Latin square has a unified histogram, which means that using the Latin square for image encryption can effectively resist statistical analysis. Because of the good characteristics of the Latin square, Wu [14] proposed a symmetric encryption algorithm and designed a new loom-like 2D substitution-permutation network. This network maintains good confusion and diffusion characteristics while also with additional fault tolerance. Panduranga [15] used a chaotic system and Latin square to construct an image encryption method, which was later cracked by Ahmad M. and Ahmad F. [16].

Aiming at the characteristics of chaotic system and Latin square matrix, an image encryption algorithm (LSRS) based on Latin square and random shift is designed. The algorithm is divided into four parts: key generation, pixel scrambling, pixel replacement, and bit scrambling. Latin squares are all produced by chaotic sequences, which further enhances the complexity of Latin squares. In the pixel scrambling part, the cyclic shift step length of the pixel is controlled by the chaotic sequence, which makes the pixel distribution more random. The histogram of the 256-order Latin square matrix is evenly distributed; using it as a lookup table to replace the pixels can effectively increase the Shannon entropy of the cipher image. At the same time, the image pixel value and the rounded chaotic sequence jointly calculate the coordinates of the replaced pixels, which effectively improves the randomness of the algorithm, through the Latin square matrix to scramble the bit plane of the image matrix to enhance the security of the algorithm. The security analysis of the encrypted cipher image shows that the LSRS algorithm has good security performance and is suitable for practical applications.

The rest of this article is organized as follows: in Section 2, the related concepts of Latin squares and chaotic systems are introduced. Section 3 gives the detailed scheme of the LSRS method. Section 4 analyzes the safety of the proposed method. Finally, the conclusions are given in Section 5.

2. Related Work

2.1. Latin Square. The famous mathematician and physicist Euler used the Latin alphabet as the symbol of the elements in the Latin square, and the Latin square got its name. For an $N \times N$ matrix with only N different elements and each element appears only once in any row or column, the matrix is called a Latin square matrix. The application of the Latin square matrix is to double control the row vector and column vector of the image matrix to promote the uniform distribution of pixels and improve the balance of pixels in the matrix. Figure 1 shows examples of Latin squares with different symbol sets.

2.2. Hyperchaotic Lorenz System. The hyperchaotic Lorenz system has multiple positive Lyapunov exponents, has a high keyspace, and can improve the confidentiality performance of the encryption algorithm [17]. The dynamic formula of hyperchaotic Lorenz is

$$\begin{cases} \dot{x} = -a(y - x) + w, \\ \dot{y} = cx - y - xz, \\ \dot{z} = xy - bz, \\ \dot{w} = -yz + rw, \end{cases} \tag{1}$$

where a, b, c, and r are all control parameters, and when $a = 10$, $b = 8/3$, $c = 28$, and $-1.52 \le r \le -0.06$, formula (1) is in a hyperchaotic state. When $r = -1$, the four Lyapunov exponents in formula (1) are, respectively, $\lambda_1 = 0.3381$, $\lambda_2 = 0.1586$, $\lambda_3 = 0$, $\lambda_4 = -15.1752$, as shown in Figure 2 for the phase of the hyperchaotic Lorenz in the hyperchaotic state.

3. Encryption Scheme

In the encryption algorithm, $\mathbf{P}$ represents a plain image with a size of $N \times N$, and $\boldsymbol{C}$ represents the corresponding cipher image. The frame diagram of the encryption scheme is shown in Figure 3.

3.1. Generate Key. The plain image to generate the key can associate the plain image with the cipher image and enhance the sensitivity of the key. The method of generating the key in this paper is as follows: divide the input image matrix $\mathbf{P}$ into four blocks; calculate the sum of the elements in each matrix; and obtain LL_1, LL_2, LL_3, and LL_4 to generate keys x_0, y_0, z_0, and w_0. The method of its generation is

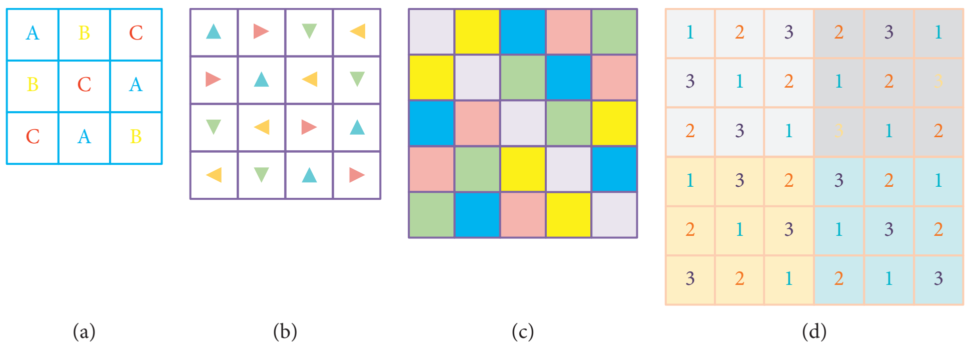

Figure 1: Latin square example. (a) 3×3. (b) 4×4. (c) 5×5. (d) 9×9.

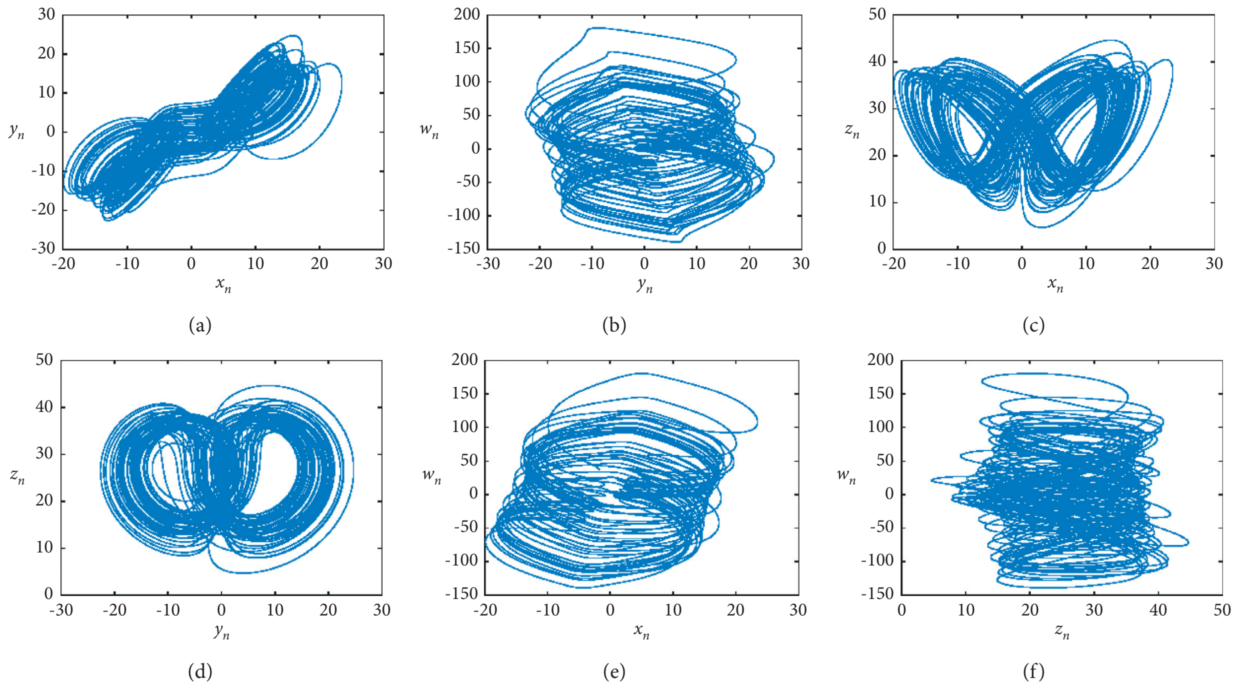

Figure 2: Phase diagram of the hyperchaotic Lorenz system. (a) x_n-y_n phase diagram, (b) y_n-w_n phase diagram, (c) x_n-z_n phase diagram, (d) y_n-z_n phase diagram, (e) x_n-w_n phase diagram, and (f) z_n-w_n phase diagram.

$$\begin{cases} LL1 = \bmod\left(\sum_{i=1}^{\text{floor}(N/2)} \sum_{j=1}^{\text{floor}(N/2)} P(i,\ j), 256\right), \\ LL2 = \bmod\left(\sum_{i=1}^{\text{floor}(N/2)} \sum_{j=\text{floor}(N/2)+1}^{N} P(i,\ j), 256\right), \\ LL3 = \bmod\left(\sum_{i=\text{floor}(N/2)+1}^{N} \sum_{j=1}^{\text{floor}(N/2)} P(i,\ j), 256\right), \\ LL4 = \bmod\left(\sum_{i=\text{floor}(N/2)+1}^{N} \sum_{j=\text{floor}(N/2)+1}^{N} P(i,\ j), 256\right), \end{cases} \tag{2}$$

where $LL = \bmod(x, y)$ returns the remainder after dividing x by y, x is the dividend, y is the divisor, and floor (x) means rounding the elements of x to the negative infinity. The key generation method is

$$\begin{cases} x_0 = \dfrac{|(LL1 - \text{bitxor}(LL2, \text{bitxor}(LL3, LL4))|}{256}, \\ y_0 = \dfrac{|(LL2 - \text{bitxor}(LL1, \text{bitxor}(LL3, LL4))|}{256}, \\ z_0 = \dfrac{|(LL3 - \text{bitxor}(LL1, \text{bitxor}(LL2, LL4))|}{256}, \\ w_0 = \dfrac{|(LL4 - \text{bitxor}(LL1, \text{bitxor}(LL2, LL3))|}{256}, \end{cases} \tag{3}$$

where bitxor represents the bitwise exclusive or between two values and $|x|$ represents rounding of x. Input the generated

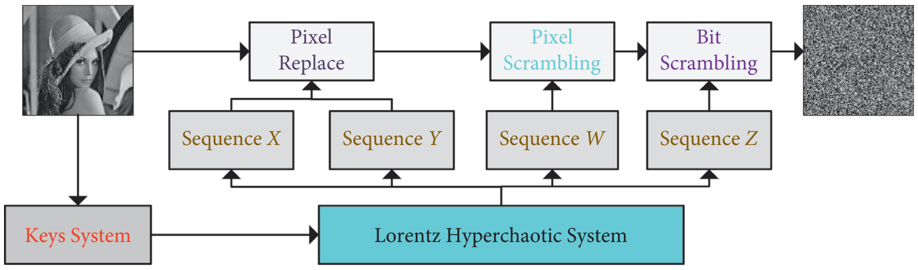

FIGURE 3: Block diagram of the encryption scheme.

keys x_0, y_0, z_0, and w_0 into the chaotic system, and iterate $3M \times M$ times (when $N < 256$, $M = 256$; when $N \geq 256$, $M = N$); the chaotic sequence X, Y, Z, and W used for pixel scrambling, pixel replacement, and bit scrambling is obtained.

3.2. The Generation of Latin Square Matrix.

Given two sequences $\mathbf{Q}_1$ and $\mathbf{Q}_2$ of equal length and sorting them respectively, the corresponding index sequences $\mathbf{Q}_{\mathbf{seed}}$ and $\mathbf{Q}_{\mathbf{shift}}$ are obtained. According to Algorithm 1 [18], generate Latin square matrix $\boldsymbol{L}$.

3.3. Pixel Scrambling.

Pixel scrambling can effectively break the correlation between adjacent pixels and improve the security of encryption algorithms. In this paper, adaptive shifting is used to realize pixel position scrambling, and each pixel in each row (column) is cyclically moved to the right in order from left to right. The step length of each pixel shift in the same row (column) is controlled by the element value of the chaotic sequence, and it is also affected by the previous pixels. This method can increase the complexity of scrambling; even if a chaotic sequence of a certain length is deciphered, it is difficult to restore the image pixels after scrambling correctly. The $N \times N$ elements before the chaotic sequence $\mathbf{X}$ and $\mathbf{Y}$ are, respectively, intercepted, and two equal sequences $\mathbf{Q}_{\mathbf{r1}}$ and $\mathbf{Q}_{\mathbf{r2}}$ are obtained. The preprocessing is performed according to formula (4) to get the sequences $\mathbf{Q}_{\mathbf{u1}}$ and $\mathbf{Q}_{\mathbf{u2}}$, and they are, respectively, converted into $N \times N$. The matrix $\mathbf{Q}_{\mathbf{row}}$ and the matrix $\mathbf{Q}_{\mathbf{col}}$ are used for row scrambling and column scrambling of the input image matrix:

$$\begin{cases} \mathbf{Q}_{\mathbf{u}1}(k) = \mathrm{mod}\left(\mathbf{Q}_{\mathbf{r}1}(k) \times 2^{32}, N\right), \\ \mathbf{Q}_{\mathbf{u}2}(k) = \mathrm{mod}\left(\mathbf{Q}_{\mathbf{r}2}(k) \times 2^{32}, N\right), \end{cases} \tag{4}$$

where $k = 1, 2, 3, \ldots, N$.

First, use the matrix $\mathbf{Q}_{\mathrm{row}}$ to scramble each row of the image matrix $\mathbf{P}$ to obtain the matrix $\mathbf{P}'$, and then use the matrix $\mathbf{Q}_{\mathrm{col}}$ to scramble each column of the matrix $\mathbf{P}'$ to get the scrambled matrix $\mathbf{P}_s$. Figure 4 shows an example of the scrambling process of the image matrix $\mathbf{P}$. To better explain this scrambling method, the first row of the image matrix $\mathbf{P}$ and the first row of the matrix Qrow shown in Figure 4 are taken as an example to introduce the adaptive scrambling process in detail, where the adaptive shifting process of the remaining rows (columns) of the image matrix $\mathbf{P}$ is similar to the scrambling process of the first row of adaptive shifting and will not be described in detail. The scrambling method is as follows:

(1) The sequence of the first row of the image matrix $\mathbf{P}$ is sequence $\mathbf{T} = [101, 33, 44, 55, 12]$, and the sequence of the first row of the matrix $\mathbf{Q}_{\mathrm{row}}$ used to shift the pixels is sequence $\mathbf{U} = [2, 0, 4, 2, 3]$;

(2) The first element in the sequence $\mathbf{T}$ is 101, and its first element in the corresponding pixel shift sequence $\mathbf{U}$ is 2; then, the first element 101 in the sequence $\mathbf{T}$ is cyclically moved two positions to the right, getting sequence $\mathbf{T}_1 = [33, 44, 101, 55, 12]$;

(3) The second element in sequence $\mathbf{T}_1$ is 44, and its second element in the corresponding pixel shift sequence $\mathbf{U}$ is 0; then, the second element 44 in sequence $\mathbf{T}_1$ is cyclically moved zeros positions to the right, getting sequence $\mathbf{T}_2 = [33, 44, 101, 55, 12]$;

(4) The third element in sequence $\mathbf{T}_2$ is 101, and its third element in the corresponding pixel shift sequence $\mathbf{U}$ is 4; then, the third element 101 in sequence $\mathbf{T}_2$ is cyclically moved four positions to the right, getting sequence $\mathbf{T}_3 = [33, 101, 44, 55, 12]$;

(5) The fourth element in sequence $\mathbf{T}_3$ is 55, and its fourth element in the corresponding pixel shift sequence $\mathbf{U}$ is 2; then, the fourth element 55 in sequence $\mathbf{T}_3$ is cyclically moved two positions to the right, getting sequence $\mathbf{T}_4 = [55, 33, 101, 44, 12]$;

(6) The fifth element in sequence $\mathbf{T}_4$ is 12, and its fifth element in the corresponding pixel shift sequence $\mathbf{U}$ is 3; then, the fifth element 12 in sequence $\mathbf{T}_4$ is cyclically moved three positions to the right, getting sequence $\mathbf{T}_5 = [55, 33, 12, 101, 44]$;

(7) Repeat the scrambling processes (1)–(6) to the remaining rows of the image matrix $\mathbf{P}$, until each row is scrambled to obtain the matrix $\mathbf{P}'$.

(8) Similar to processes (1)–(7), use matrix $\mathbf{Q}_{\mathbf{col}}$ to scramble each column of matrix $\mathbf{P}'$ to obtain the scrambled matrix $\mathbf{P}_{\mathrm{S}}$.

3.4. Pixel Replacement.

In the encryption algorithm, replacing the pixels of the plain image can effectively hide the original information of the plain image. However, some encryption algorithms use the replacement method of addition, subtraction, and xor, which is too simple to protect the image information effectively. To solve this defect,

```
Input: $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are two sequences of equal length
Output: Latin matrix with order equal to the length of the input sequence
[~, $\mathbf{Q}_{\mathbf{seed}}$] = Sort ($\mathbf{Q}_1$);
[~, $\mathbf{Q}_{\mathbf{shift}}$] = Sort ($\mathbf{Q}_2$);
for $i = 0:1$: length ($\mathbf{Q}_{\mathbf{shift}}$)-1
        $\mathbf{L}$ = Rowshift ($\mathbf{Q}_{\mathbf{seed}}$, $\mathbf{Q}_{\mathbf{shift}}$ ($i$))
end
where Sort ($x$) is a sorting function, which can sort the sequence $x$ in ascending order and return the sorted sequence and its position index sequence in the original sequence. $\mathbf{Q}_{\mathbf{seed}}$ and $\mathbf{Q}_{\mathbf{shift}}$ are the position index sequences obtained after sorting the sequences $\mathbf{Q}_1$ and $\mathbf{Q}_2$, length ($x$) represents the length of the sequence $x$, and Rowshift ($x$, $y$) means to move the sequence $\mathbf{Q}_{\mathbf{seed}}$ loop to the left by $y$ positions.
```

ALGORITHM 1: Latin square produces $\mathbf{L} = \text{Latin}(\mathbf{Q}_1, \mathbf{Q}_2)$.

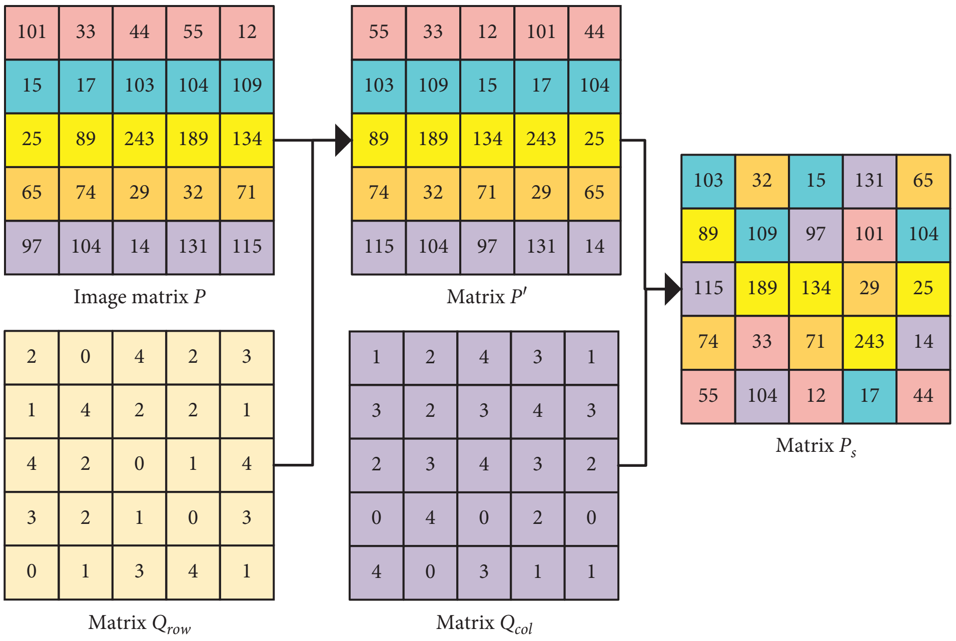

FIGURE 4: Example of scrambling of image matrix (P).

generate a 256-order Latin square matrix, and use the elements in this Latin square matrix to replace the pixels of the image. The number of Latin squares of order 256 is about $256! \approx 2^{1684}$, so its keyspace is large enough, and its histogram is evenly distributed, which is difficult to crack using statistical attack analysis.

Given two sequences $\mathbf{Q}_{t1}$ and $\mathbf{Q}_{t2}$ of length 256, generate a Latin square matrix $\mathbf{L}_{\text{table}}$ as a lookup table. According to formula (5), get $\mathbf{Q}_{s1}$ and $\mathbf{Q}_{s2}$ and respectively converted into $N \times N$ matrices $\mathbf{L}_c$ and $\mathbf{L}_r$:

$$\begin{cases} \mathbf{Q}_{s1} = \text{mod}\left(\mathbf{Z}[513:\ (512 + N \times N))] \times 2^{32},\ 256\right), \\ \mathbf{Q}_{s2} = \text{mod}\left(\mathbf{Z}[(513 + N \times N):\ (512 + 2 \times N \times N)] \times 2^{32},\ 256\right), \end{cases} \tag{5}$$

where $\mathbf{Z}[a\text{: } b]$ means to intercept the elements whose index values are between a and b from the sequence $\mathbf{Z}$ (including the elements corresponding to the index values a and b). Through the matrix $\mathbf{L}_c$ and the corresponding coordinate element of the image matrix to be replaced to do the addition operation, take the remainder so that the elements in the resulting remainder matrix $\mathbf{L}_c'$ are between 0 and 255. Using $\mathbf{L}_c'$ as the row coordinate index matrix and the matrix $\mathbf{L}_\mathbf{r}$ as the column coordinate index matrix, from the lookup table, search the corresponding element for replacement, and get the pixel replacement matrix $\mathbf{P}_r$. The replacement method is $\mathbf{P}_r(i,j) = \mathbf{L}_{\mathbf{table}}(\mathbf{L}_c'(i,j), \mathbf{L}_r(i,j))$, where $\mathbf{L}_c'(i,j)$ indicates the row index and $\mathbf{L}_r(i,j)$ indicates the column index. Algorithm 2 is the Latin square replacement process.

3.5. Bit Scrambling.

3.5. Bit Scrambling. The pixel value of grayscale images ranges from 0 to 255, and an 8-bit binary sequence can represent each pixel. Therefore, the grayscale image can be decomposed into eight bit-planes, where the ith bit plane is composed of the ith bit of all pixels ($i = 1, 2, \ldots, 8$). The high bit-plane contains the visual information of the plain image, and the low-bit plane contains the detailed information of the plain image [19]. Mixing the bits in the high-bit plane and the low-bit plane of the plain image can hide the

```
Input: Plain image $\mathbf{P}$, $\mathbf{L}_c$, $\mathbf{L}_r$, and $\mathbf{L}_{\text{table.}}$
Output: Image $\mathbf{P}_r$ after pixel replacement.
for $i = 0:1: N-1$ do
    for $j = 0:1: N-1$ do
        $\mathbf{P}_r\ (i, j) = \mathbf{L}_{\text{table}}\ (\mathbf{L}_c'\ (i, j), \mathbf{L}_r\ (i, j))$
    end
end
where bitget ($\mathbf{P}$, $i$) ($i = 1, 2, \ldots, 8$) represents obtaining the $i$th bit plane of the plain image $\mathbf{P}$.
```

Algorithm 2: Latin square substitution $\mathbf{P}_r = \text{Replace}\ (\mathbf{P}, \mathbf{L}_c'\ (i, j), \mathbf{L}_r, \mathbf{L}_{\text{table}})$.

information of the plain image and improve the security of the algorithm.

To better scramble the bits of the image, we decompose the image into eight bit-planes and then recombine and scramble them. First, the image matrix **P** is decomposed into eight bit-planes **P** (1)–**P** (8). Secondly, the four bit-planes **P** (1), **P** (3), **P** (5), and **P** (7) of the image matrix **P** combine into a $2N \times 2N$ bit matrix **PA**, and the four bit-planes **P** (2), **P** (4), **P** (6), and **P** (8) of the image matrix **P** are combined into a $2N \times 2N$ bit matrix **PB**. Then, intercept $8N$ elements from the chaotic sequence **W** and divide them into four equal sequences $\mathbf{QD}_1$, $\mathbf{QD}_2$, $\mathbf{QD}_3$, and $\mathbf{QD}_4$. Finally, use the sequences $\mathbf{QD}_1$ and $\mathbf{QD}_2$ to generate the Latin square matrix $\mathbf{LH}_1$, and use the sequences $\mathbf{QD}_3$ and $\mathbf{QD}_4$ to generate the Latin square matrix $\mathbf{LH}_2$, $\mathbf{LH}_1$, and $\mathbf{LH}_2$, respectively used for bit scrambling of matrix **PA** and matrix **PB**. Figure 5 shows the bit plane of the image matrix **P** is scrambled. Decompose the image matrix P into eight bit-planes, and the bit plane **P** (1) as shown in Figure 5(a). Arrange according to the method shown in Figure 5(b) to obtain a bit matrix **PA** and a bit matrix **PB**. Use the matrix $\mathbf{LH}_1$ to perform row scrambling on the bit matrix **PA** to get $\mathbf{PA}'$, use $\mathbf{LH}_2$ to perform column scrambling on the bit matrix **PB** to obtain $\mathbf{PB}'$, as shown in Algorithm 3, the scrambling process. The bit matrix $\mathbf{PA}'$ and bit matrix $\mathbf{PB}'$ as shown in Figure 5(c) are obtained. Finally, through the bit matrix $\mathbf{PA}'$ and bit matrix $\mathbf{PB}'$ recombined into eight bit-planes and converted to decimal, the resulting image matrix $\mathbf{P_d}$ is shown in Figure 5(d).

3.6. Image Encryption Process.

The LSRS algorithm proposed in this paper consists of four parts: key generation, pixel scrambling, pixel replacement, and bit scrambling. The steps of the encryption algorithm are described as follows:

Step 1. Through the image matrix **P** of size $N \times N$, generate the initial parameters x_0, y_0, z_0, and w_0 of the *keys*. Input the generated *keys* parameters x_0, y_0, z_0, and w_0 into the hyperchaotic Lorenz system, iterate $999 + 3M \times M$ times (when $N < 256$, $M = 256$, when $N \geq 256$, $M = N$), and discard the first 999 times; the chaotic sequence **X**, **Y**, **Z**, **W** was obtained.

Step 2. Intercept the first $N \times N$ elements of the chaotic sequence **X** and **Y**, respectively, as the sequences $\mathbf{Q}_{r1}$ and $\mathbf{Q}_{r2}$; preprocess them according to formula (4); convert them, respectively, into $N \times N$ matrix $\mathbf{Q}_{\text{row}}$ and matrix $\mathbf{Q}_{\text{col}}$; and perform row scrambling and column scrambling of matrix **P** on the image to obtain the scrambling image matrix $\mathbf{P}_s$.

Step 3. Intercept the first 512 elements of the chaotic sequence **Z**, divide them into two equal sequences $\mathbf{Q}_{t1}$ and $\mathbf{Q}_{t2}$, and use them to generate a Latin square matrix $\mathbf{Q}_{\text{table}}$ as a lookup table. According to formula (5) to obtain $\mathbf{Q}_{s1}$ and $\mathbf{Q}_{s2}$, respectively convert into $N \times N$ matrices $\mathbf{L}_c$ and $\mathbf{L}_r$. Through the matrix $\mathbf{L}_c$ and the corresponding coordinate element of the image matrix to be replaced to do the addition operation, take the remainder so that the elements in the resulting remainder matrix $\mathbf{L}_c^{'}$ are between 0 and 255. The matrix $\mathbf{L}_c^{'}$ and the matrix $\mathbf{L}_r$ form a two-dimensional index matrix, and the index matrix is used to select elements from the corresponding coordinates in the matrix $\mathbf{Q}_{\text{table}}$ and replace the elements in the matrix $\mathbf{P}_s$ to obtain the pixel replacement matrix $\mathbf{P}_r$.

Step 4. Intercept $8N$ elements from the chaotic sequence **W** , divide them into four sequences $\mathbf{QD}_1$, $\mathbf{QD}_2$, $\mathbf{QD}_3$, and $\mathbf{QD}_4$ of equal length. Use $\mathbf{QD}_1$ and $\mathbf{QD}_2$ to generate Latin square matrix $\mathbf{LH}_1$, $\mathbf{QD}_3$, and $\mathbf{QD}_4$ to generate Latin square matrix $\mathbf{LH}_2$. Divide the matrix $\mathbf{P}_\mathrm{r}$ into eight bit-planes **P** (1)–**P** (8), where the four bit-planes **P** (1), **P** (3), **P** (5), and **P** (7) are combined into a $2N \times 2N$ bit matrix **PA**, and **P** (2), **P** (4), **P** (6), and **P** (8) four bit-planes are combined into a $2N \times 2N$ bit matrix **PB**. Through matrix $\mathbf{LH}_1$ to scramble the rows of matrix **PA**, $\mathbf{LH}_2$ performs column scrambling on the matrix **PB**. The scrambled matrix $\mathbf{PA}'$ and matrix $\mathbf{PB}'$ are, respectively, divided into four bit-planes, and the resulting bit planes are combined into a bit plane matrix P_{bit}'. Convert the elements in the bit plane matrix P_{bit}' to decimal data and finally obtained the cipher image **C**.

The reverse process of the encryption algorithm is the decryption process, which will not be repeated.

3.7. Simulation Results.

In order to study the confidentiality performance of the LSDR method, use MATLAB 2019a to simulate the encryption algorithm. The configuration environment of the computer is Windows 10, 8.00 GB RAM, Intel (R) Core (TM) i7-4510 CPU @ 2.00 GHz. Figure 6 shows the plain image, cipher image, and decrypted image of Lena, Boat, Hill, and Peppers. By directly observing the cipher image, no valid information can be identified, so the algorithm is feasible.

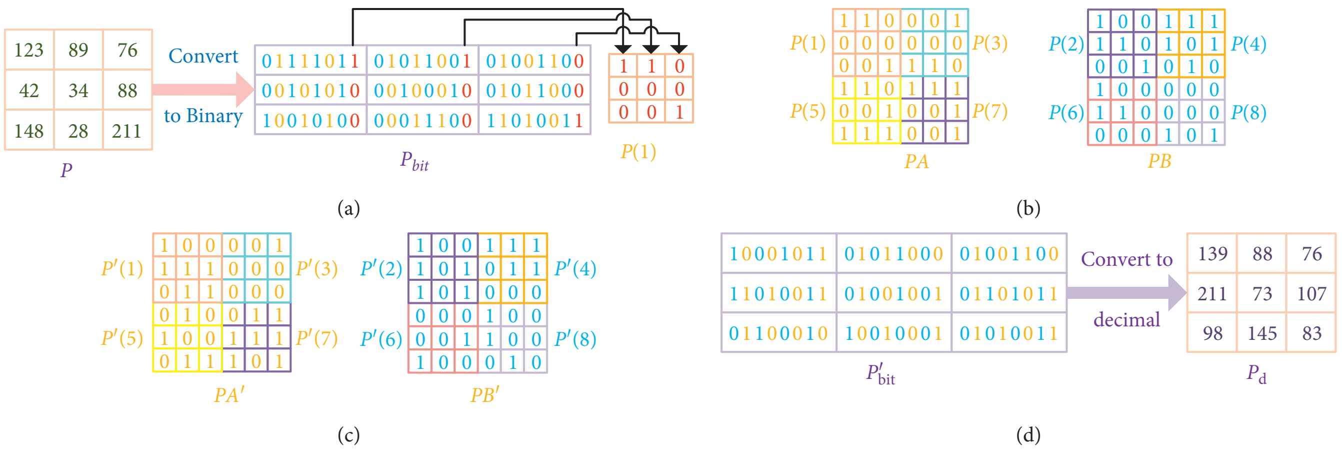

Figure 5: The bit plane of the image matrix (P) is scrambled. (a) Decompose the bit plane of the image matrix (P) to obtain the bit plane matrix $\mathbf{P}_{\text{bit}}$, (b) the bit matrix PA and the bit matrix PB, (c) the bit matrix $PA\prime$ and the bit matrix $PB\prime$, and (d) the bit plane matrix P_{bit}' and image matrix P_d.

```
Input: Image matrix $\mathbf{P}$, $\mathbf{LH}_1$, and $\mathbf{LH}_2$.
Output: The image $\mathbf{P_d}$ after scrambling.
$\mathbf{PA}$ = [bitget ($\mathbf{P}$, 1), bitget ($\mathbf{P}$, 3); bitget ($\mathbf{P}$, 5), bitget ($\mathbf{P}$, 7)]
$\mathbf{PB}$ = [bitget ($\mathbf{P}$, 2), bitget ($\mathbf{P}$, 4); bitget ($\mathbf{P}$, 6), bitget ($\mathbf{P}$, 8)]
for $i = 1: N$
    $\mathbf{A} = \mathbf{PA}\ (i, :)$
    $\mathbf{B} = \mathbf{PB}\ (:, i)$
    $\mathbf{PA}\prime\ (i, :) = \mathbf{A}\ (\mathbf{L_{h1}}\ (i, :))$
    $\mathbf{PB}\prime\ (:, i) = \mathbf{B}\ (\mathbf{L_{h2}}\ (:, i))$
end
for $i = 0:1:3$ do
    $\mathbf{P_d}\ (i) = \mathbf{PA}\prime\ (i + 1: N \times (i + 1))$
    $\mathbf{P_d}\ (2i + 1) = \mathbf{PB}\prime\ (i + 1: N \times (i + 1))$
end
```

Algorithm 3: Bit scrambling $\mathbf{P_d}$ = diffusion ($\mathbf{P}$, $\mathbf{LH}_1$, $\mathbf{LH}_2$).

4. Experimental Results and Performance Analysis

4.1. Keyspace. The key is the most critical part of the encryption scheme. The larger the keyspace, the stronger the ability to resist brute force attacks, so the key should have enough space to resist brute force attacks. In theory, when the keyspace reaches 2^{100}, it is enough to resist the brute force attacks that currently exist. The algorithm proposed has four parameters x_0, y_0, z_0, and w_0. The calculation accuracy of these four parameters is 10^{-15}, and the keyspace can reach $10^{60} \approx 2^{190}$, which is much larger than 2^{100}. Therefore, the keyspace of the LSRS algorithm is large enough to protect the security of the image.

4.2. Differential Attack Analysis. Differential attack refers to studying the influence of the difference between plain images on their cipher image and establishing a relationship between the plain image and its corresponding cipher image, thereby cracking the encryption method. The number of Pixel Change Rate (NPCR) and Unified Pixel Average Change Intensity (UACI) are two methods to test whether the encryption method can resist differential attacks [23]. NPCR reflects the ratio of the number of unequal pixels in the same position of two images to the number of all pixels in the image. UACI is the overall average change density, which represents the average change intensity of the planar image. The ideal values of NPCR and UACI are, respectively, 99.6094% and 33.4635%. Assuming that $\mathbf{P}_1$ and $\mathbf{P}_2$ are two cipher images and their plain images only have a one-bit difference, their NPCR and UACI values are calculated as shown in the following formula:

$$\begin{cases} \text{NPCR} = \dfrac{\sum_{i=1}^{M}\sum_{j=1}^{N} D_{(i,j)}}{M \times N} \times 100\%, \\ \\ \text{UACI} = \dfrac{\sum_{i=1}^{M}\sum_{j=1}^{N} \left| P_1(i,j) - P_2(i,j) \right|}{255 \times M \times N} \times 100\%, \end{cases} \tag{6}$$

where $\mathbf{P}_1\ (i, j) \neq \mathbf{P}_2\ (i, j)$, $D\ (i, j) = 1$; otherwise, $D\ (i, j) = 1$.

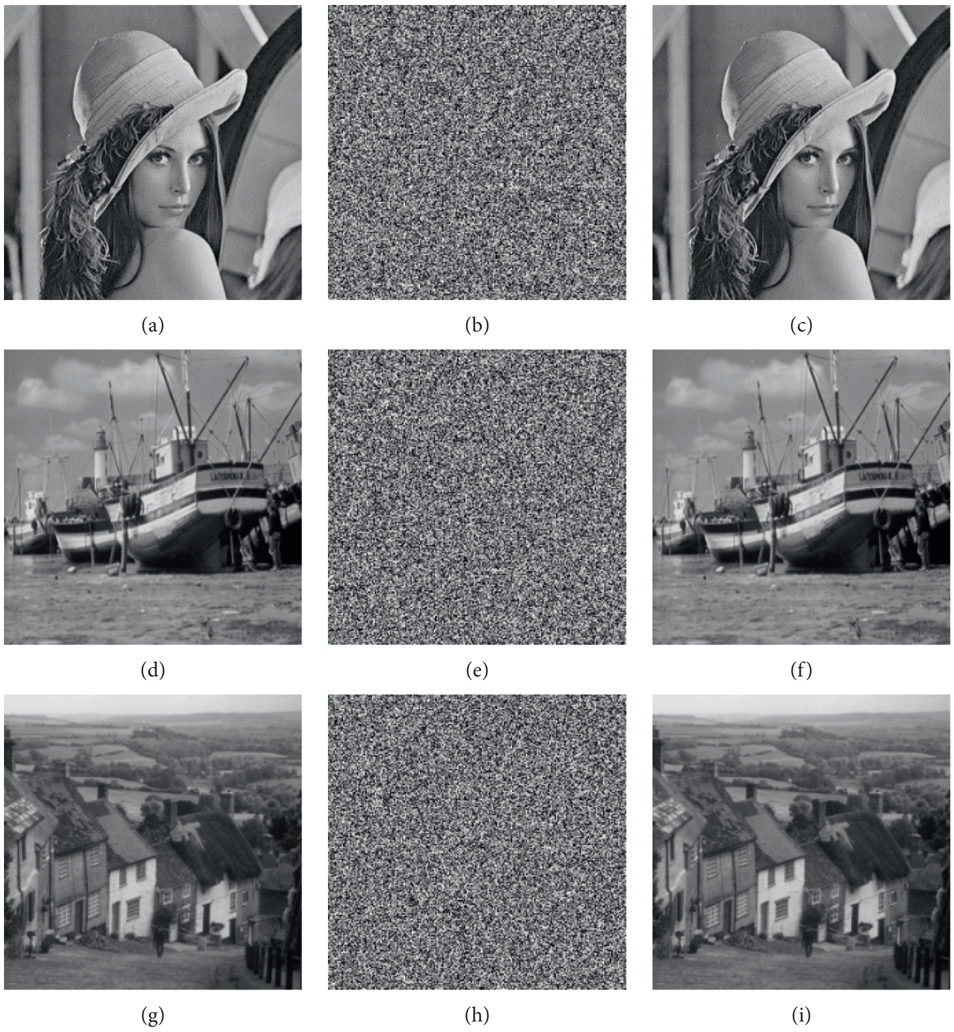

FIGURE 6: Plain image, cipher image, and decrypted image. (a) Plain image of Lena. (b) Cipher image of Lena. (c) Decrypted image of Lena. (d) Plain image of Boat. (e) Cipher image of Boat. (f) Decrypted image of Boat. (g) Plain image of Hill. (h) Cipher image of Hill. (i) Decrypted image of Hill.

Taking Lena image as an example, Table 1 compares the test results of NPCR and UACI under different algorithms. It can be seen that the encryption method we propose can resist differential attacks more effectively.

4.3. Key Sensitivity Analysis. To ensure the security of the encryption algorithm, the encryption algorithm should be highly sensitive to the input key. When the wrong key is entered to decrypt the cipher image, the output is an image that cannot identify any information. The LSDR algorithm proposed in this paper uses the Lena image as a test. First, input the image into the LSDR algorithm to obtain the keys $= [x_0, y_0, z_0, w_0]$ and the cipher image. The element x_0 in the keys is slightly modified by adding 1 to the 15th digit after the decimal point and then using the modified keys$\prime = [x_0 + 10^{-15}, y_0, z_0, w_0]$ to decrypt. Figure 7 shows the decrypted image under the correct key and the decrypted image obtained after modifying the key parameters x_0, y_0, z_0, and w_0, respectively. It can be seen that even if the decryption key changes slightly, the correct decrypted image cannot be obtained. Therefore, the key sensitivity in the LSDR algorithm is high enough to resist all types of brute force attacks.

4.4. Histogram Analysis. The histogram can intuitively reveal the distribution of pixel values in the image. Cipher image has a unified histogram, which can effectively resist statistical analysis, making it difficult for an attacker to obtain valuable information. The more even the pixel distribution in the cipher image, the more ideal the encryption algorithm. As shown in Figure 8, the plain image is input to the LSRS algorithm for encryption, and the histogram of the cipher image obtained is evenly distributed. In addition, the flatness of the histogram can be quantified numerically. The standard method is the chi-square test [24], which is defined as

$$X^2 = \sum_{0}^{255} \frac{(V_i - V_0)^2}{V_0}, \tag{7}$$

where $V_0 = ((M \times N)/256)$, $M \times N$ is the size of the image, and V_i and V_o, respectively, represent the actual and expected frequency of each gray level. Set the significance level as $\alpha = 0.05$. If X^2_{test} is less than $X^2_{\alpha} = 293.25$ [25], the histogram can be considered to be uniformly distributed. In this paper, test the three images of Lena, Boat, and Hill. From the results in Table 2, it can be seen that the proposed method is

Table 1: The Lena image test results in different schemes.

Index (%)	Ideal value	Ours	[20]	[21]	[22]
NPCR	99.6094	99.6101	99.65	99.61	99.66
UACI	33.4635	33.4583	33.56	33.48	33.49

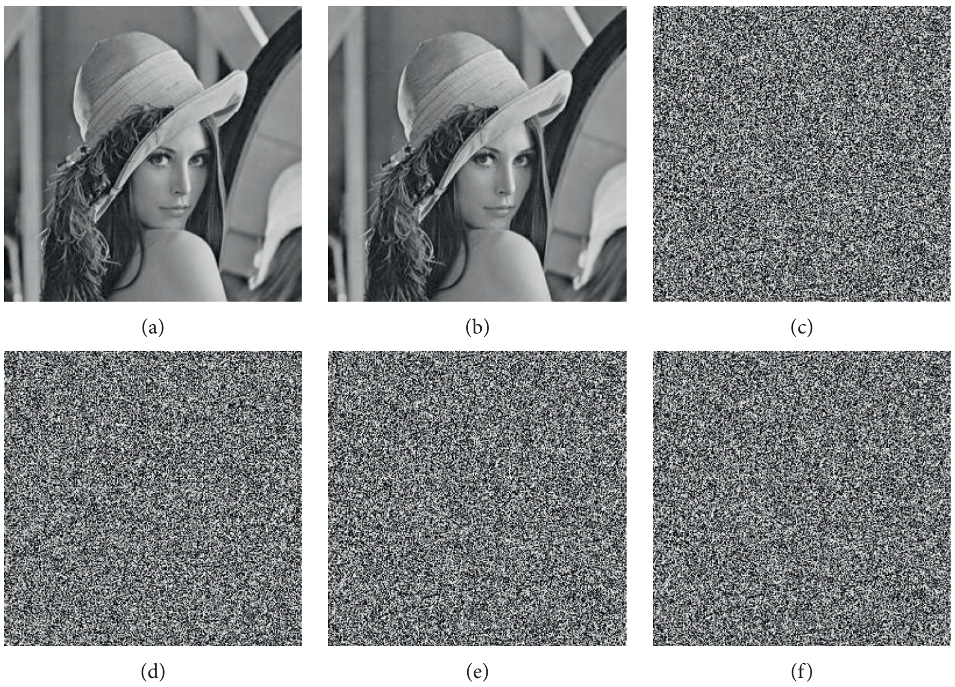

Figure 7: Plain image of Lena and the decrypted image after different parameters of the decryption key are changed: (a) plain image of Lena, (b) correct keys, (c) $x_0 + 10^{-15}$, (d) $y_0 + 10^{-15}$, (e) $z_0 + 10^{-15}$, and (f) $w_0 + 10^{-15}$.

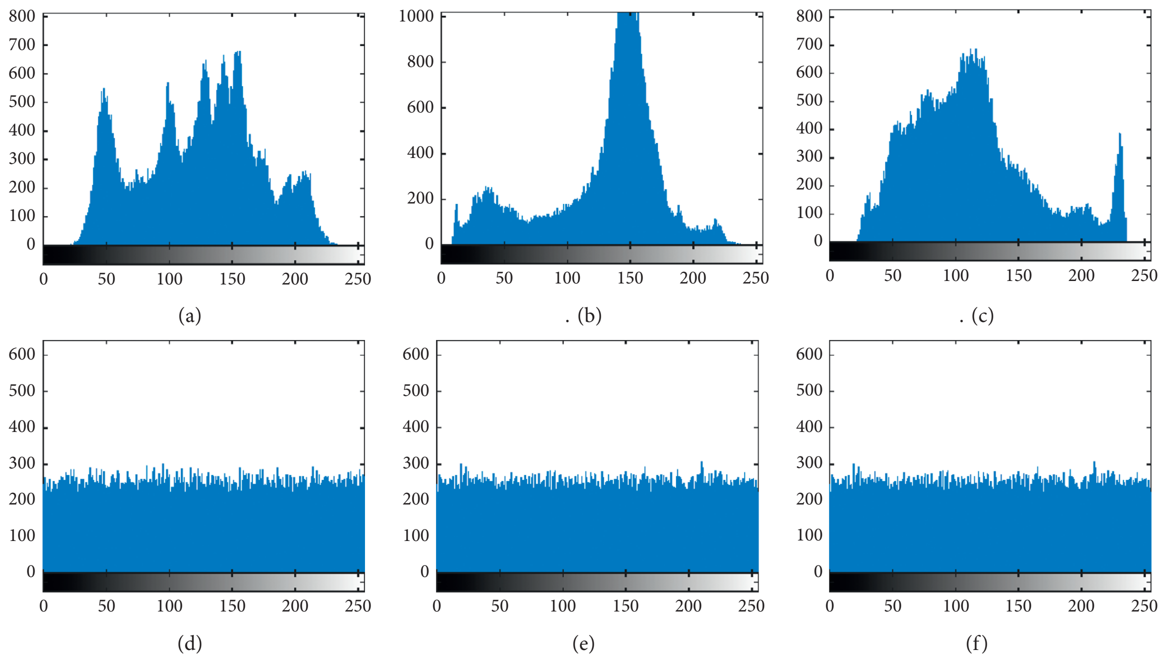

Figure 8: The histogram of the plain image and the corresponding cipher image. (a) The histogram of the Lena plain image. (b) The histogram of the Boat plain image. (c) The histogram of the Hill plain image. (d) The histogram of the Lena cipher image. (e) The histogram of the Boat cipher image. (f) The histogram of the Hill cipher image.

Table 2: $\mathbf{x}^2$ test results.

Image	Lena	Boat	Hill	Peppers
$x^2_{0.05}$	293.25	293.25	293.25	293.25
x^2_{test}	257.3516	237.9063	222.5469	270.8047
Decision	Pass	Pass	Pass	Pass

lower than the theoretical value of 293.25. Therefore, it can be considered that the program passed the chi-square test.

4.5. Correlation Analysis. The adjacent pixels of the plain image have a high correlation in the horizontal, vertical, and diagonal directions. The ideal encryption algorithm can reduce the correlation of adjacent pixels in the cipher image, thereby effectively resisting statistical attacks. The correlation coefficient calculation formula is

$$\begin{cases} E(x) = \dfrac{1}{N}\sum\limits_{i=1}^{N} x_i, \\ D(x) = \dfrac{1}{N}\sum\limits_{i=1}^{N} \left(x_i - E(x)\right)^2, \\ \operatorname{cov}(x, y) = \dfrac{1}{N}\sum\limits_{i=1}^{N} \left(\left(x_i - E(x)\right)\left(y_i - E(y)\right)\right), \\ \rho_{xy} = \dfrac{\operatorname{cov}(x, y)}{\sqrt{D_x} \times \sqrt{D_y}}, \end{cases} \tag{8}$$

where x and y are pixel values, cov (x, y) is the covariance, D (x) is the variance, E (x) is the mean, and ρ_{xy} is the correlation coefficient.

To analyze the correlation between adjacent pixels in the plain image and the cipher image, take Lena image as an example, randomly select 2000 pairs of adjacent pixels in the plain image and the cipher image to test. As shown in Table 3, the distribution of adjacent pixels in the plain image of Lena is highly concentrated, so the correlation between adjacent pixels of the plain image is very high. The distribution of adjacent pixels in the cipher image of Lena is random, which means that after encryption, the correlation between adjacent pixels of the cipher image of Lena is low. By comparing with the other three encryption methods, the results of the LSDR algorithm are also satisfactory.

4.6. Global Shannon Entropy and Local Shannon Entropy. Global Shannon Entropy (GSE) is a disordered statistical measure that reflects the randomness of information. The calculation formula of GSE [26] is

$$H(m) = -\sum_{i=0}^{Q} P(m_i)\log_2 P(m_i), \tag{9}$$

where Q is the gray level of the image. For 8-bit grayscale images, $Q = 255$. m_i is the ith gray value on the image, and $P(m_i)$ is the probability of m_i. In an entirely randomly generated image, the ideal value of GSE is 8. The closer the GSE of the image is to 8, the more random the image information.

Local Shannon Entropy (LSE) is proposed by Wu [27] to measure the randomness of encrypted images. For image $\mathbf{P}$, randomly select k nonoverlapping image blocks $S_1, S_2, \ldots, S_k$ and T_B pixels, and LSE is defined as

$$\overline{H_{k,T_B}}(\mathbf{P}) = \sum_{i=1}^{k} \frac{H(S_i)}{k}, \tag{10}$$

where H (S_i) is the Shannon entropy of the image block S_i. This article selects $(k, T_B) = (30, 1936)$ to test the cipher image. If the value of LSE is in the interval $[h_{\text{left}}^{l*\alpha}, h_{\text{right}}^{l*\alpha}]$, it means that the randomness is passed inspection; it can be considered that the cipher image has high randomness. Tested by encrypting the images in the USC-S IPI image database, the test results are shown in Table 4. It can be seen from Table 4 that after LSRS algorithm encryption, the GSE of the cipher image is very close to the ideal value, and most of the cipher images have passed the LSE critical value test. It can be considered that the generated cipher images have high randomness.

4.7. Antinoise Attack Analysis and Cropping Attack Analysis. Digital images may lose data or be disturbed by noise due to various reasons during the transmission process. An effective image encryption method can reconstruct an identifiable decrypted image in the case of noise interference or data loss. Add 1%, 5%, and 10% salt and pepper noise to cipher image of Lena, and then decrypt it. As shown in Figure 9, even if a certain salt and pepper noise is added to the cipher image, the information of the decrypted image can still be identified. As shown in Figure 10, cipher images of Lena images are cropped 1/64, 1/16, and 1/4, respectively, and then decrypted, which can identify the information of the decrypted image, so the algorithm can resist the analysis of cropping attacks.

4.8. Efficiency Analysis. The efficiency of image encryption is also one of the important indicators to measure the quality of image encryption methods. The main time-consuming parts of the LSRS algorithm are the iteration of the chaotic sequence, Latin square scrambling, Latin square replacement, and Latin square diffusion. In the simulation, take Lena as an example, we compared the encryption time of four different algorithms. Table 5 lists the running time of each algorithm, and the simulation result shows that the encryption efficiency of the LSRS algorithm is higher.

Table 3: Correlation analysis.

	Horizontal (%)	Vertical (%)	Diagonal (%)
Plain image Lena	0.9618	0.9854	0.9618
Our method	0.0023	0.0158	0.0147
Ref. [20]	−0.0226	0.0041	0.0368
Ref. [21]	−0.0059	−0.0146	0.0211
Ref. [22]	0.0220	0.01792	$7E{-}06$

Table 4: GSE and LSE test analysis.

Image	GSE	LSE	LSE critical value $k = 30$, $T_B^{Q=256^*} = 1936$		
			$h_{\text{left}}^{l*0.001} = 7.9015$ $h_{\text{right}}^{l*0.001} = 7.9034$ 0.001-level	$h_{\text{left}}^{l*0.01} = 7.9017$ $h_{\text{right}}^{l*0.01} = 7.9032$ 0.01-level	$h_{\text{left}}^{l*0.05} = 7.9019$ $h_{\text{right}}^{l*0.05} = 7.9030$ 0.05-level
5.1.09	7.9976	7.9027	Pass	Pass	Pass
5.1.10	7.9973	7.9022	Pass	Pass	Pass
5.1.11	7.9976	7.9030	Pass	Pass	Pass
5.1.12	7.9975	7.9033	Pass	Pass	Pass
5.1.13	7.9972	7.9021	Pass	Pass	Pass
5.1.14	7.9970	7.9024	Pass	Pass	Pass
5.2.08	7.9992	7.9034	Pass	No	No
5.2.09	7.9994	7.9035	No	No	No
5.2.10	7.9992	7.9028	Pass	Pass	Pass
7.1.01	7.9993	7.9032	Pass	Pass	No
7.1.02	7.9994	7.9019	Pass	Pass	Pass
7.1.03	7.9993	7.9036	No	No	No
7.1.04	7.9993	7.9020	Pass	Pass	Pass
7.1.05	7.9992	7.9054	No	No	No
7.1.06	7.9993	7.9020	Pass	Pass	Pass
7.1.07	7.9993	7.9025	Pass	Pass	Pass
7.1.08	7.9994	7.9028	Pass	Pass	Pass
7.1.09	7.9993	7.9049	No	No	No
7.1.10	7.9993	7.9030	Pass	Pass	Pass
7.2.01	7.9998	7.9020	Pass	Pass	Pass

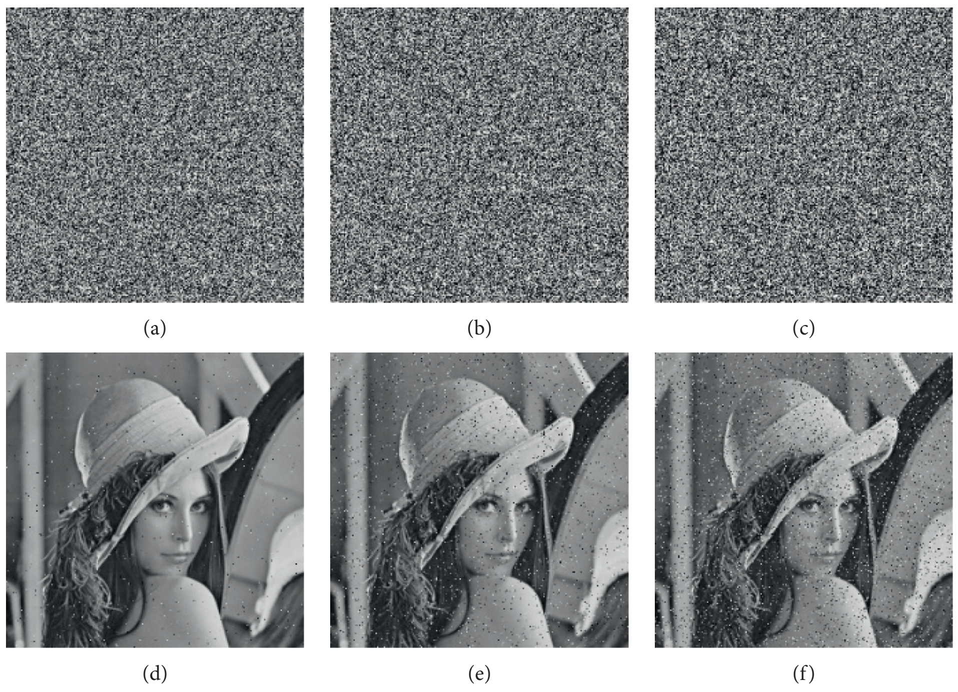

Figure 9: The encrypted image and the decrypted image after salt and pepper noise attack. (a) 1% attack encrypted image, (b) 5% attack encrypted image, (c) 10% attack encrypted image, (d) 1% attack decrypted image, (e) 5% attack decrypted image, and (f) 10% attack decrypted image.

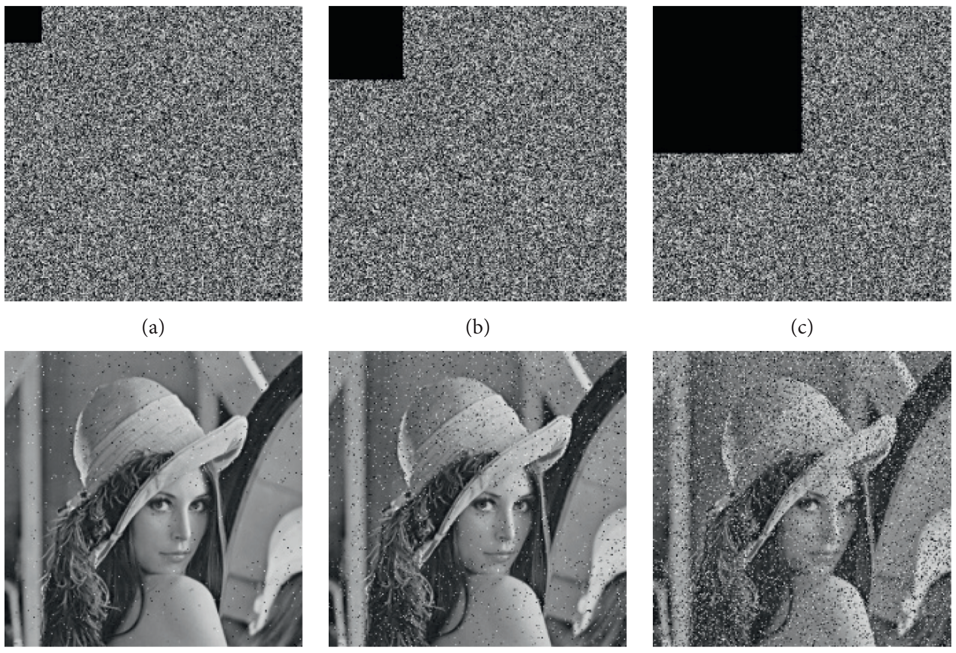

Figure 10: The cropped encrypted image and decrypted image. (a) The 1/64 cropping encrypted image, (b) the 1/16 cropping encrypted the image, (c) the 1/4 cropping the encrypted image, (d) the 1/64 cropping decrypted image, (e) the 1/16 cropping decrypted image, and (f) the 1/4 cropping decrypted image.

Table 5: Running time of four different algorithms.

Algorithm	Ours	[20]	[21]	[22]
Time (second)	0.325	0.613	0.3243	0.425

5. Conclusion

This paper proposes a novel chaotic image encryption algorithm based on Latin square and random shift. The LSRS algorithm adopts the structure of pixel scrambling-replacement-bit scrambling. The generation of Latin squares in the encryption process is related to the chaotic sequence, which improves the security of the entire encryption system. Since the histogram of the Latin square is evenly distributed, use the elements in the Latin square lookup table to replace pixels, and the algorithm can effectively resist differential attacks. Each cipher image corresponds to a Latin square lookup table, which increases the difficulty of the algorithm to decipher. The simulation results prove the safety and effectiveness of the LSRS algorithm.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China, under Grants 62072417, 62102374, 62073299, and U1804262 and the Key Research and Development Program of Henan Province, under Grants 212102210028 and 202102210177.

References

[1] C. Li, Y. Zhang, and E. Y. Xie, "When an attacker meets a cipher-image in 2018: a year in review," *Journal of Information Security and Applications*, vol. 48, Article ID 102361, 2019.
[2] Q. Liu, P.-Y. Li, M.-C. Zhang, Y.-X. Sui, and H.-J. Yang, "A novel image encryption algorithm based on chaos maps with Markov properties," *Communications in Nonlinear Science and Numerical Simulation*, vol. 20, no. 2, pp. 506–515, 2015.
[3] A.V. Diaconu, "Circular inter-intra pixels bit-level permutation and chaos-based image encryption," *Information Sciences*, vol. 355-366, pp. 314–327, 2015.
[4] Z. Hua, Y. Zhou, C.-M. Pun, and C. L. P. Chen, "2D sine logistic modulation map for image encryption," *Information Sciences*, vol. 297, pp. 80–94, 2015.
[5] Y. Wu, Y. Zhou, S. Agaian, and J. P. Noonan, "2D sudoku associated bijections for image scrambling," *Information Sciences*, vol. 327, pp. 91–109, 2016.
[6] X. Y. Wang, Y. Li, and J. Jin, "A new one-dimensional chaotic system with applications in image encryption," *Chaos, Solitons & Fractals*, vol. 139, 2020.
[7] Y. Zhou, L. Bao, and C. L. P. Chen, "A new 1D chaotic system for image encryption," *Signal Processing*, vol. 97, pp. 172–182, 2014.

[8] Y.-Q. Zhang and X.-Y. Wang, "A new image encryption algorithm based on non-adjacent coupled map lattices," *Applied Soft Computing*, vol. 26, pp. 10–20, 2015.

[9] Y. Liu, X. Tong, and J. Ma, "Image encryption algorithm based on hyper-chaotic system and dynamic S-box," *Multimedia Tools and Applications*, vol. 75, no. 13, pp. 7739–7759, 2016.

[10] Z. P. Peng, C. H. Wang, L. Yuan, and X.-W. Luo, "A novel four-dimensional multi-wing hyper-chaotic attractor and its application in image encryption," *Acta Physica Sinica*, vol. 63, no. 24, Article ID 240506, 2014.

[11] Z.-H. Gan, X.-L. Chai, D.-J. Han, and Y.-R. Chen, "A chaotic image encryption algorithm based on 3-D bit-plane permutation," *Neural Computing and Applications*, vol. 31, no. 11, pp. 7111–7130, 2019.

[12] W. Zhang, H. Yu, Y.-L. Zhao, and Z.-L. Zhu, "Image encryption based on three-dimensional bit matrix permutation," *Signal Processing*, vol. 118, pp. 36–50, 2016.

[13] C. E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949.

[14] Y. Wu, Y. C. Zhou, J. P. Noonan, and S. Agaian, "Design of image cipher using latin squares," *Information Sciences*, vol. 264, pp. 317–339, 2014.

[15] H. T. Panduranga, S. K. N. Kumar, and S. K. Kiran, "Image encryption based on permutation-substitution using chaotic map and latin square image cipher," *The European Physical Journal Special Topics*, vol. 223, no. 8, pp. 1663–1677, 2014.

[16] M. Ahmad and F. Ahmad, "Cryptanalysis of image encryption based on permutation-substitution using chaotic map and latin square image cipher," *Advances in Intelligent Systems and Computing*, Springer International Publishing, vol. 327, pp. 481–488, 2015.

[17] S. Moon, J.-J. Baik, and J. M. Seo, "Chaos synchronization in generalized Lorenz systems and an application to image encryption," *Communications in Nonlinear Science and Numerical Simulation*, vol. 96, Article ID 105708, 2021.

[18] Y. Wu, J. P. Noonan, and S. Agaian, "Dynamic and implicit latin square doubly stochastic S-boxes with reversibility," in *Proceedings of the 2011 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 3358–3364, Anchorage, AK, USA, October 2011.

[19] Z. Tang, J. Song, X. Zhang, and R. Sun, "Multiple-image encryption with bit-plane decomposition and chaotic map," *Optics and Lasers in Engineering*, vol. 80, pp. 1–11, 2016.

[20] L. Xu, X. Gou, Z. Li, and J. Li, "A novel chaotic image encryption algorithm using block scrambling and dynamic index based diffusion," *Optics and Lasers in Engineering*, vol. 91, pp. 41–52, 2017.

[21] C. Cao, K. Sun, and W. Liu, "A novel bit-level image encryption algorithm based on 2D-LICM hyperchaotic map," *Signal Processing*, vol. 143, pp. 122–133, 2018.

[22] X. Ming and Z. Tian, "A novel image encryption algorithm based on self-orthogonal Latin squares," *Optik*, vol. 17, pp. 891–903, 2018.

[23] Y. Wu, J. P. Noonan, and S. Agaian, "NPCR and UACI randomness tests for image encryption," *Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT)*, vol. 1, no. 2, pp. 31–38, 2011.

[24] S. Ma, Y. Zhang, Z. Yang, J. Hu, and X. Lei, "A new plaintext-related image encryption scheme based on chaotic sequence," *IEEE Access*, vol. 7, pp. 30344–30360, 2019.

[25] A. Belazi, M. Talha, S. Kharbech, and W. Xiang, "Novel medical image encryption scheme based on chaos and DNA encoding," *IEEE Access*, vol. 7, pp. 36667–36681, 2019.

[26] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[27] Y. Wu, Y. Zhou, G. Saveriades, S. Agaian, J. P. Noonan, and P. Natarajan, "Local Shannon entropy measure with statistical tests for image randomness," *Information Sciences*, vol. 222, pp. 323–342, 2013.