entropy

MDPI

Article

An Image Encryption Algorithm Based on Improved Hilbert Curve Scrambling and Dynamic DNA Coding

Shengtao Geng, Jiahao Li, Xuncai Zhang * and Yanfeng Wang

School of Electrical and Information Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China; gst@zzuli.edu.cn (S.G.); 332101050056@email.zzuli.edu.cn (J.L.); wangyanfeng@zzuli.edu.cn (Y.W.)

* Correspondence: zhangxuncai@pku.edu.cn

Abstract: As an effective method for image security protection, image encryption is widely used in data hiding and content protection. This paper proposes an image encryption algorithm based on an improved Hilbert curve with DNA coding. Firstly, the discrete wavelet transform (DWT) decomposes the plaintext image by three-level DWT to obtain the high-frequency and low-frequency components. Secondly, different modes of the Hilbert curve are selected to scramble the high-frequency and low-frequency components. Then, the high-frequency and low-frequency components are reconstructed separately using the inverse discrete wavelet transform (IDWT). Then, the bit matrix of the image pixels is scrambled, changing the pixel value while changing the pixel position and weakening the strong correlation between adjacent pixels to a more significant correlation. Finally, combining dynamic DNA coding and ciphertext feedback to diffuse the pixel values improves the encryption effect. The encryption algorithm performs the scrambling and diffusion in alternating transformations of space, frequency, and spatial domains, breaking the limitations of conventional scrambling. The experimental simulation results and security analysis show that the encryption algorithm can effectively resist statistical attacks and differential attacks with good security and robustness.

Keywords: image encryption; DWT; Hilbert curve; bit-level scramble; DNA coding; ciphertext feedback

Citation: Geng, S.; Li, J.; Zhang, X.; Wang, Y. An Image Encryption Algorithm Based on Improved Hilbert Curve Scrambling and Dynamic DNA Coding. *Entropy* **2023**, *25*, 1178. https://doi.org/10.3390/e25081178

Academic Editor: Congxu Zhu

Received: 30 June 2023
Revised: 30 July 2023
Accepted: 5 August 2023
Published: 8 August 2023

1. Introduction

With the fast development of the internet and multimedia technology, information security is gaining more and more attention. To prevent images from being stolen during transmission, researchers have proposed many methods for image protection, and image encryption is a common method for securing image transmission. Early encryption methods are mainly data encryption. With continuous research, researchers have proposed new encryption methods that address the drawbacks of early encryption algorithms, namely low operational efficiency, small key space, and poor security [1,2].

In 1963, Lorentz [3] introduced the concept of chaos theory, which is widely used in image encryption due to the sensitivity, ergodicity, and unpredictability of chaotic systems to initial states and control parameters [4–7]. Xu [8] proposed a new image encryption algorithm based on one-dimensional logistic mapping and the orthogonal Latin square, improving ciphertext image security. The advantages of one-dimensional logistic mapping are its simple structure, high computational efficiency, and lower level of difficulty in implementation. However, its disadvantages are the short period window, limited range of chaotic behavior, small generated key space, and vulnerability to attacks [9]. To address the insufficiency of low-dimensional chaotic systems in image encryption, Gao [10] combined two one-dimensional chaotic systems and proposed a new two-dimensional chaotic system, which uses the chaotic sequence generated by the two-dimensional chaotic system to displace the row and column pixels and then performs nonlinear diffusion of the pixels, which improves the randomness of the chaotic sequence and also enhances the resistance

of the encryption algorithm to attacks. Arthi [11] added a state variable to the three-dimensional Lorenz chaotic system and constructed a four-dimensional hyperchaotic system containing two positive Lyapunov exponents. It has the advantage of iterating once to obtain multiple chaotic sequences, which is more efficient, and the system parameters can also make the key space larger and effectively resist brute force attacks. Using iterative chaotic sequences for image encryption improves the complexity and robustness of the encryption algorithm.

In image encryption, scrambling and diffusion techniques are the core part of the encryption algorithm [12]. Scramble changes the position of pixels and reduces the correlation between adjacent pixels, while diffusion randomly changes the pixel values, making the ciphertext image more chaotic. Researchers have proposed many image encryption algorithms based on scrambling and diffusion techniques, most of which perform scrambling followed by diffusion [13–15]. Although this encryption algorithm has good security, there are some problems. For example, in [16], the scrambling part uses only the Hilbert fill curve to scramble pixels, which does not entirely break the correlation between adjacent pixels, making it less effective and more vulnerable to brute force attacks. The single scrambling and diffusion operations are too simple, resulting in a less secure encryption algorithm [17]. In contrast, multiple scrambling and diffusion repetitions are time-consuming and significantly reduce the encryption efficiency [18]. To address the above shortcomings, researchers have combined scrambling and diffusion and proposed bit-level scrambling with simultaneous scrambling and diffusion to encrypt images [19–22]. The bit-level scrambling divides the pixel value into eight bits, disrupting the bit positions to achieve the simultaneous scrambling and diffusion of pixels. Xiang [23] proposed an image encryption algorithm that encrypts only the upper four bits of the image pixel value, which improves the encryption performance and reduces the encryption time by half. Li [24] proposed a bit-level scrambling method based on the binary tree, simultaneously changing the pixel position and value. Wang [25] used bit cyclic displacement in the scrambling phase, and the algorithm performs well through security and performance analysis.

The scrambling algorithm disrupts the pixels' position and eliminates the correlation between them; thus, to better conceal the key information of the image, further diffusion of the pixel values is necessary. DNA coding has received great attention from more researchers because of its low power consumption, high density, and parallelism. DNA coding was first proposed by Clelland [26] in cryptography, and since then, cryptographic algorithms combining DNA coding with chaotic systems have emerged [27–30]. Other diffusion algorithms that have been proposed are the matrix half-tensor product [31], the Feistel-like network [32], filtered convolution [33], etc. In [29], Jithin divides the color image into three planes of RGB, converts these three planes into DNA base planes using fixed encoding rules, and performs heteroskedastic operations with these three DNA base planes using DNA matrices generated from chaotic sequences. In [30], Wang uses different encoding rules to convert multiple plaintext images into multiple DNA matrices. Nevertheless, each pixel has the same encoding rules for the same plaintext image matrix, performs operations with the chaotic sequence-generated DNA matrix, and uses different decoding rules. However, these DNA coding-based image encryption algorithms achieve pixel diffusion for their purposes; they also have a significant drawback, as the fixed DNA coding and decoding rules cannot change the bit distribution of pixels and are vulnerable to brute force attacks [34].

Image encryption methods have frequency domain encryption in addition to spatial domain encryption. In [35], Shafique uses multiple S-boxes combined with wavelet transform to encrypt images, which shortens the encryption time and solves the problem of a weak single S-box encryption. In [36], Yan used fractional-order wavelet transform to perform third-order fractional wavelet transform on plaintext images to obtain high-frequency and low-frequency components, index scrambling for each component using an index sequence generated by the chaotic sequence, and finally, diffusing the scrambled image using a cyclic shift. The resulting ciphertext image has good robustness. In [37],

Qin uses dynamic wavelet decomposition and scrambling diffusion simultaneously to combine spatial-domain and frequency-domain encryption, ensuring both the security and robustness of the encryption algorithm.

This paper proposes an image encryption algorithm based on improved Hilbert curve scrambling and dynamic DNA coding by combining a 4D hyperchaotic system to summarize the above. Firstly, the hash value of the plaintext image is obtained using the SHA-384 algorithm, and the initial value of the hyperchaotic system is calculated. Secondly, the decomposition of the plaintext image is achieved using three-level DWT to obtain one low-frequency component and nine high-frequency components. These ten components are scrambled using different modes of the Hilbert curve, and the high-frequency and low-frequency components are then reconstructed using IDWT. Then, the bit matrix of the image pixels is position-scrambled to enhance the scrambling effect. Finally, the pixel values are further diffused using dynamic DNA coding and ciphertext feedback to improve the security of the encryption algorithm.

The rest of this paper is as follows: Section 2 introduces the 4D hyperchaos system, DWT, and the Hilbert curve; Section 3 presents the proposed encryption algorithm; Section 4 shows the experimental simulation results; Section 5 is an analysis of the various security of encryption algorithm; and Section 6 gives the conclusion.

2. Preparation

2.1. D Hyperchaotic System

Li introduced a nonlinear controller W into the chaotic system, which constitutes a 4D hyperchaotic system [38]; this is a nonlinear four-dimensional chaotic system that is reversible and discrete and can simultaneously generate four chaotic sequences with more complex behaviors and increased key space, enabling the encryption algorithm to effectively resist various attacks such as known plaintext attacks and brute force attacks, compensating for the small key space of the low-dimensional chaotic system. The expression of the 4D hyperchaotic system is shown in Equation (1).

$$\begin{cases} \dot{x} = \delta(y - x) \\ \dot{y} = -xz + \tau x + \mu y - w \\ \dot{z} = xy - \theta z \\ \dot{w} = x + \varphi \end{cases} \tag{1}$$

where δ, θ, μ, τ, and φ are the parameters that affect the behavior of the hyperchaotic system, and $\varphi \in [-0.7, 0.7]$. When $\delta = 36$, $\theta = 3$, $\mu = 28$, $\tau = 16$, and $\varphi = 0.2$, the Lyapunov exponents $\lambda_1 = 1.552$, $\lambda_2 = 0.023$, $\lambda_3 = 0$, and $\lambda_4 = -12.573$ of the hyperchaotic system, which contains two positive Lyapunov exponents, has better chaotic behavior. The computation time is somewhat shorter than the usual chaotic system.

Based on the above parameters, Figure 1 shows the phase diagrams of the hyperchaotic system in two and three dimensions after discretization by the 4th-order Runge–Kutta method. The phase diagram in each dimension indicates that the system has multiple attractors and is a hyperchaotic system with complex variations. In this paper, the parameters $\delta = 36$, $\theta = 3$, $\mu = 28$, and $\tau = 16$ are taken to iterate the equations of the hyperchaotic system to obtain four sequences for image encryption.

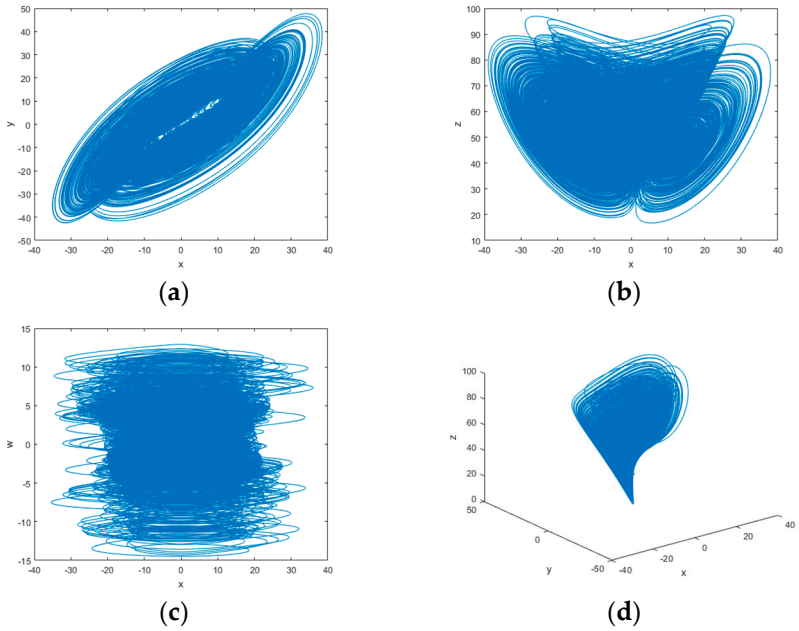


Figure 1. Phase diagram of the 4D hyperchaotic system: (**a**) x–y space; (**b**) x–z space; (**c**) x–w space; (**d**) x–y–z space.

2.2. Discrete Wavelet Transform

The DWT is a discretization of the scales and translations of the fundamental wavelet that can decompose the signal at different scales and decompose the signal into components of different frequencies. 2D-DWT is defined by Equation (2).

$$\begin{cases} T_{\Phi}(j_0,m,n) = \frac{1}{M\times N}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} g(x,y)\Phi_{j_0,m,n}(x,y) \\ T_{\Psi}^{i}(j,m,n) = \frac{1}{M\times N}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} g(x,y)\Psi_{j,m,n}^{i}(x,y) \end{cases} \tag{2}$$

where $T_{\Phi}(j_0,m,n)$ denotes the approximate part of the image, $T_{\Psi}^{i}(j,m,n)$ denotes the edge section of the image, $\Phi_{j_0,m,n}(x,y)$ denotes the scaling function, and $\Psi_{j,m,n}^{i}(x,y)$ denotes the wavelet function. The 2D-IDWT is defined by Equation (3).

$$g(x,y) = \left(\begin{array}{l} \frac{1}{M\times N}\sum_m\sum_n T_{\Phi}(j_0,m,n)\Phi_{j_0,m,n}(x,y) \\ +\frac{1}{M\times N}\sum_{i=H,V,D}\sum_{j=j_0}^{\infty}\sum_m\sum_n T_{\Phi}(j_0,m,n)\Psi_{j,m,n}^{i}(x,y) \end{array} \right) \tag{3}$$

The advantage of DWT is that it eliminates the connection between pixels and is less distorted than the conventional discrete cosine transform (DCT) after multilevel wavelet decomposition. Figure 2 shows the DWT and IDWT process of the plaintext image. DWT decomposes the image to obtain the four components LL1, HL1, LH1, and HH1. IDWT is the column and row reconstruction of the resulting four components to obtain the original image [39]. Figure 3 shows the one-level, two-level, and three-level DWT. In the two-level DWT, the component LL1 continues to be decomposed into four components: LL2, HL2, LH2, and HH2. Similarly, in the three-level DWT, the component LL2 continues to be decomposed into four components: LL3, HL3, LH3, and HH3.

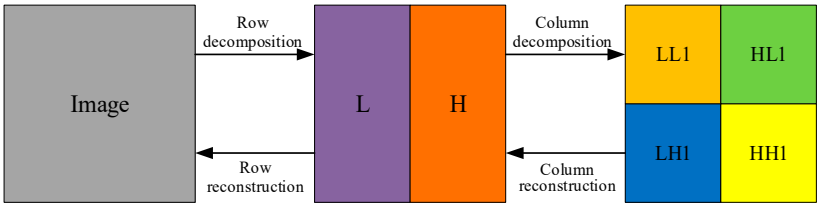


Figure 2. The DWT and IDWT process.

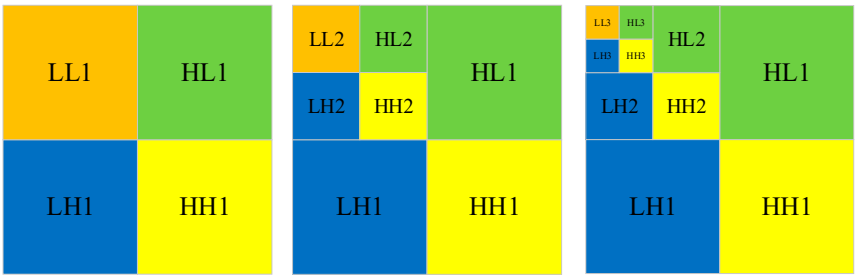


Figure 3. Schematic diagram of different levels of DWT: (**a**) One-level DWT; (**b**) Two-level DWT; (**c**) Three-level DWT.

2.3. Hilbert Curve

The Hilbert curve is one of the classical space-filling curves [40], and similar space-filling curves include the Z-curve [41], Gray codes [42], etc. A Hilbert curve can linearly traverse every pixel point in the two-dimensional plane and travels each pixel point only once, according to the properties of their own spatially filled curve. The 2D Hilbert curve is a square divided equally into four little squares. The first iteration is completed by starting from the center of the bottom left square up to the center of the top right square, then right to the center of the entire right square, and then down to the center of the bottom right square in turn. The image is divided into several 2×2 submatrices according to the size of the image, and each sub-matrix is traversed by a first-order Hilbert curve with different directions. Each sub-matrix is traversed by a One-order Hilbert curve in a different direction, and then the first and last pixel points of each sub-matrix are connected. The Hilbert curve can traverse the whole image. Figure 4 illustrates the different orders of the Hilbert curve. In this paper, eight different filling modes are designed, based on the four starting positions of the Hilbert curve and two directions: horizontal and vertical. Taking the Two-order Hilbert curve as an example, Figure 5 shows these eight modes.

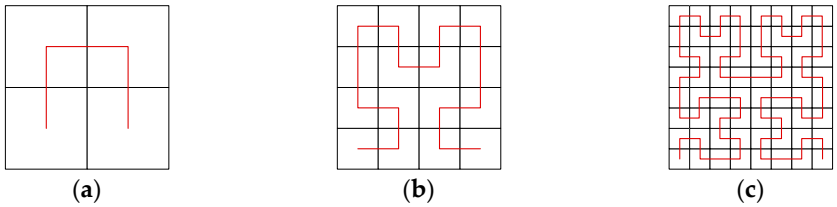


Figure 4. The Hilbert curve of different orders: (**a**) One-order Hilbert curve; (**b**) Two-order Hilbert curve; (**c**) Three-order Hilbert curve.

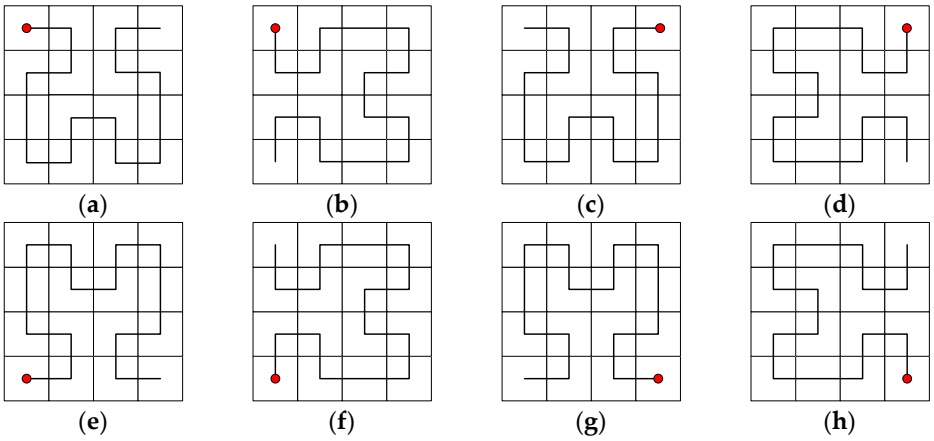


Figure 5. Schematic diagram of the eight modes of the Two-order Hilbert curve: (**a**) Mode1; (**b**) Mode2; (**c**) Mode3; (**d**) Mode4; (**e**) Mode5; (**f**) Mode6; (**g**) Mode7; (**h**) Mode8.

3. Encryption Algorithm

3.1. Secret Key Generation

The parameters of the chaotic system are calculated by the intermediate variable a_i to make the encryption algorithm dependent on the key. Assuming that the size of the plaintext image is $M \times N$, and $P_{i,j}$ is the pixel value of the plaintext, then $i \in [1, N]$ and $j \in [1, M]$, and the parameter φ of the hyperchaotic system is calculated by Equations (4) and (5).

$$\begin{cases} a_1 = flood\left(\sum_{i=1,j=1}^{i=\frac{M}{2},j=\frac{N}{2}} mod\left(P_{i,j}, 256\right) \times \frac{4}{M \times N}\right) \\ a_2 = flood\left(\sum_{i=1,j=\frac{N}{2}+1}^{i=\frac{M}{2},j=N} mod\left(P_{i,j}, 256\right) \times \frac{4}{M \times N}\right) \\ a_3 = flood\left(\sum_{i=\frac{M}{2}+1,j=1}^{i=M,j=\frac{N}{2}} mod\left(P_{i,j}, 256\right) \times \frac{4}{M \times N}\right) \\ a_4 = flood\left(\sum_{i=\frac{M}{2}+1,j=\frac{N}{2}+1}^{i=M,j=N} mod\left(P_{i,j}, 256\right) \times \frac{4}{M \times N}\right) \end{cases} \tag{4}$$

$$\varphi = \frac{1}{2}\sin\left(mod((a_1 - a_2 + a_3 - a_4), \frac{\pi}{2})\right) \tag{5}$$

where the intermediate variables $[a_1, a_2, a_3, a_4]$ are obtained by the calculation of Equation (4), *flood* is the downward rounding function, and the *mod* is the mod function.

The plaintext image is input into the SHA-384 algorithm, which outputs a 384-bit binary, H. The H binary is divided into 48 groups of binary sequences of 8 bits each, i.e., $H_k = k_1,\ k_2, k_3, \cdots k_{48}$, and the initial values x_0, y_0, z_0, w_0 of the hyperchaotic system are calculated by Equations (6) and (7). The generated parameters φ and the initial values x_0, y_0, z_0, and w_0 are substituted into the hyperchaotic system for 1000 + $4M \times N$ times, and the first 1000 times are rounded off to eliminate the transient effect and to obtain the four chaotic sequences SX, SY, SZ, and SW. The four chaotic sequences are processed using Equation (8) to obtain the sequences X, Y, Z, and W that are used in the encryption algorithm.

$$\begin{cases} x_0 = \frac{mod(Q_1+Q_2+Q_3+Q_4,256)}{256} \\ y_0 = \frac{mod(Q_3+Q_4+Q_5+Q_6,256)}{256} \\ z_0 = \frac{mod(Q_5+Q_6+Q_7+Q_8,256)}{256} \\ w_0 = \frac{mod(Q_2+Q_4+Q_6+Q_8,256)}{256} \end{cases} \tag{6}$$

$$Q_i = k_{6*i-5} \bigoplus k_{6*i-4} \bigoplus k_{6*i-3} \bigoplus k_{6*i-2} \bigoplus k_{6*i-1} \bigoplus k_{6*i} \tag{7}$$

where $[Q_1 \sim Q_8]$ is the intermediate variable, $1 \le i \le 8$, and $\oplus$ is the XOR operation.

$$\begin{cases} X(i) = mod\left(SX(i) \times 10^{12}, 256\right) \\ Y(i) = mod\left(SY(i) \times 10^{12}, 256\right) \\ Z(i) = mod\left(SZ(i) \times 10^{12}, 256\right) \\ W(i) = mod\left(SW(i) \times 10^{12}, 256\right) \end{cases} \quad 1 \le i \le 4M \times N \tag{8}$$

3.2. Pixel Scrambling

3.2.1. Hilbert Curve Scrambling

To completely disrupt the image pixel location distribution, a pixel-level scrambling method is proposed. DWT is used to decompose the plaintext image into four components—LL_1, HL_1, LH_1, and HH_1—and the results are shown in Equation (9).

$$DWT(P) = [LL_1, HL_1, LH_1, HH_1] \tag{9}$$

where P is the plaintext, and LL_1 is the low-frequency component. HL_1, LH_1, and HH_1 are the high-frequency components. To reduce the redundancy and encryption time, only the low-frequency components need to be scrambled with complex behavior. LL_1 is used as the new matrix for two-level DWT to obtain four components—LL_2, HL_2, LH_2, and

HH_2. Decomposition of LL_2 is continued for three-level DWT, and the result is shown in Equation (10).

$$\begin{cases} DWT(LL_1) = [LL_2, HL_2, LH_2, HH_2] \\ DWT(LL_2) = [LL_3, HL_3, LH_3, HH_3] \end{cases} \tag{10}$$

The plaintext image P is decomposed by three-level DWT to obtain the low-frequency component LL_3 and nine high-frequency components, HL_1, LH_1, HH_1, HL_2, LH_2, HH_2, HL_3, LH_3, and HH_3. The sequence FX is obtained by intercepting the first $3M$ elements of the sequence X, and then the sequence FX is processed by using Equation (11). The sequence DX is divided into three sub-sequences, DX_1, DX_2, DX_3, and the high-frequency components HL_1, LH_1, and HH_1 are scanned for scrambling with the sub-sequence DX_1 to select different modes of a$[(\log_2 M) - 1]$-order Hilbert curve. The sub-sequence DX_2 is used to select different modes of a $[(\log_2 M) - 2]$-order Hilbert curve to scan for scrambling of the high-frequency components HL_2, LH_2 and HH_2; then, the sub-sequence DX_3 is used to select different modes of a $[(\log_2 M) - 3]$-order Hilbert curve to scan for the scrambling of the components HL_3, LH_3, HH_3, and LL_3. The pixels on the scan path are arranged into two-dimensional matrices by rows to obtain the scrambled matrices Shl_1, Slh_1, Shh_1, Shl_2, Slh_2, Shh_2, Shl_3, Slh_3, Shh_3, and Sll_3. Finally, the ten scrambled matrices are reconstructed by Equation (12) to obtain the scrambled matrix P'. Assuming that the size of the low-frequency component LL_2 is 8×8, Figure 6 shows the Hilbert curve scrambling process.

$$DX = mod(FX, 8) + 1 \tag{11}$$

$$\begin{cases} Sll_2 = IDWT(Sll_3, Shl_3, Slh_3, Shh_3) \\ Sll_1 = IDWT(Sll_2, Shl_2, Slh_2, Shh_2) \\ P' = IDWT(Sll_1, Shl_1, Slh_1, Shh_1) \end{cases} \tag{12}$$

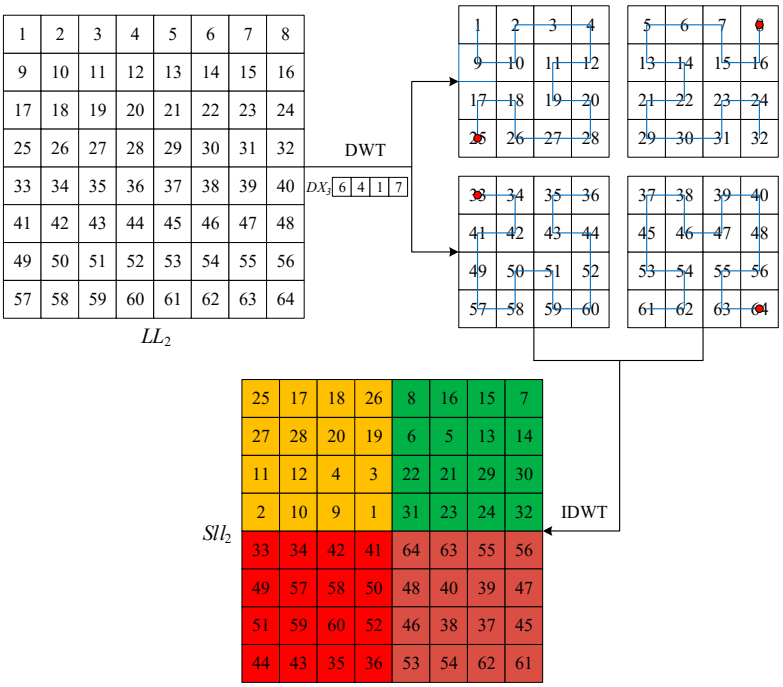


Figure 6. Schematic diagram of Hilbert curve scrambling process.

3.2.2. Bit-Level Scrambling

Although random pixel positions are scrambled, the pixel values are not changed, and the statistical attack can still obtain valid information in plaintext to resist statistical attacks. To resist statistical attacks and prevent an encrypted image from being cracked,

each pixel value needs to be changed. To solve this problem, this paper uses the main-diagonal extraction model to scramble the position of the pixel's bits and thus change each pixel value.

The Sarrus rule is a standard method to expand second-order and third-order determinants in linear algebra theory. In the determinant, the elements on the main-diagonal from the upper left corner to the lower right corner are called main-diagonal elements. The sub-diagonal is from the upper right corner to the lower left corner, and the elements on the sub-diagonal are called sub-diagonal elements. Since the calculation method is the product of the main-diagonal elements minus the development of the sub-diagonal elements, it is also called the diagonal rule. Taking the third-order determinant as an example, Figure 7 shows the expansion of the third-order determinant.

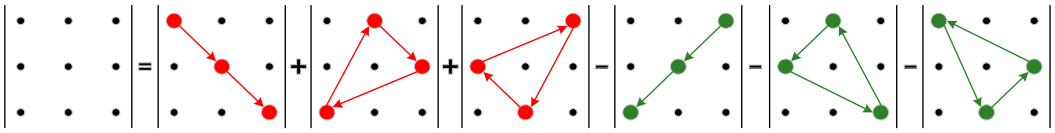


Figure 7. Third-order determinant expansion.

Based on the connection of the three favorable terms of the main-diagonal in the third-order determinant, this connection can be extended to the eighth-order to create a model of the eighth-order main-diagonal extraction, and the specific transformation process is shown in Figure 8. This matrix is partitioned into two upper and lower triangles according to the main-diagonal, and the upper triangle is extracted in the direction of the main-diagonal. In comparison, the lower triangle is removed in the opposite direction of the main-diagonal. By connecting the elements with the same color in the order of an upper triangle and then a lower triangle, eight one-dimensional sequences of length eight can be obtained, and these eight one-dimensional sequences are arranged in order in rows to form an 8×8 bit matrix. This model can scramble the bit matrix of image pixels.

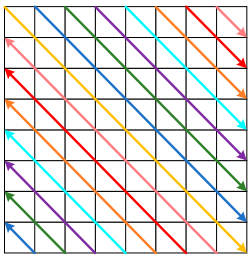


Figure 8. Main-diagonal extraction model.

The pixels of an image matrix P of size $M \times N$ are arranged in rows to form a one-dimensional sequence P', and the intercept of the last $M \times N$ elements of the sequence X is used to obtain the sequence LX. The sequence LX is then arranged in ascending order to obtain its index sequence ILX. This ILX index sequence is used to randomly selected eight pixels in the sequence P' and convert them into binary form, and each binary sequence is then arranged in order by rows to obtain the 8×8 bit matrix B_P. Then, the main-diagonal extraction model is used to scramble the bit matrix B_P to obtain the bit matrix B'_P. The scrambled bit matrix B'_P is converted into a decimal by rows, and this process is repeated $M \times N/8$ times until all the bits of pixels have been scrambled to obtain the scrambled matrix P''. Figure 9 shows the process of the bit-level scrambling for any eight pixels.

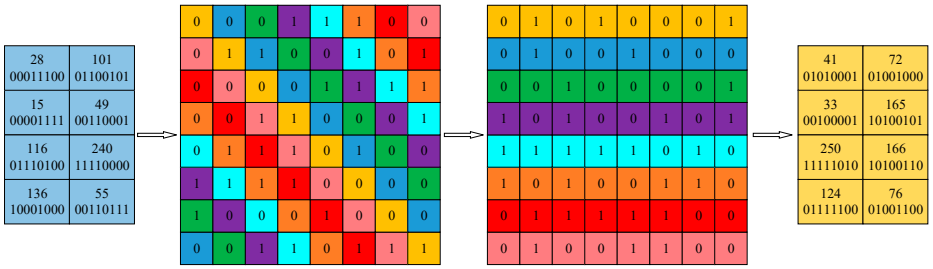


Figure 9. Example of the bit-level scrambling process.

3.3. Pixel Diffusion

3.3.1. Dynamic DNA Coding

In biology, DNA consists of four bases: A (adenine), C (cytosine), G (guanine), and T (thymine). A and T as well as C and G are complementary. In binary, 00 and 11 and 01 and 10 are also complementary. Therefore, the four bases A, G, C, and T can be used to represent 00, 01, 10, and 11. There are 24 coding rules, of which only 8 satisfy the Watson–Crick complementarity rule [43], as Table 1 shows.

Table 1. DNA coding rule.

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

Pixel diffusion is an algorithm that changes the pixel value of an image, and it plays a crucial role in combating differential attacks. Coding a binary sequence using any DNA coding rule and decoding it using a different DNA decoding rule can change the pixel value. However, such fixed coding and decoding rules are poorly randomized, vulnerable to attacks, and lack security. Meanwhile, if fixed DNA coding rules are used for binary sequences containing consecutive '0' or '1' values, consecutive identical bases will occur, posing a significant security risk. To overcome this drawback and to further improve the security of ciphertext images, the binary sequences with different pixel-value bit crossovers and various DNA coding and decoding rules are randomly selected in combination with chaotic sequences, thus changing each pixel value. Dynamic DNA coding for diffusion is more random and has a better diffusion effect.

Four pixels are selected in the image matrix using Equation (13) to form a 2×2 sub-block, and a total of $M \times N/4$ sub-blocks can be selected. The four pixels in a sub-block are converted into a binary sequence, and the binary sequence of pixels $P(i, j)$ and $P(i, N + 1 - j)$ are concatenated in the first row, followed by concatenation of the binary sequence of pixels $P(M + 1 - i, j)$ and $P(M + 1 - i, +1 - j)$ in the second row. The bits of these two rows of binary sequences crossover to perform an information fusion and hide the detailed information of the pixels. The sequences Y and Z are then processed with Equation (14) to obtain the sequences DY and DZ, which are coded by the sequence DY with randomly selected DNA coding rules for bit crossover pairs, and then decoded by the sequence DZ with randomly selected DNA decoding rules. Lastly, the decoded binary sequence is converted into decimal form. The specific process is shown in Figure 10.

$$\begin{bmatrix} P(i,j) & P(i,N+1-j) \\ P(M+1-i,j) & P(M+1-i,+1-j) \end{bmatrix} \quad \begin{matrix} 1 \le i \le M \\ 1 \le j \le N \end{matrix} \tag{13}$$

$$\begin{cases} DY = mod(Y, 8) + 1 \\ DZ = mod(Z, 8) + 1 \end{cases} \tag{14}$$

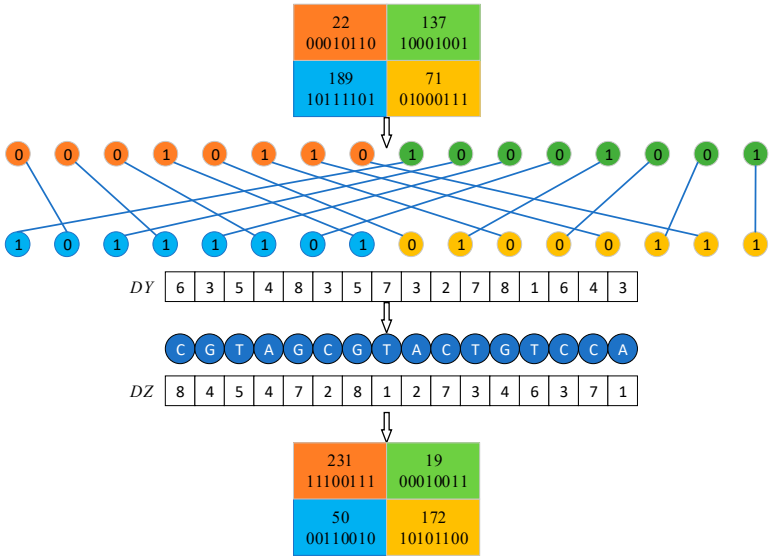


Figure 10. Dynamic DNA coding process.

3.3.2. Two-Way Ciphertext Diffusion

To further improve the diffusion effect, a slight change is diffused to each pixel value of the ciphertext. We use ciphertext diffusion to modify each pixel value. Sequences FW and LW are obtained by intercepting the first and last $M \times N$ values of the sequence W, respectively. The sequences FW and LW are converted into the two-dimensional matrices U and V with M rows and N columns, respectively.

Positive diffusion:

$$E_{i,j} = \begin{cases} mod((P_{i,j} + P_{M,N}), 256) \oplus U_{i,j} & i = 1,\ j = 1 \\ mod((P_{i,j} + E_{i,j-1}), 256) \oplus U_{i,j} & j \neq 1 \\ mod((P_{i,j} + E_{i-1,N}), 256) \oplus U_{i,j} & i \neq 1,\ j = 1 \end{cases} \tag{15}$$

Reverse diffusion:

$$C_{i,j} = \begin{cases} mod((E_{i,j} + E_{1,1}), 256) \oplus V_{i,j} & i = M,\ j = N \\ mod((E_{i,j} + C_{i,j+1}), 256) \oplus V_{i,j} & j \neq 1 \\ mod((E_{i,j} + C_{i+1,1}), 256) \oplus V_{i,j} & i \neq M,\ j = N \end{cases} \tag{16}$$

where i is the row, and j is the column, P is the plaintext, E represents the result of forward diffusion, and C is the ciphertext.

3.4. Encryption Algorithm

This encryption algorithm encrypts a square image; if the input image is non-square, it needs to be filled with "0" to make it a square with maximum side length.

The encryption algorithm proposed first obtains the key to the chaotic system using the plaintext image and the SHA-384 algorithm. A three-level DWT on the plaintext image and a Hilbert curve with different modes are randomly selected to scramble the pixels of each component locally, and then all components are reconstructed by IDWT. After that, the main-diagonal extraction model combined with the index sequence is used to scramble and diffuse each pixel simultaneously to achieve global scrambling. Finally, the scrambling

matrix is modified by dynamic DNA coding and two-way ciphertext diffusion for each pixel value to obtain the ciphertext image. The encryption steps are as follows:

Input: Plaintext image P.

Output: Ciphertext image C.

Step 1: Input a plaintext image P of size $M \times N$.

Step 2: The hash value H of the plaintext image P is calculated using the SHA-384 algorithm, and the parameters φ of the chaotic system are computed according to Equations (4) and (5). The initial values x_0, y_0, z_0, and w_0 of the chaotic system are computed according to Equations (6) and (7).

Step 3: Substitute the parameters and initial values into the hyperchaotic system for $4M \times N + 1000$ iterations and remove the first 1000 values. Obtain four chaotic sequences SX, SY, SZ, and SW, which are processed according to Equation (9) to obtain the sequences X, Y, Z, and W.

Step 4: The three-level DWT is performed on the plaintext image P to obtain the low-frequency component LL_3 and nine high-frequency components: HL_1, LH_1, HH_1, HL_2, LH_2, HH_2, HL_3, LH_3, and HH_3. The pixels of these ten components are scrambled according to the Hilbert curve scramble in Section 3.2.1 to obtain ten scrambled matrices: Shl_1, Slh_1, Shh_1, Shl_2, Slh_2, Shh_2, Shl_3, Slh_3, Shh_3, and Sll_3. IDWT reconstructs these to obtain the scrambled matrix P_1.

Step 5: Arrange the pixels in the matrix P_1 by rows into a one-dimensional sequence SP_1. According to the bit-level scramble method in Section 3.2.2, use the index sequence ILX in the sequence SP_1 to randomly select eight pixels at a time for the bit-level scramble, repeating this $M \times N/8$ times to obtain the scrambled matrix P_2.

Step 6: According to the dynamic DNA coding method in Section 3.3.1, four pixels in the matrix P_2 are first selected and converted into two binary sequences and allowed to undergo a bit crossover. Code the two bits of crossover with the sequence DY to obtain a DNA base sequence, which is then decoded with sequence DZ to obtain a binary sequence. This is then turned into a decimal form. Repeat $M \times N/4$ times to obtain the diffusion matrix P_3.

Step 7: Diffusion methods are as described in Section 3.3.2. The matrix P_3 and the matrices U and V are globally diffused using Equations (15) and (16) to obtain the ciphertext image C.

Figure 11 shows the complete steps of the encryption algorithm.

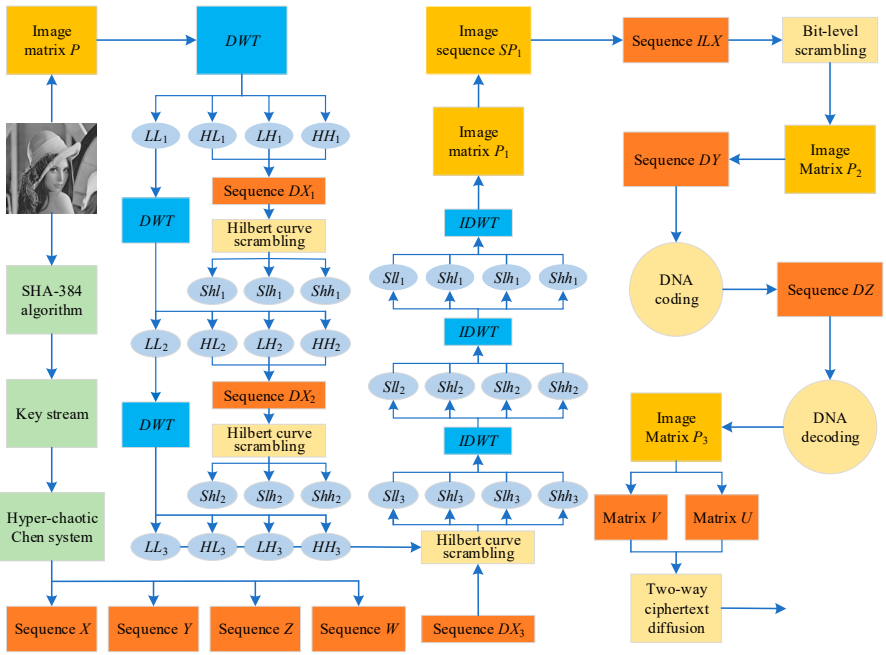


Figure 11. Encryption algorithm flow chart.

4. Experimental Simulation Results

Simulation of plaintext images of Lena, Jokul, Bridge, Bank, and Peppers was performed using the proposed encryption algorithm in the MATLAB experimental platform, and the simulation results are shown in Figure 12. It is obvious that the ciphertext image loses the feature information, and the image cannot be recognized. The decrypted image can be fully recovered without distortion, which proves that the algorithm encrypts very well and has high security.

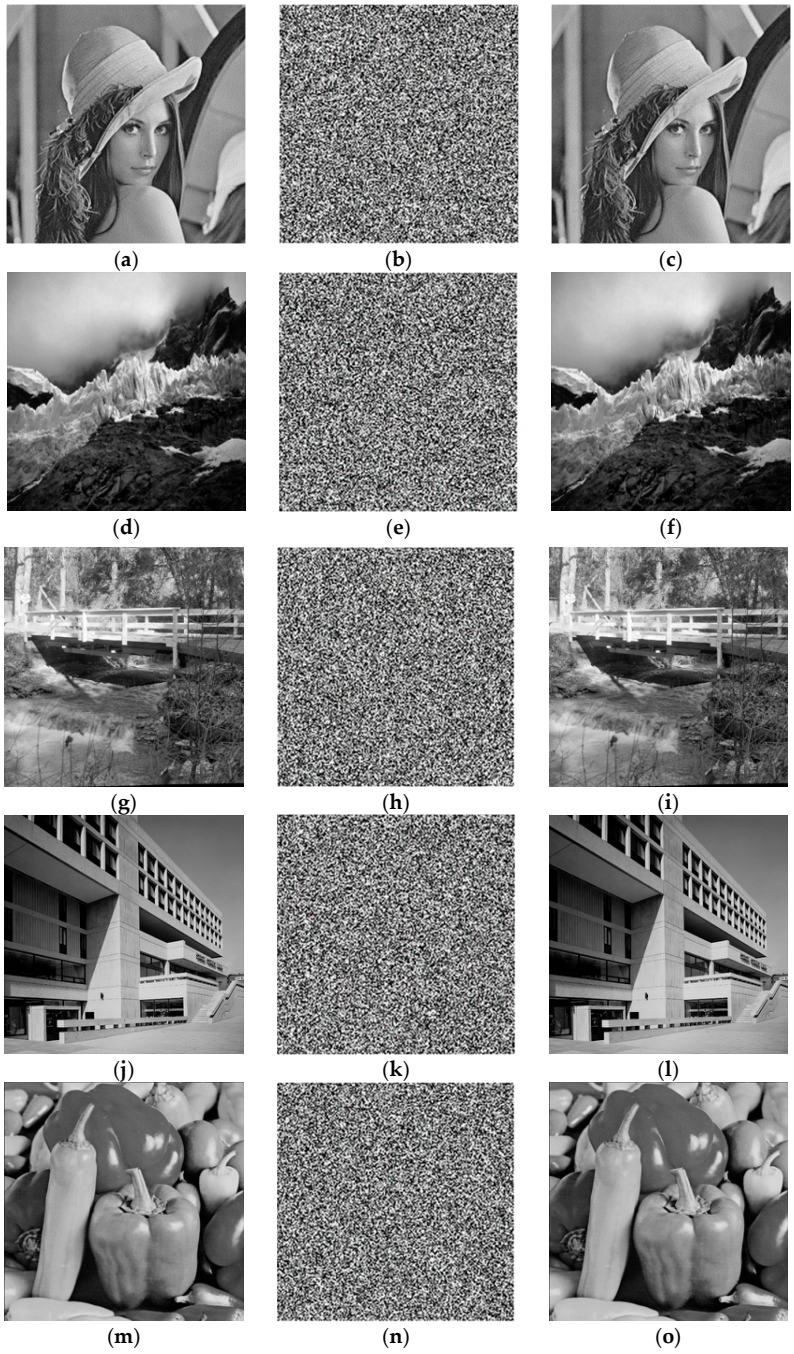


Figure 12. Simulation results: (**a**) Lena plaintext image; (**b**) Lena ciphertext image; (**c**) Lena decrypted image; (**d**) Jokul plaintext image; (**e**) Jokul ciphertext image; (**f**) Jokul decrypted image; (**g**) Bridge plaintext image; (**h**) Bridge ciphertext image; (**i**) Bridge decrypted image; (**j**) Bank plaintext image; (**k**) Bank ciphertext image; (**l**) Bank decrypted image; (**m**) Peppers plaintext image; (**n**) Peppers ciphertext image; (**o**) Peppers decrypted image.

5. Security Analysis

The feasibility of an encryption algorithm cannot be judged by the degree of blurring of the ciphertext image alone but also by more refined experimental analyses. To test whether the proposed encryption algorithm can withstand malicious attacks by unscrupulous elements, this section compares tests in terms of key, histogram, Chi-square, correlation, information entropy, local information entropy, homogeneity, contrast, energy, PSNR, MES, and robustness.

5.1. Key Space Analysis

To illegally obtain some valuable information, unscrupulous people often use brute force attacks to decrypt the images being transmitted. The key space is the set of keys for the image, which is a direct criterion to judge whether an encryption algorithm has the ability to resist malicious attacks and ensure that data information is not compromised. In general, an encryption algorithm has a key space of not less than 2^{100}. The encryption algorithm's security becomes stronger as the key space increases. In the encryption algorithm proposed, SHA-384 is used to generate a key with a length of 384 bits, and its key space $KS_1 = 2^{192}$. The initial values x_0, y_0, z_0, and w_0 are calculated with a precision of 10^{15}, and its keyspace $KS_2 = 10^{60}$. The total keyspace $KS = KS_1 \times KS_2 \approx 6.28 \times 10^{117}$, i.e., $KS \gg 2^{100}$, so the encryption algorithm proposed is sufficient to resist various violent attacks.

5.2. Key Sensitivity Analysis

A secure image encryption algorithm with ample space for the key should also have strong sensitivity. During sensitivity testing of the key, only minor changes are made to the original key, which is then used to decrypt the encrypted image. The greater the difference between the plaintext image and the ciphertext image, the stronger the sensitivity of the key.

Key sensitivity test with the Lena image: The initial parameters x_0, y_0, z_0, and w_0 are slightly changed and decrypted with the changed initial parameters. Figure 13 shows the test results, and it is apparent from Figure 13c–f that the decrypted images are entirely different, even though only minor changes were made to the key. Also, Figure 13g–l show no difference between any two decrypted images in Figure 13c–f, thus indicating that the algorithm is susceptible to the key.

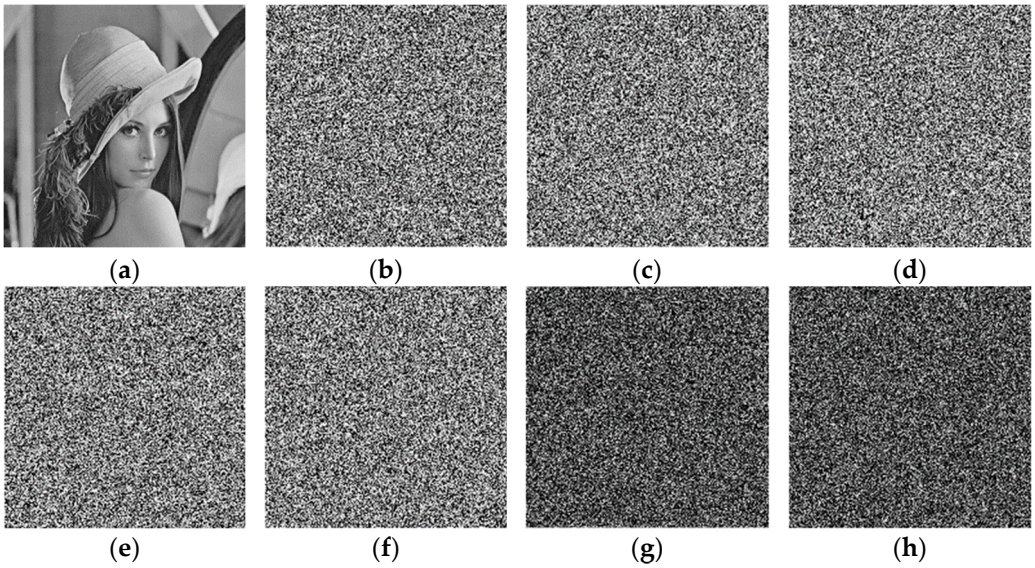
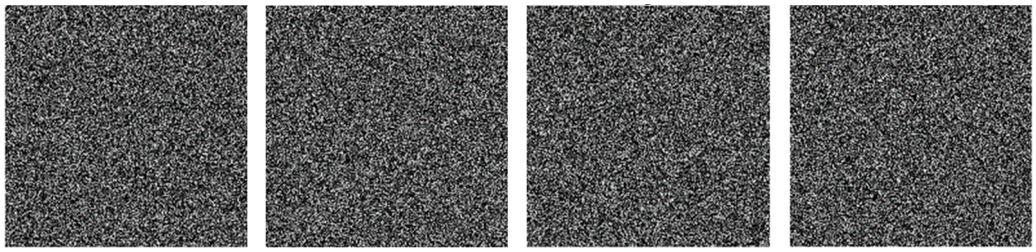


Figure 13. *Cont.*



(**i**) (**j**) (**k**) (**l**)

Figure 13. Key sensitivity test: (**a**) Lena; (**b**) Lena ciphertext; (**c**) $x_0 + 10^{-12}$; (**d**) $y_0 + 10^{-12}$; (**e**) $z_0 + 10^{-12}$; (**f**) $w_0 + 10^{-12}$; (**g**) Difference between (**c**,**d**); (**h**) Difference between (**c**,**e**); (**i**) Difference between (**c**,**f**); (**j**) Difference between (**d**,**e**); (**k**) Difference between (**d**,**f**); (**l**) Difference between (**e**,**f**).

5.3. Histogram Analysis

The distribution of the image pixels can be reflected visually from the histogram. All characteristic statistical attacks break the image by analyzing a histogram with a comparatively uneven distribution. The highly secure encryption algorithm resists statistical attacks and completely breaks up the image pixels, making the histogram more uniform. The histograms of different images are shown in Figure 14. It is not difficult to see that the distribution of ciphertext image pixels is more uniform than that of plaintext image pixels, which better hides the image information and proves that the encryption algorithm proposed can resist statistical attacks well.

5.4. Chi-Square Test

The intuitive histogram is only a rough assessment of the homogeneity of the image; to accurately quantify the uniformity of the histogram, a numerical operation using the difference square formula is required, which is defined by Equation (17):

$$s^2 = \frac{1}{256}\sum_{0}^{255}\frac{(f_i - \mathrm{g})^2}{\mathrm{g}} \tag{17}$$

where s^2 represents the Chi-square, f_i is the appearance rate of this gray-level pixel value in the histogram of the ciphertext image, and g is the theoretical rate of appearances of the pixel value at that gray level in the histogram, denoted as $\mathrm{g} = (M \times N)/256$. The significance level α is chosen to be 0.05, $s^2_{0.05} = 293.24783$. When the Chi-square of the test ciphertext image is less than this, i.e., $s^2 < s^2_{0.05}$, the histogram is approximately uniformly distributed. Table 2 shows the Chi-square test results. By comparison, the ciphertext image has a much lower Chi-square value than the plaintext image, suggesting that ciphertext images have uniform pixel values.

Table 2. Chi-square test results.

Images	Lena	Jokul	Bridge	Bank	Peppers
plaintext	39,387	18,649	27,574	18,569	31,602
ciphertext	226.4821	234.8416	252.1406	267.2613	255.4587

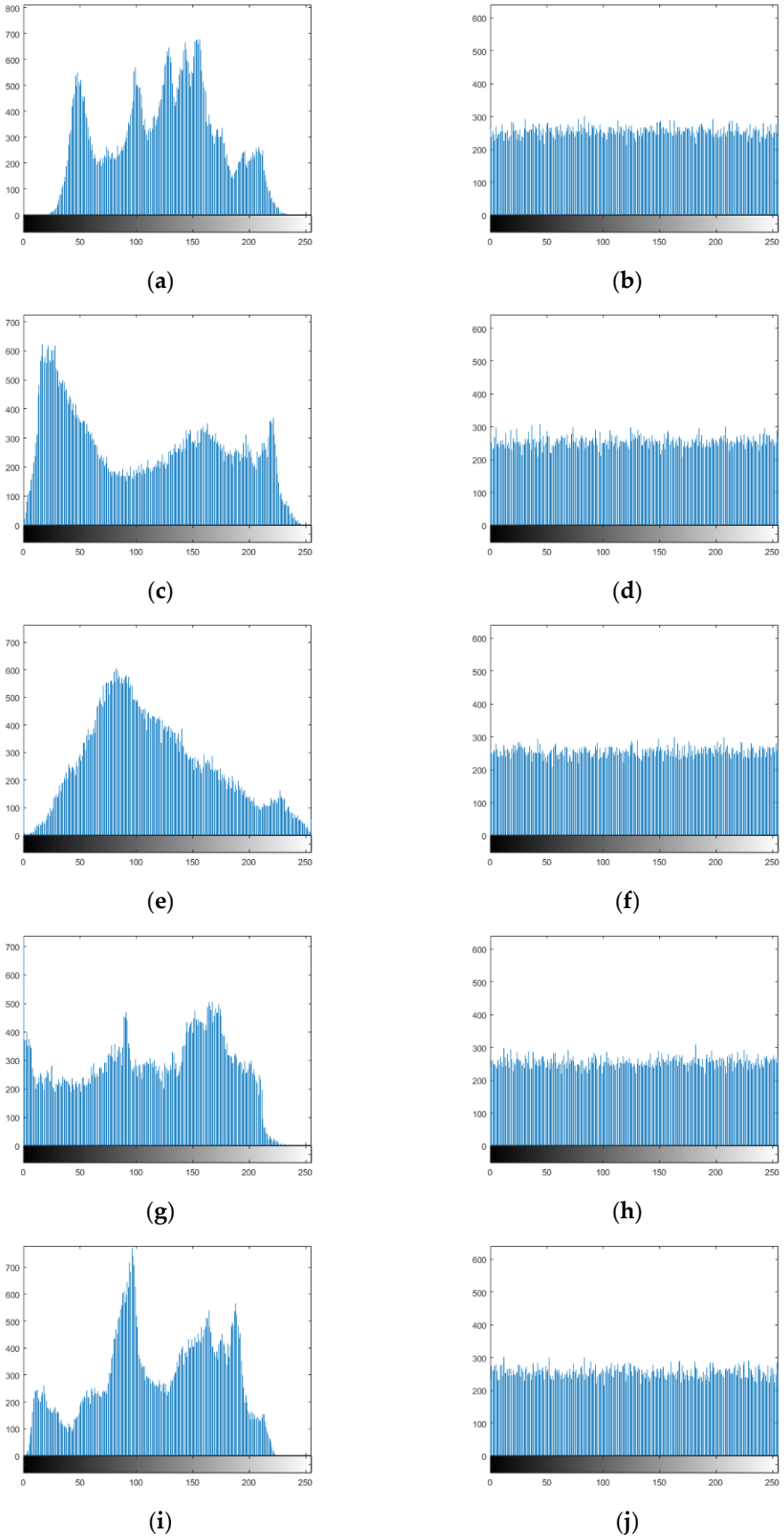


Figure 14. Histogram results: (**a**) Lena histogram; (**b**) Lena ciphertext histogram; (**c**) Jokul histogram; (**d**) Jokul ciphertext histogram; (**e**) Bridge histogram; (**f**) Bridge ciphertext histogram; (**g**) Bank histogram; (**h**) Bank ciphertext histogram; (**i**) Peppers histogram, (**j**) Peppers ciphertext histogram.

5.5. Correlation Analysis

Adjacent pixels in all directions are highly correlated, and if a piece of information is compromised, statistical attacks can decipher other information based on this information, causing a chain reaction. To ensure that the image is not cracked, successfully reducing the correlation between adjacent pixels becomes the key to the encryption algorithm. The correlation coefficient is calculated using Equation (18).

$$r_{x,y} = \frac{\operatorname{cov}(x,y)}{\sqrt{D(x)D(y)}} \tag{18}$$

The formula for each parameter in Equation (18) is as follows in Equation (19).

$$\begin{cases} E(x) = \frac{1}{N}\sum_{i=1}^{N} x_i \\ D(x) = \frac{1}{N}\sum_{i=1}^{N} [x_i - E(x)]^2 \\ \operatorname{cov}(x,y) = E([x - E(x)][y - E(y)]) \end{cases} \tag{19}$$

where $r_{x,r}$ is the correlation coefficient, x and y are a pair of pixel values, $E(x)$ is the expectation of x, $D(x)$ is the variance of x, $cov(x,y)$ is the covariance, and N is the total number of pixels in the image. From plaintext and ciphertext images, 5000 pairs of pixels are randomly selected, and their correlation coefficients in each direction are calculated separately. The results are shown in Table 3, which shows that the correlation coefficient of the ciphertext images tends to be 0. This shows that the proposed encryption algorithm can effectively weaken the pixel correlation. Figure 15 also shows that the correlation between adjacent pixels of the ciphertext image is broken. Meanwhile, the comparison of the correlation under different encryption algorithms is shown in Table 4. The results show that the proposed encryption algorithm in this paper is more destructive to the correlation of the plaintext image compared with other encryption algorithms.

Table 3. Correlation coefficients.

Images		Horizontal Direction	Vertical Direction	Diagonal Direction
Lena	plaintext	0.9663	0.9249	0.9194
	ciphertext	0.0013	0.0021	−0.0028
Jokul	plaintext	0.9777	0.9795	0.9635
	ciphertext	0.0024	−0.0035	−0.0029
Bridge	plaintext	0.9205	0.9402	0.8903
	ciphertext	−0.0016	0.0043	−0.0027
Bank	plaintext	0.9385	0.9235	0.8955
	ciphertext	0.0102	−0.0061	0.0054
Peppers	plaintext	0.9686	0.9651	0.9357
	ciphertext	0.0048	−0.0036	−0.0014

Table 4. Correlation coefficient comparison.

Images	Algorithms	Horizontal Direction	Vertical Direction	Diagonal Direction
Lena	Proposed	0.0013	0.0021	−0.0028
	[44]	−0.0059	−0.0046	−0.0003
	[14]	0.0060	0.0021	−0.005
	[45]	0.0015	−0.0090	−0.0120
	[46]	0.0058	−0.0051	−0.0030

Table 4. *Cont.*

Images	Algorithms	Horizontal Direction	Vertical Direction	Diagonal Direction
Peppers	Proposed	0.0048	−0.0036	−0.0014
	[44]	−0.0026	−0.0012	−0.0050
	[14]	−0.0025	−0.0040	−0.0015
	[45]	−0.0083	0.0081	−0.0142
	[46]	−0.0011	−0.0073	−0.0019

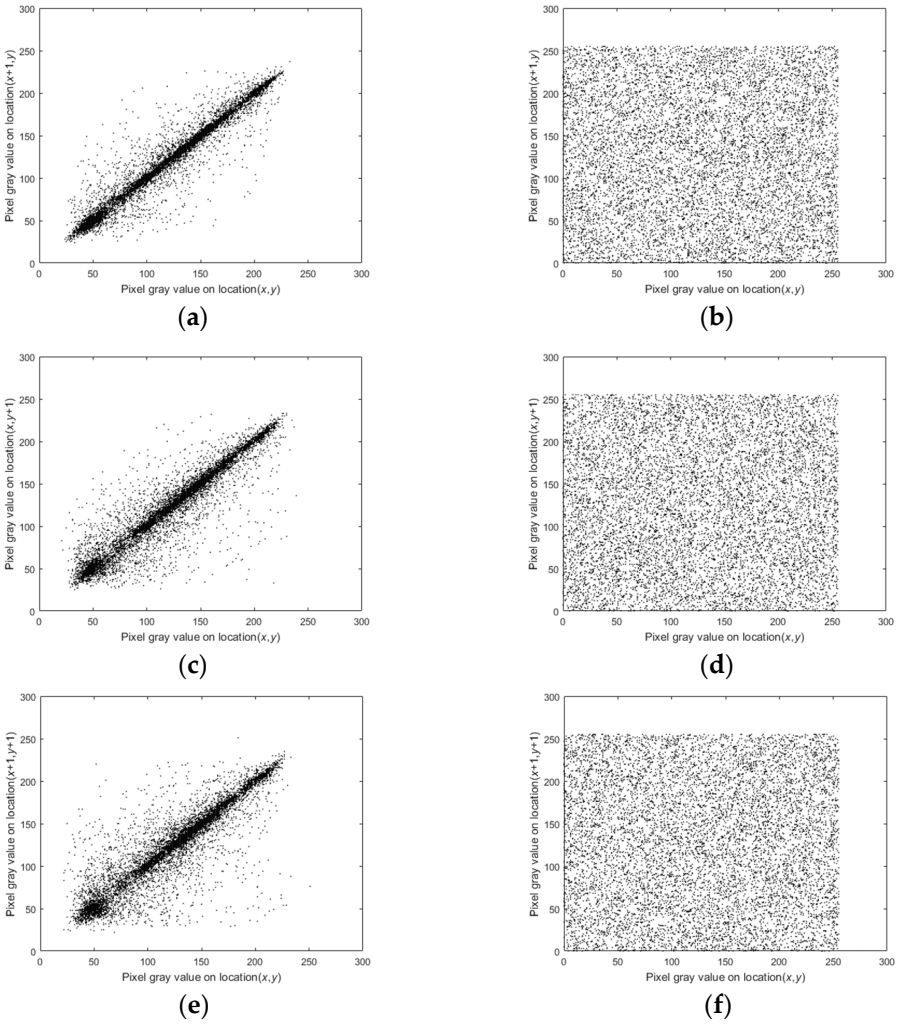


Figure 15. Correlation coefficients of Lena: (**a**) Horizontal correlation of the Lena plaintext image; (**b**) Horizontal correlation of the Lena ciphertext image; (**c**) Vertical correlation of the Lena plaintext image; (**d**) Vertical correlation of the Lena ciphertext image; (**e**) Diagonal correlation of the Lena plaintext image; (**f**) Diagonal correlation of the Lena ciphertext image.

5.6. Information Entropy and Local Information Entropy

Information entropy is a key metric for detecting the randomness of image pixels and for quantifying the average amount of information in the image. In images with high information entropy, pixels are distributed more uniformly; the stronger the randomness, the less it is likely to be cracked. Information entropy is calculated by Equation (20):

$$H(m) = -\sum_{0}^{L-1} P(m_i)\log_2 P(m_i) \tag{20}$$

where L is the image's gray level. When $L = 256$, its information entropy in the ideal state is 8. m_i is the ith pixel value of the image, and $P(m_i)$ is the chance of occurrence of the corresponding pixel value. In the information entropy test, five different images are converted into ciphertext images separately using the proposed encryption algorithm. Then, their information entropy is obtained, and the test results are displayed in Table 5. All ciphertext images have an information entropy value close to 8. Meanwhile, the comparison with other algorithms in Table 6 also visually proves that the proposed encryption algorithm has high security.

Table 5. Information entropy.

Information Entropy	Lena	Jokul	Bridge	Bank	Peppers
Plaintext	7.4539	7.5209	7.4236	7.3841	7.5794
Ciphertext	7.9987	7.9985	7.9982	7.9977	7.9981

Table 6. Comparison of information entropy.

Images	Algorithms	Plaintext	Ciphertext
Lena	Proposed	7.4539	7.9987
	[44]	7.4492	7.9971
	[14]	7.4446	7.9974
	[45]	7.3875	7.9939
	[46]	7.4446	7.9974
Peppers	Proposed	7.5794	7.9981
	[44]	7.3576	7.9967
	[14]	7.3800	7.9972
	[45]	7.5697	7.9973
	[46]	7.5327	7.9967

For ciphertext images with uniform pixel distribution, the calculated results are accurate, but some ciphertext images may have an uneven local pixel distribution. To overcome this drawback of information entropy and to further improve the accuracy of this evaluation criterion, Wu proposed the calculation method of local information entropy [47], as shown in Equation (21).

$$H_{(k,T_B)(S)} = \sum_{i=1}^{k} \frac{H(S_i)}{k} \tag{21}$$

where S_i is the ciphertext information entropy, k is the number of selected groups, and T_B is the number of pixels in each group. The ideal range of the local information entropy is obtained when k is chosen to be 3, T_B is 1936, and the significance level α is 0.05 in the field [7.901515698, 7.903422936]. If the test result is within this range, it means that the ciphertext image passes the test. In this test section, five different images are converted into ciphertext images using the proposed encryption algorithm; then, their local information entropy is obtained, and the test results are in Table 7. All the ciphertext images pass the test.

Table 7. Local information entropy.

Images	Lena	Jokul	Bridge	Bank	Peppers
Local information entropy	7.9021	7.9023	7.9019	7.9031	7.9026
Pass or fail	Pass	Pass	Pass	Pass	Pass

5.7. Homogeneity Analysis

The gray level co-occurrence matrix (GLCM) represents the various combinations of pixel luminance. Homogeneity analysis quantifies the distribution of elements in the

GLCM and further determines how similar they are to the diagonal. The homogeneity value decreases as the distance of the elements from the diagonal becomes more prominent in the range of [0, 1], and the smaller the homogeneity value is, the more efficient the encryption algorithm is. It is calculated as shown in Equation (22) [48].

$$Homogeneity = \sum_i \sum_j \frac{P(i,j)}{1+|i-j|} \tag{22}$$

where i and j are two horizontally adjacent gray values, and $P(i,j)$ is the element's value in the normalized GLCM. The homogeneity values of different plaintext images and ciphertext images are calculated, and the results are shown in Table 8. From Table 8, it can be seen that the homogeneity value of the ciphertext image is at a shallow level. Also, in Table 9, by comparing with other algorithms, the ciphertext image in this paper has the lowest homogeneity value, which shows that the encryption algorithm strongly resists statistical attacks.

Table 8. Homogeneity.

Images	Lena	Jokul	Bridge	Bank	Peppers
Plaintext	0.8456	0.8857	0.8237	0.8810	0.9002
Ciphertext	0.3895	0.3892	0.3877	0.3893	0.3907

Table 9. Comparison of homogeneity, contrast, and energy.

Algorithm	Homogeneity	Contrast	Energy
Proposed	0.3895	10.5331	0.0156
[49]	0.3896	10.4968	0.0156
[50]	0.3901	10.5324	0.0156
[51]	0.3887	10.5325	0.0156
[52]	0.4633	10.3011	0.0152

5.8. Contrast Analysis

The contrast is usually measured for the intensity between an image pixel and its neighboring pixels. Generally, the contrast value of plaintext images is shallow, while the contrast value of ciphertext images is high. The higher the contrast value, the higher the randomness of the ciphertext image and the more resistant the encryption algorithm is to statistical attacks. The contrast value is calculated as shown in Equation (23) [48].

$$Contrast = \sum_{i,j=0} P_{i,j} \times (i-j)^2 \tag{23}$$

where i and j are two horizontally adjacent gray values, and $P(i,j)$ is the element's value in the normalized GLCM. The contrast values of plaintext and ciphertext images are shown in Table 10, from which it can be seen that the ciphertext image has a higher contrast value than the plaintext image. It can also be seen in the comparison in Table 9 that the ciphertext image of this paper has the highest contrast value, which shows that the encryption algorithm is effective against statistical attacks.

Table 10. Contrast.

Images	Lena	Jokul	Bridge	Bank	Peppers
Plaintext	0.5591	0.2997	0.4602	0.6230	0.2980
Ciphertext	10.5331	10.3813	10.3729	10.4099	10.4157

5.9. Energy Analysis

The energy is the cumulative sum of the squares of all elements in the GLCM and represents how much image information is contained. The larger the energy value of an image, the more information it contains, and the easier it is to be cracked by a statistical attack. An encryption algorithm with strong resistance to statistical attacks should have very low energy for the ciphertext image. The energy is calculated as shown in Equation (24) [48].

$$Energy = \sum_{i} \sum_{j} P(i,j)^2 \tag{24}$$

where i and j are two horizontally adjacent gray values, and $P(i,j)$ is the element's value in the normalized GLCM. The energy values of the plaintext image and the ciphertext image are shown in Table 11, from which it can be seen that the ciphertext image has a shallow energy value. The results of comparing the energy values of this encryption algorithm with other algorithms are shown in Table 9. It is easy to see that the energy value of the ciphertext image of this encryption algorithm is one of the lowest, which shows that this encryption algorithm has a certain degree of security in terms of resistance to statistical attacks.

Table 11. Energy.

Images	Lena	Jokul	Bridge	Bank	Peppers
Plaintext	0.0786	0.0946	0.0838	0.0971	0.1135
Ciphertext	0.0156	0.0156	0.0156	0.0156	0.0156

5.10. MES and PSNR Analyses

PSNR (Peak Signal Noise Ratio) and MSE (Mean Square Error) are two objective metrics used to evaluate image quality. The more significant the PSNR value, the smaller the image distortion; the more precise the image, the worse the encryption effect. As a result, higher MSE values indicate a better encryption performance when testing plaintext and ciphertext images. They are calculated according Equations (25) and (26).

$$\text{MSE} = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} (P(i,j) - C(i,j))^2 \tag{25}$$

$$\text{PSNP} = 10 \times \log_{10} \left(\frac{L^2}{\text{MSE}} \right) \tag{26}$$

where P is the plaintext, C is the ciphertext, $M \times N$ is the size of the images, and L is the pixel gray level. The MSE and PSNR values of different images after encryption and the comparison with other algorithms are shown in Table 12. From this, we can see that the PSNR of the ciphertext image is very small, and it also outperforms other algorithms when compared and analyzed against different algorithms, which indicates the high performance of this encryption algorithm.

Table 12. Comparison of MSE and PSNR values.

Algorithms	Images	MSE	PSNR
Proposed	Jokul	8395.9526	8.8901
	Bridge	7730.2871	9.2488
	Bank	9505.5392	8.3510
	Peppers	8312.6971	8.9334
	Lena	10,659.0009	7.8536

Table 12. *Cont.*

Algorithms	Images	MSE	PSNR
[45]	Lena	7752.6	9.2363
[53]	Lena	9056.1634	8.5613
[54]	Lena	7802.1	9.2087
[55]	Lena	7797.7	9.2111

5.11. Differential Attack Analysis

The differential attack mainly involves making a small change to a pixel value of the plaintext image, then encrypting the two plaintext images using an encryption algorithm, and finally comparing and analyzing the two ciphertext images to discover their connection, from which the images can be cracked.

There are two extremely critical factors in evaluating differential attacks. One is the rate of change of pixel values, NPCR, and the other is the uniform average change intensity, UACI, and these are calculated as shown in Equation (27):

$$\begin{cases} D(i,j) = \begin{cases} 0 & C_1(i,j) = C_2(i,j) \\ 1 & C_1(i,j) \neq C_2(i,j) \end{cases} \\ NPCR = \frac{\sum_{i,j} D(i,j)}{M \times N} \times 100\% \\ UACI = \frac{1}{255 \times M \times N} \sum_{i,j} |C_1(i,j) - C_2(i,j)| \times 100\% \end{cases} \tag{27}$$

where $M \times N$ is the scale size of the ciphertext, C_1 and C_2 are the two ciphertexts to be compared, and $D(i,j)$ is used to discriminate C_1 and C_2.Under ideal conditions, the values of NPCR and UACI were 99.6049% and 33.4635%, respectively. With the key unchanged, the two plaintext images are converted into ciphertext images with the encryption algorithm. Tables 13 and 14 show the calculated NPCR and UACI values and the comparison results with different algorithms. Compared with other algorithms, this algorithm's NPCR and UACI values are closer to the theoretical values than most of them.

5.12. Noise Attack Analysis

Ciphertext is vulnerable to noise attacks during data transmission. The typical noise attacks are gaussian noise and pepper noise, and the noise attacks can damage the ciphertext image and reduce the clarity. Since pepper noise has more impact on ciphertext images than other noise, in this study, different strengths of pepper noise were included for testing. While keeping the key unchanged, we added pepper noise with intensities of 0.01, 0.05, 0.1, and 0.15 to interfere with the Lena image and encrypted and decrypted it using the encryption algorithm proposed. Figure 16 shows the ciphertext and decrypted images. It is evident that even when the noise intensity reached 0.15, the decrypted image could still be recognized, indicating that the encryption algorithm resists noise attacks.

Table 13. NPCR and UACI values.

Size	Images	NPCR (%)	UACI (%)
256 × 256	Lena	99.6073	33.4682
	Jokul	99.6154	33.4557
	Bridge	99.5886	33.4564
	Bank	99.6012	33.4623
	Peppers	99.5979	33.4708
512 × 512	Lena	99.6023	33.4658
	Jokul	99.6135	33.4717
	Bridge	99.5963	33.4576
	Bank	99.6155	33.4698
	Peppers	99.6004	33.4618

Table 14. Comparison of NPCR and UACI values.

Size	Algorithm	NPCR (%)	UACI (%)
Lena (256 × 256)	Proposed	99.6073	33.4682
	[44]	99.6197	33.0443
	[14]	90.6047	33.4719
	[45]	99.6185	33.4561
	[46]	99.6085	33.4633
Lena (512 × 512)	Proposed	99.6023	33.4618
	[45]	99.6044	33.4117
	[56]	99.6101	33.4945
	[57]	99.6002	33.5079
	[58]	99.6140	31.4646

5.13. Cropping Attack Analysis

If the decryption algorithm is not robust against cropping attacks, the decryption will fail due to the missing information in the decryption process. If the decryption algorithm can restore the plaintext image to a large extent, the encryption algorithm is highly resistant to cropping attacks. In the test, the Lena ciphertext image was decrypted after cropping attacks of 1/64, 1/16, 1/4, and 1/2, respectively, and the results are shown in Figure 17. The Lena ciphertext image can still be seen as the basic outline of the original image after decryption with the addition of different degrees of cropping attacks, which indicates that the encryption algorithm strongly resists a cropping attack.

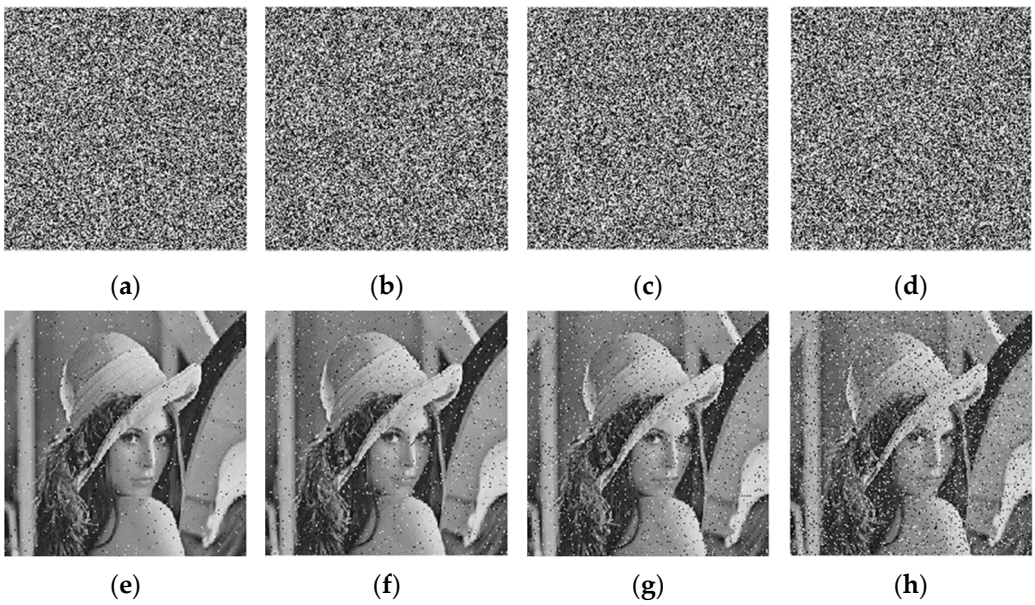


Figure 16. Noise attack experiment results: (**a**) 0.01 intensity ciphertext image; (**b**) 0.05 intensity ciphertext image; (**c**) 0.1 intensity ciphertext image; (**d**) 0.15 intensity ciphertext image; (**e**) 0.01 intensity decrypted image (**f**) 0.05 intensity decrypted image; (**g**) 0.1 intensity decrypted image; (**h**) 0.15 intensity decrypted image.

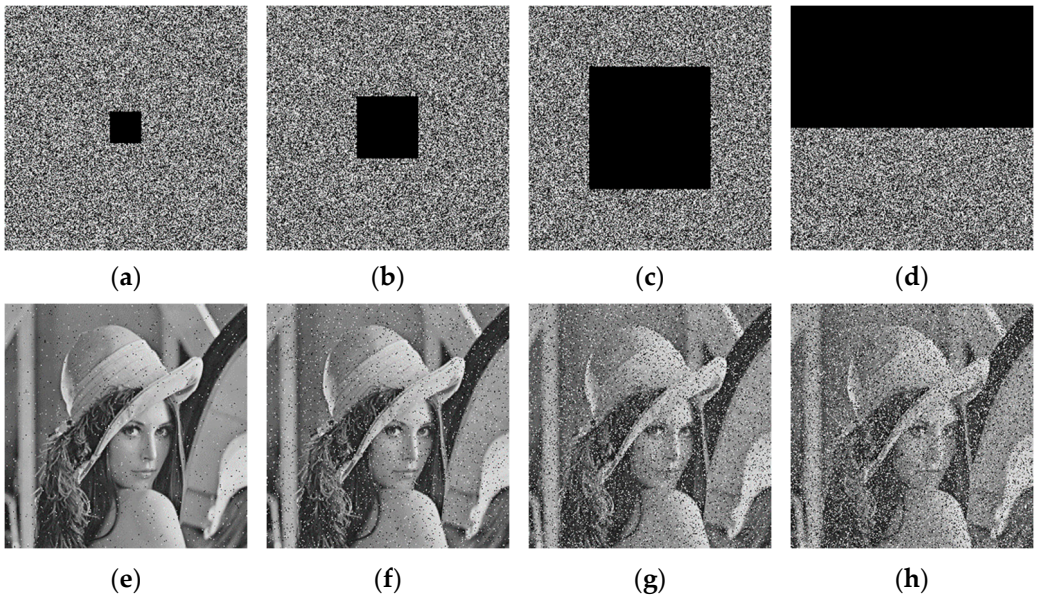


Figure 17. Cropping attacks experiment results: (**a**) 1/64 cropping; (**b**) 1/16 cropping; (**c**) 1/4 cropping; (**d**) 1/2 cropping; (**e**) Decrypted image of (**a**); (**f**) Decrypted image of (**b**); (**g**) Decrypted image of (**c**); (**h**) Decrypted image of (**d**).

6. Conclusions

In this paper, an image encryption algorithm based on improved Hilbert curve scrambling and dynamic DNA coding is proposed. First, the image matrix is divided into ten component matrices by using three-level DWT of the plaintext image, and the chaotic sequence generated by iterating with the hyperchaotic system randomly selects one of the eight Hilbert curve modes and scrambles each of the ten component matrices. The scrambled component matrices are then reconstructed with IDWT to obtain the scrambled image matrix. Next, eight pixels at a time are randomly selected with a chaotic sequence and bit-level scrambled using the main-diagonal extraction model until all pixels are scrambled. Then, bit crossover is performed between different pixels, and the pixel values are modified using dynamic DNA coding. Finally, the pixels are globally diffused using two-way ciphertext feedback to obtain the ciphertext image.

Simulation experiments and theoretical analysis verify the effectiveness of this encryption algorithm against chosen plaintext attacks, violent attacks, statistical attacks, differential attacks, noise attacks, and cropping attacks. Therefore, the encryption algorithm proposed in this paper has good security performance.

Author Contributions: Conceptualization, S.G.; Methodology, S.G.; Software, J.L.; Validation, J.L.; Writing—original draft, J.L.; Writing—review & editing, X.Z.; Supervision, X.Z.; Project administration, Y.W.; Funding acquisition, Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grants 62072417 and 62102374 and in part by the Henan provincial Science and Technology research project under Grants 202102210177 and 212102210028, with Xuncai Zhang as the corresponding author.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Wang, Y.; Quan, C.; Tay, C.J. Nonlinear multiple-image encryption based on mixture retrieval algorithm in Fresnel domain. *Opt. Commun.* **2014**, *330*, 91–98. [CrossRef]
2. Zahmoul, R.; Ejbali, R.; Zaied, M. Image encryption based on new Beta chaotic maps. *Opt. Lasers Eng.* **2017**, *96*, 39–49. [CrossRef]
3. Lorenz, E.N.; Hilborn, R.C. The essence of chaos. *Am. J. Phys.* **1995**, *63*, 862. [CrossRef]
4. Zhang, Y. The unified image encryption algorithm based on chaos and cubic S-Box. *Inf. Sci.* **2018**, *450*, 361–377. [CrossRef]
5. Kumar, C.M.; Vidhya, R.; Brindha, M. An efficient chaos based image encryption algorithm using enhanced thorp shuffle and chaotic convolution function. *Appl. Intell.* **2022**, *52*, 2556–2585. [CrossRef]
6. Zhu, H.; Zhao, Y.; Song, Y. 2D Logistic-Modulated-Sine-Coupling-Logistic Chaotic Map for Image Encryption. *IEEE Access* **2019**, *7*, 14081–14098. [CrossRef]
7. Zhang, Z.; Tang, J.; Zhang, F.; Ni, H.; Chen, J.; Huang, Z. Color Image Encryption Using 2D Sine-Cosine Coupling Map. *IEEE Access* **2022**, *10*, 67669–67685. [CrossRef]
8. Xu, M.; Tian, Z.H. A novel image encryption algorithm based on self-orthogonal Latin squares. *Optik* **2018**, *171*, 891–903. [CrossRef]
9. Li, C.; Arroyo, D.; Lo, K.T. Breaking a chaotic cryptographic algorithm based on composition maps. *Int. J. Bifurc. Chaos Appl. Sci. Eng.* **2010**, *20*, 2561–2568. [CrossRef]
10. Gao, X.H. Image encryption algorithm based on 2D hyperchaotic map. *Opt. Laser Technol.* **2021**, *142*, 107252. [CrossRef]
11. Arthi, G.; Thanikaiselvan, V.; Amirtharajan, R. 4D Hyperchaotic map and DNA encoding combined image encryption for secure communication. *Multimed. Tools Appl.* **2022**, *81*, 15859–15878. [CrossRef]
12. Fridrich, J. Symmetric ciphers based on two-dimensional chaotic maps. *Int. J. Bifurc. Chaos* **1998**, *8*, 1259–1284. [CrossRef]
13. Hua, Z.Y.; Zhu, Z.H.; Chen, Y.Y.; Li, Y.M. Color image encryption using orthogonal Latin squares and a new 2D chaotic system. *Nonlinear Dyn.* **2021**, *104*, 4505–4522. [CrossRef]
14. Wang, X.; Su, Y.; Xu, M.; Zhang, H.; Zhang, Y. A new image encryption algorithm based on Latin square matrix. *Nonlinear Dyn.* **2022**, *107*, 1277–1293. [CrossRef]
15. Norouzi, B.; Mirzakuchaki, S.; Seyedzadeh, S.M.; Mosavi, M.R. A simple, sensitive and secure image encryption algorithm based on hyper-chaotic system with only one round diffusion process. *Multimed. Tools Appl.* **2014**, *71*, 1469–1497. [CrossRef]
16. Sivakumar, T.; Venkatesan, R. Image encryption based on pixel shuffling and random key stream. *Int. J. Comput. Inf. Technol.* **2014**, *3*, 6.
17. Li, C.Q. Cracking a hierarchical chaotic image encryption algorithm based on permutation. *Signal Process.* **2016**, *118*, 203–210. [CrossRef]
18. Zhang, Y.; Tang, Y. A plaintext-related image encryption algorithm based on chaos. *Multimed. Tools Appl.* **2018**, *77*, 6647–6669. [CrossRef]
19. Zou, C.Y.; Wang, X.Y.; Zhou, C.J.; Xu, S.J.; Huang, C. A novel image encryption algorithm based on DNA strand exchange and diffusion. *Appl. Math. Comput.* **2022**, *430*, 127291. [CrossRef]
20. Hua, Z.Y.; Li, J.X.; Li, Y.M.; Chen, Y.Y. Image Encryption Using Value-Differencing Transformation and Modified ZigZag Transformation. *Nonlinear Dyn.* **2021**, *106*, 3583–3599. [CrossRef]
21. Sha, Y.; Cao, Y.; Yan, H.; Gao, X.; Mou, J. An Image Encryption Algorithm Based on IAVL Permutation Algorithm and DNA Operations. *IEEE Access* **2021**, *9*, 96321–96336. [CrossRef]
22. Chen, J.X.; Zhu, Z.L.; Fu, C.; Zhang, L.B.; Zhang, Y.S. An efficient image encryption algorithm using lookup table-based confusion and diffusion. *Nonlinear Dyn.* **2015**, *81*, 1151–1166. [CrossRef]
23. Xiang, T.; Wong, K.W.; Liao, X.F. Selective image encryption using a spatiotemporal chaotic system. *Chaos* **2007**, *17*, 023115. [CrossRef]
24. Li, C.L.; Zhou, Y.; Li, H.M.; Feng, W.; Du, J.R. Image encryption algorithm with bit-level scrambling and multiplication diffusion. *Multimed. Tools Appl.* **2021**, *80*, 18479–18501. [CrossRef]
25. Wang, X.Y.; Du, X.H. Pixel-level and bit-level image encryption method based on Logistic-Chebyshev dynamic coupled map lattices. *Chaos Solitons Fractals* **2022**, *155*, 111629. [CrossRef]
26. Clelland, C.; Risca, V.; Bancroft, C. Hiding messages in DNA microdots. *Nature* **1999**, *399*, 533–534. [CrossRef] [PubMed]
27. Zhao, J.F.; Wang, S.Y.; Zhang, L.T. Block Image Encryption Algorithm Based on Novel Chaos and DNA Encoding. *Information* **2023**, *14*, 2078–2489. [CrossRef]
28. Zhang, X.Q.; Wang, X.S. Multiple-image encryption algorithm based on DNA encoding and chaotic system. *Multimed. Tools Appl.* **2019**, *78*, 7841–7869. [CrossRef]
29. Jithin, K.C.; Sankar, S. Colour image encryption algorithm combining arnold map, dna sequence operation, and a mandelbrot set. *J. Inf. Secur. Appl.* **2020**, *50*, 102428. [CrossRef]
30. Feng, W.; Zhao, X.Y.; Zhang, J.; Qin, Z.T.; Zhang, J.K.; He, Y.G. Image Encryption Algorithm Based on Plane-Level Image Filtering and Discrete Logarithmic Transform. *Mathematics* **2022**, *10*, 2227–7390. [CrossRef]
31. Zou, C.Y.X.; Wang, Y. Image encryption algorithm with matrix semi-tensor product. *Nonlinear Dyn.* **2021**, *105*, 859–876. [CrossRef]
32. Li, J.Q.; Wang, J.; Di, X.Q. Image encryption algorithm based on bit-level permutation and "Feistel-like network" diffusion. *Multimed. Tools Appl.* **2022**, *81*, 44335–44362. [CrossRef]

33. Wang, T.; Wang, M.H. Hyperchaotic image encryption algorithm based on bit-level permutation and dna encoding. *Opt. Laser Technol.* **2020**, *132*, 106355. [CrossRef]
34. Chen, J.X.; Chen, L.; Zhou, Y.C. Cryptanalysis of a DNA-based image encryption algorithm. *Inf. Sci.* **2020**, *520*, 130–141. [CrossRef]
35. Shafique, A.; Ahmed, F. Image Encryption Using Dynamic S-Box Substitution in the Wavelet Domain. *Wirel. Pers. Commun.* **2020**, *115*, 2243–2268. [CrossRef]
36. Yan, X.P.; Wang, X.Y.; Xian, Y.J. Chaotic Image Encryption Algorithm Based on Fractional Order Scrambling Wavelet Transform and 3D Cyclic Displacement Operation. *IEEE Access* **2020**, *8*, 208718–208736. [CrossRef]
37. Qin, Q.; Liang, Z.Y.; Liu, S.; Wang, X.; Zhou, C.J. A Dual-Domain Image Encryption Algorithm Based on Hyperchaos and Dynamic Wavelet Decomposition. *IEEE Access* **2022**, *10*, 122726–122744. [CrossRef]
38. Li, Y.X.; Tang, W.K.S.; Chen, G.R. Generating hyperchaos via state feedback control. *Int. J. Bifurc. Chaos* **2005**, *15*, 3367–3375. [CrossRef]
39. Shaheen, A.M.; Sheltami, T.R.; Al-Kharoubi, T.M. Digital image encryption techniques for wireless sensor networks using image transformation methods: DCT and DWT. *J. Ambient Intell. Humaniz. Comput.* **2019**, *10*, 4733–4750. [CrossRef]
40. Deepak, V.K.; Rajakumaran, C.; Kavitha, R. Chaos based encryption of quantum images. *Multimed. Tools Appl.* **2020**, *79*, 23849–23860. [CrossRef]
41. Zhang, X.; Gong, Z. Color image encryption algorithm based on 3D Zigzag transformation and view planes. *Multimed. Tools Appl.* **2022**, *81*, 31753–31785. [CrossRef]
42. Wang, X.; Lin, S.; Li, Y. Bit-level image encryption algorithm based on BP neural network and gray code. *Multimed. Tools Appl.* **2021**, *80*, 11655–11670. [CrossRef]
43. Wang, X.; Li, B.; Wang, Y. An efficient batch images encryption method based on DNA encoding and PWLCM. *Multimed. Tools Appl.* **2021**, *80*, 943–971. [CrossRef]
44. Jun, W.J.; Fun, T.S. A New Image Encryption Algorithm Based on Single S-Box and Dynamic Encryption Step. *IEEE Access* **2021**, *9*, 120596–120612. [CrossRef]
45. Wang, X.Y.; Zhao, M.C. An image encryption algorithm based on hyperchaotic system and DNA coding. *Opt. Laser Technol.* **2021**, *143*, 107316. [CrossRef]
46. Wang, X.; Du, X. Chaotic image encryption method based on improved zigzag permutation and DNA rules. *Multimed. Tools Appl.* **2022**, *81*, 43777–43803. [CrossRef]
47. Wu, Y.; Zhou, Y.C.; Saveriades, G.; Agaian, S.; Noonan, J.P.; Natarajan, P. Local Shannon entropy measure with statistical tests for image randomness. *Inf. Sci.* **2013**, *222*, 323–342. [CrossRef]
48. Alghamdi, Y.; Munir, A.; Ahmad, J. A Lightweight Image Encryption Algorithm Based on Chaotic Map and Random Substitution. *Entropy* **2022**, *24*, 1344. [CrossRef]
49. Ahmad, J.; Hwang, S.O. Chaos-based diffusion for highly autocorrelated data in encryption algorithms. *Nonlinear Dyn.* **2015**, *82*, 1839–1850. [CrossRef]
50. Ahmad, J.; Tahir, A.; Khan, J.S.; Khan, M.A.; Khan, F.A.; Habib, Z. A partial ligt-weight image encryption scheme. In Proceedings of the 2019 UK/China Emerging Technologies (UCET), Glasgow, UK, 21–22 August 2019; pp. 1–3.
51. Qayyum, A.; Ahmad, J.; Boulila, W.; Rubaiee, S.; Arshad; Masood, F.; Khan, F.; Buchanan, W.J. Chaos-Based Confusion and Diffusion of Image Pixels Using Dynamic Substitution. *IEEE Access* **2020**, *8*, 140876–140895. [CrossRef]
52. Mujeeb, U.R.; Arslan, S.; Kashif, H.K.; Mohammad, M.H. Efficient and secure image encryption using key substitution process with discrete wavelet transform. *J. King Saud Univ.-Comput. Inf. Sci.* **2023**, *35*, 101613.
53. Wang, X.Y.; Chen, X. An image encryption algorithm based on dynamic row scrambling and Zigzag transformation. *Chaos Solitons Fractals* **2021**, *147*, 110962. [CrossRef]
54. Wang, X.Y.; Xue, W.H.; An, J.B. Image encryption algorithm based on LDCML and DNA coding sequence. *Multimed. Tools Appl.* **2021**, *80*, 591–614. [CrossRef]
55. Yousif, S.F.; Abboud, A.J.; Alhumaima, R.S. A new image encryption based on bit replacing, chaos and DNA coding techniques. *Multimed. Tools Appl.* **2022**, *81*, 27453–27493. [CrossRef]
56. Wang, X.Y.; Zhang, M.Z. A new image encryption algorithm based on ladder transformation and DNA coding. *Multimed. Tools Appl.* **2021**, *80*, 13339–13365. [CrossRef]
57. Zefreh, E.Z. An image encryption algorithm based on a hybrid model of DNA computing, chaotic systems and hash functions. *Multimed. Tools Appl.* **2020**, *79*, 24993–25022. [CrossRef]
58. Xie, H.W.; Gao, Y.J.; Zhang, H. An image encryption algorithm based on novel block scrambling algorithm and Josephus sequence generator. *Multimed. Tools Appl.* **2022**, *82*, 16431–16453. [CrossRef]