mathematics

MDPI

Article

Image Encryption Scheme Based on Newly Designed Chaotic Map and Parallel DNA Coding

Shenli Zhu [1,2], Xiaoheng Deng [2,*], Wendong Zhang [1] and Congxu Zhu [2,*]

[1] Software School, Xinjiang University, Urumqi 830091, China
[2] School of Computer Science and Engineering, Central South University, Changsha 410083, China
* Correspondence: dxh@csu.edu.cn (X.D.); zhucx@csu.edu.cn (C.Z.); Tel.: +86-135-4968-3946 (C.Z.)

Abstract: In this paper, a new one-dimensional fractional chaotic map is proposed and an image encryption scheme based on parallel DNA coding is designed by using the chaotic map. The mathematical model of the new chaotic system combines a sine map and a fraction operation. Compared with some traditional one-dimensional chaotic systems, the new chaotic system has a larger range of chaotic parameters and better chaotic characteristics, which makes it more suitable for applications in information encryption. In addition, an image encryption algorithm based on parallel DNA coding is proposed, which overcomes the shortcoming of common DNA coding-based image encryption algorithms. Parallel computing significantly increases the speed of encryption and decryption algorithms. The initial key of the cryptosystem is designed to be related to the SHA-3 hash value of the plaintext image so that the algorithm can resist a chosen-plaintext attack. Simulation experiments and security analysis results show that the proposed image encryption scheme has good encryption performance and less time overhead, and has strong robustness to noise and data loss attacks, which indicates that the proposed image encryption scheme has good application potential in secure communication applications.

Keywords: image encryption; chaotic system; parallel computing; DNA coding

MSC: 37E05; 68P30

Citation: Zhu, S.; Deng, X.; Zhang, W.; Zhu, C. Image Encryption Scheme Based on Newly Designed Chaotic Map and Parallel DNA Coding. *Mathematics* **2023**, *11*, 231. https://doi.org/10.3390/math11010231

Academic Editor: Lingfeng Liu

Received: 27 November 2022
Revised: 29 December 2022
Accepted: 30 December 2022
Published: 2 January 2023

1. Introduction

Chaos is a kind of stochastic complex phenomenon produced by deterministic nonlinear systems [1]. Chaos exists in many activities of nature and human society. Since the discovery of chaos in nonlinear systems, the study of chaos has become one of the core contents of nonlinear science and has attracted the strong attention of researchers in many fields. The characteristics of chaotic signals have many similarities with the properties required by cryptography, such as the pseudo-randomness of a chaotic signal, the extreme sensitivity to initial conditions and system parameters, and the highly complex nonlinearity, which are much needed by cryptography. This has made chaotic systems widely used to construct cryptographic systems in recent years. For image encryption, chaotic cryptography shows obvious advantages and it has become the main field of chaos application. This is attributed to the following two reasons: On the one hand, the image information has a large amount of data, strong correlation and redundancy, the traditional cryptographic algorithm is inefficient to encrypt the image, and the time cost is particularly high [2]. On the other hand, image information is very widely used visual information—80% of the information acquired by humans comes from vision. Therefore, image information encryption is particularly important. Thus, the research direction of cryptography based on chaos and chaotic secure communication was born. With the development of chaotic cryptography, many image encryption algorithms based on chaos have been proposed [3–9]. In addition, designing new chaotic system models with better performances has also become a hot topic [10–13].

In the form of a mathematical model, chaotic systems can be classified into discrete time iterative mapping systems and continuous time differential equation systems. For the continuous time differential equation systems, only those systems with more than three dimensions may be chaotic. However, for the discrete time iterative mapping systems, it is possible for the one-dimensional mapping system to produce chaos. From the application of chaotic systems in the field of information security, the low-dimensional systems have two advantages over high-dimensional systems. First, low-dimensional systems are easier to implement physically. Second, the time cost to generate a chaotic sequence is shorter, which is more suitable for a real-time encryption system with a large data volume. Moreover, some studies [14,15] confirmed that the complexity of discrete systems is higher than that of continuous systems. Therefore, the design of an efficient one-dimensional discrete chaotic system model has attracted the attention of many researchers, and many image encryption algorithms based on a one-dimensional discrete chaotic system model have been proposed. In Ref. [16], Wang et al. proposed a stream encryption scheme for wireless body area networks based on a logistic map. Midoun et al. [17] proposed a sensitive dynamic mutual image encryption scheme by using a novel one-dimensional chaotic system based on the fraction of cosine over sine. Alawida et al. [18] proposed an image encryption algorithm by using a new hybrid digital chaotic system. Zhu et al. [11] proposed a new image encryption algorithm by using a 1D sinusoidal-polynomial composite chaotic system. Liu et al. [10] proposed a novel image encryption algorithm based on a class of 1D quadratic chaotic maps. Although one-dimensional discrete chaotic systems are very convenient for image encryption, some one-dimensional discrete chaotic systems have defects, such as the parameter interval of chaotic behavior being narrow and discontinuous, there being only one system parameter and the key space being small, and the chaotic performance of some one-dimensional map not being strong. Therefore, it is necessary to develop new chaotic systems with better chaotic performance.

Since Adleman proposed DNA computing [19], DNA computing technology has attracted widespread attention from researchers. In recent years, information encryption based on DNA coding has been introduced into the field of image encryption, and many image encryption algorithms combining chaos and DNA coding have been proposed. Zang et al. [1] proposed an image encryption scheme based on a 1D uniformly distributed discrete chaotic map and DNA coding, in which the DNA encoding and decoding rules are determined by plain text. Hu et al. [20] proposed an image encryption scheme combining chaos with a cycle operation and DNA sequences. Zhang et al. [21] proposed an image encryption algorithm based on a 1D compound sine-piecewise linear chaotic map and varying DNA coding. Lu et al. [22] proposed an image encryption algorithm based on a new conservative hyperchaotic system with DNA coding. Yan et al. [23] proposed an image encryption scheme based on an arithmetic sequence scrambling model, DNA coding sequence and the 1D logistic map. Chai et al. [24] proposed a novel image encryption scheme based on DNA sequence operations, the 2D logistic—adjusted—sine map and two 1D chaotic systems.

Motivated by the above analysis, a new one-dimensional fractional chaotic map has been designed and a novel image encryption scheme based on parallel DNA coding is proposed in this paper. The main contributions of this paper are as follows:

(1) A new one-dimensional chaotic system model with large chaotic range and an unbroken periodic interval is proposed.
(2) A fast image encryption algorithm based on parallel DNA encoding and decoding is proposed.
(3) The complex chaotic characteristics of the chaotic system model are proved by a series of complexity criteria, and the security of the proposed image encryption scheme is verified by a large number of experiments and a security analysis.

The rest of this paper is organized as follows. The new one-dimensional fractional chaotic map is proposed and its performances are presented in Section 2. In Section 3, a novel image encryption algorithm based on the new chaotic map and parallel DNA coding

is proposed. The experimental results and security analysis of the encryption scheme are provided in Section 4. Finally, the conclusion is given in Section 5.

2. The Newly Designed Chaotic Map

Let $f(x)$ be a function defined in interval (0, 1). Then, a new one-dimensional (1D) map with a simple structure is presented in this Section. Its mathematical model is described by Formula (1).

$$x_{n+1} = f(x_n) = \frac{\mu \times \sin(\pi x_n)}{e x_n + \mu}, x_n \in (0, 1), \tag{1}$$

where μ and e are controlling parameters of the system. In the system Equation (1), we adopt the iterative function of fractional structure to separate the points of adjacent orbits in the state space (phase space) of the dynamical system. The function is bounded by the sine function. By limiting the system parameter to be positive, when the state variable takes its value in the interval of (0, 1), the output value range of the function will also be bound to be limited in the interval range of (0, 1). Thus, the state space boundary condition of a chaotic system is satisfied. Additionally, the time sequence generated by the system (1) with parameters $\mu = 6$ and $e = 0.0001$ is depicted in Figure 1a, and the phase diagram of system (1) is depicted in Figure 1b. It can be seen from Figure 1 that the output values produced by the map (1) oscillate in the interval (0, 1), rather than tending to some fixed points when the iterative number increases, indicating that the presented map is of ergodicity.

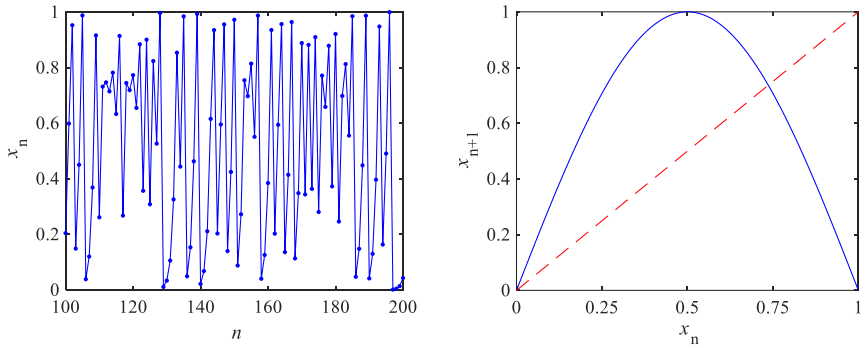


Figure 1. The time sequence and the phase diagrams of system (1): (**a**) The time sequence generated with system (1); (**b**) The phase diagram of system (1).

2.1. Boundedness Analysis

From Equation (1), one can obtain the following results:

(1) If $x_n = 0$ or $x_n = 1$, then $f(x_n) = 0$. If $0 < x_n < 1$, then $f(x_n) > 0$. The fact indicates that function $f(x_n)$ has a lower bound in the domain of $x_n \in (0, 1)$.

(2) Since $\sin(\pi x_n) \leq 1$, therefore $f(x_n) \leq \mu \times 1/(ex_n + \mu) < 1$. This indicates that the function $f(x_n)$ has an upper bound of $\mu/(ex_n + \mu) < 1$ when $x_n \in (0, 1)$.

2.2. Fixed Point and Its Stability Analysis

The derivative of $f(x)$ at point x is:

$$f'(x) = \frac{\mu\pi \cos(\pi x)}{ex + \mu} - \frac{\mu e \sin(\pi x)}{(ex + \mu)^2}. \tag{2}$$

Let $\hat{x}$ be the fixed point (or call equilibrium point) of system (1), then $f(\hat{x}) = \hat{x}$ hold. Thus $\hat{x}$ meets the following condition:

$$\frac{\mu \times \sin(\pi \hat{x})}{e\hat{x} + \mu} = \hat{x}. \tag{3}$$

Considering that the fixed point $\hat{x}$ satisfies Equation (3), the derivative of the function $f(x)$ at the fixed point $\hat{x}$ is:

$$f'(\hat{x}) = \frac{\mu\pi\cos(\pi\hat{x}) - e\hat{x}}{e\hat{x} + \mu}. \tag{4}$$

Proposition 1. *Given the assumption $\mu > e/2$, then the equilibrium point of system (1) must satisfy $\hat{x} > 0.5$.*

Proof. (1) Assuming $\hat{x} = 0.5$, then we can get $\mu = e/2$ from Equation (3). This is contrary to the assumption, so $\hat{x}$ cannot be equal to 0.5.
(2) Assuming $\hat{x} < 0.5$, then we can get $f(\hat{x}) < \frac{\mu\times 1}{e\hat{x}+\mu} = \frac{1}{(e/\mu)\hat{x}+1}$. Since $\hat{x}$ is the equilibrium point and $\hat{x} < 0.5$, then $f(\hat{x}) < 0.5$, namely, $\frac{1}{(e/\mu)\hat{x}+1} < 0.5$, that is, $(e/\mu)\hat{x} > 1$. Since $(e/\mu) < 2$, $\hat{x} < 0.5$, so $(e/\mu)\hat{x} < 1$, which leads to contradictory conclusions. Therefore, it is impossible for $\hat{x} < 0.5$. □

To sum up, the equilibrium point of system (1) can only be $\hat{x} > 0.5$. This conclusion can also be seen intuitively from Figure 1b. The point where the phase diagram intersects the 45 degree line is the equilibrium point.

Proposition 2. *The fixed points $\hat{x}$ of system (1) are unstable if $\hat{x} > 0.604$.*

Proof. According to proposition 1, the equilibrium point $\hat{x}$ of the system (1) satisfies the condition $\hat{x} > 0.5$. So we get the following result $\cos(\pi\hat{x}) < 0$. Then we have

$$|f'(\hat{x})| = \left|\frac{-\mu\pi|\cos(\pi\hat{x})| - e\hat{x}}{e\hat{x} + \mu}\right| = \frac{e\hat{x} + \mu\pi|\cos(\pi\hat{x})|}{e\hat{x} + \mu}. \tag{5}$$

If $\hat{x} > 0.604$, then $\pi|\cos(\pi\hat{x})| > 1$, and $|f'(\hat{x})| > 1$. That is, $|f'(\hat{x})| = \lim_{\Delta x_n \to 0} |\Delta x_{n+1}/\Delta x_n|_{x_n=\hat{x}} > 1$ holds at any fixed points of $\hat{x} > 0.604$, which means that $|\Delta x_{n+1}| > |\Delta x_n|$. The results prove that the map becomes unstable at any fixed points of $\hat{x} > 0.604$. The proof is over. □

According to Ref. [25], when all the fixed points of a dynamical system are unstable, the system is chaotic. So, Proposition 1 actually proves that system (1) is chaotic.

Considering Equation (3), therefore, the condition of parameters for chaotic behavior appearing in system (1) is as follows:

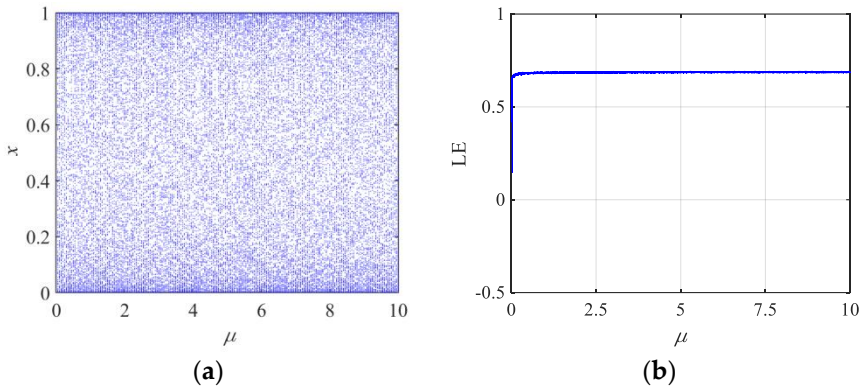
$$\mu = \frac{\hat{x}^2}{\sin(\pi\hat{x}) - \hat{x}}e > \frac{0.604^2}{\sin(0.604\pi) - 0.604}e = 1.063298753415842e. \tag{6}$$

2.3. The Diagram of Bifurcation and Lyapunov Exponent

The bifurcation diagram can intuitively show the number of possible state values of the system under various parameter conditions. If the number of state values is finite, the system is periodic under this parameter. If the number of state values is infinite, the system is chaotic under this parameter. The Lyapunov exponent describes the property of a system numerically. A positive Lyapunov exponent means chaos. For one-dimensional maps, the Lyapunov exponent can be calculated by the following equation:

$$LE = \lim_{N\to\infty} \frac{1}{N}\left(\sum_{n=1}^{N} \ln\left|f'(x_n)\right|\right), \tag{7}$$

where $f'(x_n)$ is the value of the derivative of the mapping function at x_n. N is an integer that is large enough. Figure 2a is the bifurcation diagram of the state quantity of system (1) changing with parameter μ. Figure 2b is the graph of the Lyapunov exponent of system (1) changing with parameter μ, where e is set to a fixed value of 0.0001.



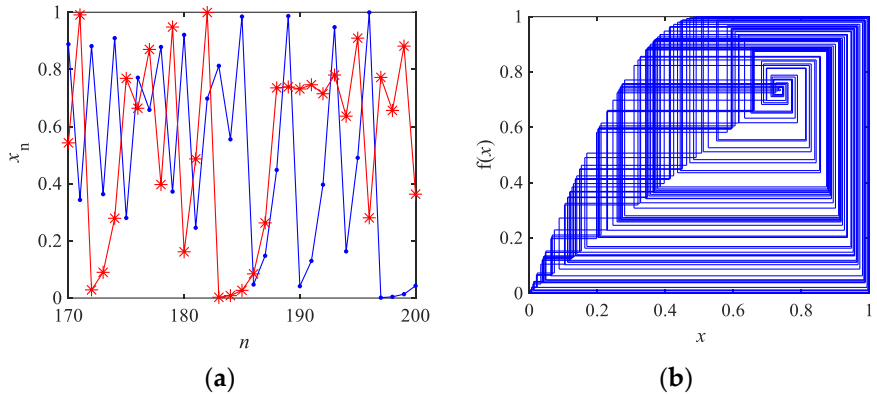
(**a**) (**b**)

Figure 2. Bifurcation diagram and Lyapunov exponent graph of system (1): (**a**) Bifurcation diagram of system state quantity versus parameter μ; (**b**) The graph of Lyapunov exponent versus parameter μ.

It can be seen from the results in Figure 2 that the system (1) presents chaos in a large parameter interval of $\mu > 1.063298753415842 \times e = 1.063298753415842 \times 10^{-4}$. Moreover, unlike the traditional logistic map or sine map, system (1) does not have a small periodic window intermittently in the chaotic region, but the chaotic phenomenon occurs continuously with the change of parameter μ, and its Lyapunov exponent is always positive in the chaotic region. Therefore, system (1) presents a robust chaotic phenomenon in a large parameter range.

2.4. The Time-Series Diagram and Cobweb Graph

Figure 3a shows two time series generated by system (1). The difference between their initial state values is only 10^{-12}. It can be seen that the orbits formed by the subtle differences in the initial state after long-term iteration are very different, indicating that system (1) is extremely sensitive to the initial conditions. Figure 3b is a cobweb diagram generated by system (1), that is, given any initial value x_0 belonging to (0, 1), the orbits generated by the system after long-time iteration are infinite chaotic orbits. The above facts further confirm the chaos of system (1). The parameters used in Figure 3 are: $e = 0.0001$ and $\mu = 6$.



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

Figure 3. The time-series diagram and cobweb graph of system (1): (**a**) The time-series diagram, the blue dot data is generated from the initial value 0.23, and the red asterisk data is generated from the initial value $0.23 + 10^{-12}$; (**b**) the cobweb graph.

2.5. Correlation Analysis

The correlation of time series can be described by a correlation coefficient. The auto-correlation and cross-correlation of time series with good randomness are the δ function

and zero, respectively. The autocorrelation coefficient at lag k of a series $\{x(i)\}$ of length N is normally given as:

$$autocorr(k) = \frac{\sum_{i=1}^{N-|k|}(x(i)-\bar{x})(x(i+|k|)-\bar{x})}{\sum_{i=1}^{N-|k|}(x(i)-\bar{x})^2} \tag{8}$$

where $\bar{x}$ is the average value of the series $\{x(i)\}$.

The cross-correlation of two series $\{x(n)\}$ and $\{y(i)\}$ of length N at lag k is defined as:

$$crosscorr(k) = \frac{\sum_{i=1}^{N-|k|}(x(i)-\bar{x})(y(i+|k|)-\bar{y})}{\sqrt{\sum_{i=1}^{N-|k|}(x(i)-\bar{x})^2}\sqrt{\sum_{i=1}^{N-|k|}(y(i)-\bar{y})^2}} \tag{9}$$

where $\bar{y}$ is the average value of the series $\{y(i)\}$.

For system (1), the autocorrelation coefficient curve of the chaotic sequence $\{x(i)\}$ generated with the system parameters $e = 0.0001$, $\mu = 6.00$, and initial state value $x(0) = 0.23$ is shown in Figure 4a, and the cross-correlation coefficient curve of two chaotic sequences $\{x(i)\}$ and $\{y(i)\}$ generated with $e = 0.0001$, $\mu = 6.00$, and $y(0) = 0.27$ is shown in Figure 4b.

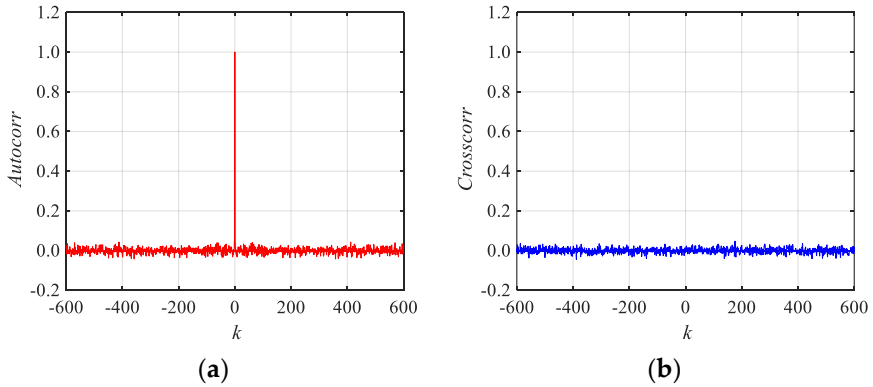


Figure 4. Correlation function of system (1): (**a**) Autocorrelation function; (**b**) Cross-correlation function.

2.6. Approximate Entropy Analysis

Approximate entropy (ApEn) is also a means to measure the complexity of a time series. Approximate entropy describes the probability of generating new patterns in a sequence with the increase of the embedding dimension [10]. Suppose a time series of length N is $\mathbf{u} = [u(1), u(2),..., u(N)]$ and, given a positive real number r (called the time delay) and a positive integer m (called the embedding dimension), the algorithm of calculating ApEn can be described as follows:

Select (N-m + 1) subsequences $\mathbf{v}$ of length m from the sequence $\mathbf{u}$:

$$\mathbf{v}^{(i)} = [u(i), u(i+1), \ldots, u(i+m-1)], i = 1, 2, \ldots, N-m+1.$$

The distance $d(i, j)$ between the vector $\mathbf{v}^{(i)}$ and $\mathbf{v}^{(j)}$ is defined as:

$$d(i,j) = \max(|u(i+k-1) - u(j+k-1)|_{k=1,2,\ldots,m})$$

Calculate the following scalars $C_i^m(r)$ for a fixed i:

$$C_i^m(r) = \frac{\#\{\mathrm{d}(i,j) < r\}}{(N-m+1)},$$

where $\#\{\mathrm{d}(i,j) < r\}$ represents the number of j satisfying the condition $\mathrm{d}(i,j) < r$.

Then, calculate the following scalars $\phi^m(r)$:

$$\phi^m(r) = \frac{\sum_{i=1}^{N-m+1} \log_e[C_i^m(r)]}{N-m+1}.$$

Finally, the approximate entropy of sequence **u** corresponding to (r, m) is obtained:

$$ApEn(\mathbf{u}, r, m) = \phi^m(r) - \phi^{m+1}(r).$$

In the test, we introduce three comparable chaotic maps as follows. The logistic map [26], the sine map [27], and the quadratic map [10]. Then, the *ApEn* values of the chaotic systems (1) and the comparable chaotic maps are drawn in Figure 5. From Figure 5, one can see that system (1) has more stable approximate entropy values and a larger parameter change range $\mu > 1.063298753415842 \times e$. Therefore, the complexity of the time series generated with system (1) is larger than that of other comparable maps.

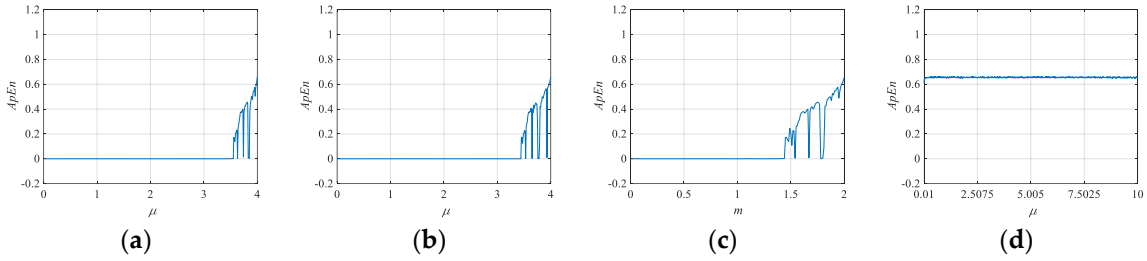


Figure 5. The values of Approximate entropy for different maps. (**a**) Logistic map; (**b**) Sine map; (**c**) Quadratic map; (**d**) The proposed system.

2.7. Correlation Dimension Analysis

A correlation dimension (CD) is applied to measure the geometric complexity of chaotic attractors in dynamical systems [11]. It can not only quantify the self-similarity of chaotic attractors, but also distinguish chaotic sequences from periodic sequences. Let **X** = {X(1), X(2),} be a time series and E be the embedded dimension, then the CD of the **X** is computed by:

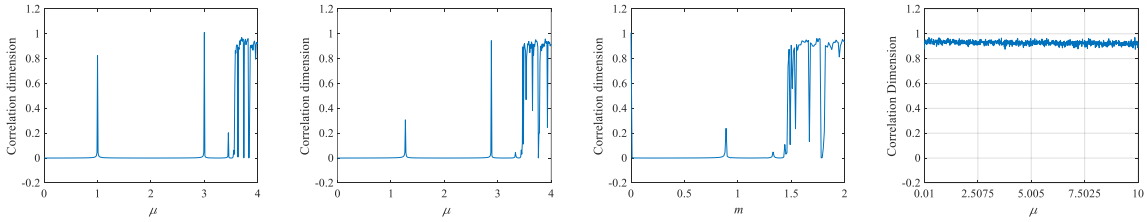
$$d = \lim_{r \to 0} \lim_{N \to \infty} \frac{\log C_E(r)}{\log r},$$

where

$$C_E(r) = \lim_{N \to \infty} \frac{\sum_{i=1}^{N-(E-1)\zeta} \sum_{j=i+1}^{N-(E-1)\zeta} \theta(r - |\bar{s}_i - \bar{s}_j|)}{[N-(E-1)\zeta][N-(E-1)\zeta-1]}.$$

$\theta(\cdot)$ is the heaviside step function and ζ is the time delay. $\bar{s}_i = (s_i, s_{i+\zeta}, s_{i+2\zeta}, \ldots, s_{i+(E-1)\zeta})$, $i = 1, 2, \ldots, N-(E-1)\zeta$. Generally, the embedding dimension $E = 2$ and time delay $\zeta = 1$ for a 1D system [28].

In this test, we use the toolbox function correlationDimension( ) in Matlab to calculate the correlation dimension of time series generated by system (1) with different parameters b; the results are presented in Figure 6. Figure 6 also shows the three comparable chaotic maps: the logistic map [26], the sine map [27], and the quadratic map [10]. As shown in Figure 6, system (1) shows chaotic behavior when $\mu > 1.063298753415842 \times e$. It can be seen that system (1) has higher maximum correlation dimension values with $\mu > 1.063298753415842 \times e$. Therefore, the phase space of the system (1) has a better chaotic performance than the three comparable maps.



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

Figure 6. The values of the correlation dimension for different maps. (**a**) Logistic map; (**b**) Sine map; (**c**) Quadratic map; (**d**) The proposed system.

2.8. NIST Test of the New Chaotic Map

NIST (National Institute of Standards and Technology, USA) is a standard test software package to evaluate the stochastic performance of time series. It requires multiple sequences to be tested and the length of each sequence is 1,000,000 bits. There are two performance indicators, p-value and pass rate, which are iemployed to measure the random performance of time series. The default significant level is $\alpha = 0.01$. The confidence interval used to test the pass rate is defined as: $1 - \alpha \pm 3\sqrt{\alpha(1-\alpha)/m}$, where m is the number of groups of bit sequences. When $\alpha = 0.01$ and $m = 100$, the confidence interval is $1 - 0.01 \pm 3\sqrt{0.01 \times 0.99/100} = 0.99 \pm 0.0094393 = [0.96, 1.0198]$, which indicates that the minimum passing rate must be 96%.

To test the stochastic performance of sequences generated by the proposed chaotic map, we generated 100 chaotic real number sequences each with a length of 125,000 real numbers. The parameters are set as $e = 0.0001$, $\mu = 6.0$, $x_0 = 0.23721$. We transform each real value to a 64-bit binary string following the IEEE 754 double precision floating point number standard. Then, the binary digital numbers from low 17-*th* to 24-*th* bit in each binary string are sampled. One hundred groups of a sequence, each with length of 1,000,000 bits are obtained for the NIST test. The experimental results are listed in Table 1. It can be seen from the test results that each p-value is larger than 0.01 and the minimum pass rate for each statistical test is 96%, which is for the Runs test. The experimental results show that all the chaotic sequences generated with system (1) pass the NIST test.

Table 1. NIST statistical test results for the generated chaotic sequences.

NNIST Statistical Test Item	p-Value	Pass Rate	Results
Frequency (monobit)	0.867692	99/100	Pass
Block Frequency (m = 128)	0.122325	99/100	Pass
Cumulative Sums (Forward)	0.319084	100/100	Pass
Cumulative Sums (Reverse)	0.224821	99/100	Pass
Runs	0.181557	96/100	Pass
Longest Run of Ones	0.419021	99/100	Pass
Rank	0.637119	99/100	Pass
FFT	0.289667	99/100	Pass
Non-Overlapping Templates (m = 9, B = 000000011)	0.014550	97/100	Pass
Overlapping Templates (m = 9)	0.030806	99/100	Pass
Universal	0.437274	99/100	Pass
Approximate Entropy (m = 10)	0.935716	100/100	Pass
Random-Excursions (X = 4)	0.022503	65/66	Pass
Random-Excursions Variant ($X = -9$)	0.671779	66/66	Pass
Serial Test 1 (m = 16)	0.019188	98/100	Pass
Serial Test 2 (m = 16)	0.719747	100/100	Pass
Linear Complexity	0.334538	99/100	Pass

3. The Proposed Image Encryption Scheme

The image encryption scheme includes the following five kinds of process: chaotic secret key streams generating parallel DNA encoding, DNA permutation, parallel DNA decoding, and parallel pixel encryption. The block diagram of the overall image encryption scheme is shown in Figure 7, where **P** and **C** are the plaintext image and encrypted image, respectively. **Pc**, **Pd**, and **Pe** are intermediate ciphertext images. h is the SHA3-256 hash value of the plaintext image **P**. M and N are the row number and column number of the image **P.** μ and e are the parameters of the chaotic system (1). $\{x_{01}, y_{01}, s_{01}, t_{01}\}$ are the initial state values of the chaotic system (1). $\{\mathbf{r}_1, \mathbf{r}_2, \mathbf{J}, \mathbf{K}\}$ are secret key sequences generated by chaotic system (1). The function of module "SHA-3 hash" is to generate the hash value h of the plaintext image **P**. The function of module "Chaotic System (1)" is to generate the key sequences of $\{\mathbf{r}_1, \mathbf{r}_2, \mathbf{J}, \mathbf{K}\}$ by using chaotic system (1). The function of module "Parallel DNA Encoding" is converting image pixel values to DNA code by using $\mathbf{r}_1$. The function of module "DNA Permutation" is shuffling the position of DNA code in parallel. The function of module "Parallel DNA Decoding" is decoding the DNA code in parallel. The function of module "Parallel Pixel Encryption" is encrypting pixel values in parallel. The detailed descriptions of each step are given in Sections 4.1–4.5 below.

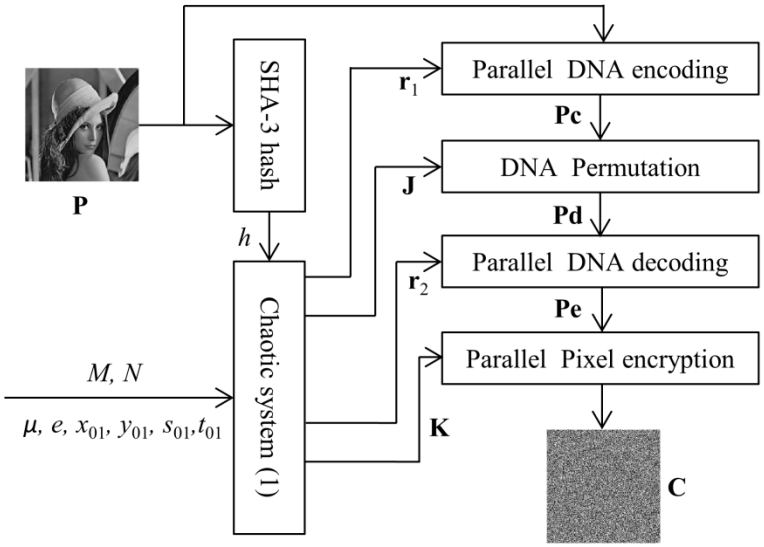


Figure 7. The block diagram of the overall image encryption scheme.

3.1. Chaotic Secret Key Streams Generating

The steps to generate chaotic secret key streams are as follows:

Step 1: The plaintext image to be encrypted is input to the SHA-3 algorithm to generate an SHA3-256 hash value h with a length of 64 hexadecimal characters: $h = h_1 h_2 \ldots h_{64}$.

Step 2: The 64 hexadecimal characters are transformed to 32 positive decimal integers, say, $num(1)$, $num(2)$,..., and $num(32)$, where, $num(i)$ = hex2dec($h_{2i-1}h_{2i}$), i=1, 2,..., 32.

Step 3: The 32 decimal integers are transformed to four fraction numbers of $\{x_{02}, y_{02}, s_{02}, t_{02}\}$ as:

$$\begin{cases} x_{02} = \sum_{i=1}^{8} num(i)/(8 \times 256) \\ y_{02} = \sum_{i=9}^{16} num(i)/(8 \times 256) \\ s_{02} = \sum_{i=17}^{24} num(i)/(8 \times 256) \\ t_{02} = \sum_{i=25}^{32} num(i)/(8 \times 256) \end{cases} \quad (10)$$

Step 4: From the input parameters $\{x_{01}, y_{01}, s_{01}, t_{01}\}$ and $\{x_{02}, y_{02}, s_{02}, t_{02}\}$, we get final initial state values $\{x_0, y_0, s_0, t_0\}$ of the chaotic system (1) as:

$$\begin{cases} x_0 = (x_{01} + x_{02})/2 \\ y_0 = (y_{01} + y_{02})/2 \\ s_0 = (s_{01} + s_{02})/2 \\ t_0 = (t_{01} + t_{02})/2 \end{cases} \tag{11}$$

Step 5: To generate chaotic sequences {**X**, **Y**, **S**, **T**} with parameters (μ, e, x_0), (μ, e, y_0), (μ, e, s_0) and (μ, e, t_0) by using chaotic system (1), respectively. The length of **X** is $4M \times N$, and the lengths of the other three sequences are $M \times N$, respectively. The algorithm for generating a chaotic sequence with length L is shown in Algorithm 1.

Algorithm 1. Generating a chaotic sequence with length of L.

Input: Parameters $\{\mu, e, x_0\}$ of chaotic system (1) and L.
Output: The chaotic sequence **X** = $\{X(1), X(2),..., X(L)\}$.
Step 1: $X(1) \leftarrow x_0$.
Step 2: For i = 2:L, do
$X(i) \leftarrow \mu \times \sin(\pi \times X(i-1))/(e \times X(i-1) + \mu)$;
End for
Step 3: Output the chaotic sequence **X** = $\{X(1), X(2),..., X(L)\}$.

Step 6: Modify chaotic sequences $\{X(i)\}$, $\{Y(i)\}$, $\{S(i)\}$, $\{T(i)\}$ to obtain secret key sequences by using the following formulas

$$[\sim, \{J(i)\}] = \text{sort}(\{X(i)\}),\ i = 1, 2,..., 4 \times M \times N. \tag{12}$$

$$K(i) = \text{mod}(\text{floor}(Y(i) \times 10^6), 256),\ i = 1, 2,..., M \times N. \tag{13}$$

$$r1(i) = \text{mod}(\text{floor}(S(i) \times 10^6), 8) + 1,\ i = 1, 2,..., M \times N. \tag{14}$$

$$r2(i) = \text{mod}(\text{floor}(T(i) \times 10^6), 8) + 1,\ i = 1, 2,..., M \times N. \tag{15}$$

where, $J(i) \in \{1, 2,..., 4 \times M \times N\}$, $K(i) \in \{0, 1,..., 255\}$, $r1(i) \in \{1, 2,..., 8\}$, $r2(i) \in \{1, 2,..., 8\}$.

3.2. Parallel DNA Encoding

DNA contains four kinds of bases, namely A (adenine), T (thymine), C (cytosine), and G (guanine). Traditional computer processing of data is usually expressed in the binary form of 0s and 1s. In general, a 2-bit binary number can be represented with a DNA base. Two-bit binary numbers include four different numbers of {00, 01, 10, 11} and each one can be represented by A or C or G or T. So, there are 24 different representation methods in total (4! = 24). Due to the principle of base complementary pairing, A and T are complementary, and C and G are complementary. Hence, there are eight kinds of expressions that conform to the principle, called eight kinds of DNA coding rules, which are shown in Table 2. For a gray image with 256 gray levels, each pixel value is an 8-bit binary number and can be encoded as a DNA sequence with a length of 4. For encoding the pixel value 156 with rule 1 as an example, firstly, the decimal value 156 is changed to the binary form of string '10011100'. Then two digits are extracted from the string from left to right, and the corresponding DNA characters in Table 2 are used to represent the two digits extracted each time. The first (leftmost) two binary digital '10' is at the third row, and rule 1 is at the first column of the character table, and therefore '10' is encoded to 'C'. In the same way, '10' is encoded to 'G', '11' is encoded to 'T', and '00' is encoded to 'A'. So, the pixel value 156 can be encoded as a DNA sequence 'CGTA' by using coding rule 1. Conversely, the DNA

sequence 'CGTA' can be decoded with rule 1 as a binary string '10011100', i.e., the value 156. However, the same DNA sequence 'CGTA' can be decoded with rule 3 as a binary string '11001001', which is the value 201. From the example, we can find the following fact: if the number 156 is encoded with rule 1 then its DNA code is decoded with rule 3, one can obtain a different number, 201. In general, to encode a number p with rule r_1 then decode its DNA code with rule r_2 ($r_2 \neq r_1$), one can obtain a different number p'($p' \neq p$). In this way, it is equivalent to encrypting data p to obtain its corresponding ciphertext p'.

Table 2. DNA coding rules.

Digitals\Rules	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
00	A	A	G	C	G	C	T	T
01	G	C	A	A	T	T	G	C
10	C	G	T	T	A	A	C	G
11	T	T	C	G	C	G	A	A

In order to improve the speed of DNA coding, the following parallel DNA coding algorithms are employed in this paper. The operational steps are as follows.

Step 1: The plaintext image **P** = {$P(i)$} is split into four sub images, **P**1 = {$P1(i)$}, **P**2 = {$P2(i)$}, **P**3 = {$P3(i)$}, and **P**4 = {$P4(i)$}, i = 1, 2,..., MN. $P1(i)$ = bin2dec('$pb(i,1)pb(i,2)$'), $P2(i)$ = bin2dec('$pb(i,3)pb(i,4)$'), $P3(i)$ = bin2dec('$pb(i,5)pb(i,6)$'), $P4(i)$ = bin2dec('$pb(i,7)pb(i,8)$'), where $pb(i,1)$ is the leftmost bit of pixel value $P(i)$ and $pb(i,8)$ is the rightmost bit of pixel value $P(i)$. bin2dec( ) is a function that converts a binary string into a decimal number.

Step 2: Do parallel DNA coding for the four sub images **P**1, **P**2, **P**3 and **P**4, then we obtain four sub images of DNA code **Pc**1, **Pc**2, **Pc**3 and **Pc**4, where $Pc1(i)$, $Pc2(i)$, $Pc3(i)$, $Pc4(i)$∈{'A','G', 'C','T'}, i = 1, 2,..., MN.

Step 3: Combine **Pc**1, **Pc**2, **Pc**3 and **Pc**4 to obtain a DNA coding matrix **Pc**, which has MN rows and four columns. Each row of **Pc is** composed of four DNA encoded characters, namely, $Pc(i, :)$ = [$Pc1(i)$, $Pc2(i)$, $Pc3(i)$, $Pc4(i)$] is a string consisting of four characters. i = 1, 2,..., MN.

3.3. DNA Permutation

The DNA permutation process scrambles the position of each character in the character matrix **Pc**. This process consists of the following two steps.

Step 1: Shuffle the character matrix **Pc** to obtain a shuffled character sequence **Pc**1 = [$Pc1(1)$, $Pc1(2)$,..., $Pc1(4MN)$], where $Pc1(i)$∈{'A', 'G', 'C', 'T'}, i = 1, 2,..., $4MN$. The permutation algorithm employs the position index sequence **J**, which is described by the following formula:

$$Pc1(i) \leftarrow Pc(J(i)), i=1, 2, ..., 4MN. \quad (16)$$

Step 2: Transform **Pc**1 into a character matrix **Pd** of (MN) rows and four columns: **Pd** = reshape(**Pc**1, MN, 4).

3.4. Parallel DNA Decoding

In this process, each row of characters in the character matrix **Pd** is converted to an integer. When the operations on all rows are complete, we can obtain a column vector consisting of MN integers. The operation to decode the i-th row can be described as follows:

Step 1: Considering that the decoding rule of the i-th row characters is $r2(i)$, find the row numbers of the four characters of $Pd(i, :)$ in column $r2(i)$ of the DNA coding table, and the row numbers of each character are obtained, which are denoted by the names $l1$, $l2$, $l3$ and $l4$, respectively. $l1$, $l2$, $l3$, $l4$∈{1, 2, 3, 4}.

Step 2: Use the following formula to calculate the integer corresponding to the i-th row DNA code:

$$Pe(i) = (l1\text{-}1) \times 4^3 + (l2\text{-}1) \times 4^2 + (l3\text{-}1) \times 4 + (l4\text{-}1). \quad (17)$$

For i = 1, 2,..., MN, repeat the above Steps 1 and 2 to obtain the pixel sequence **Pe** of the intermediate version of the encrypted image. The above circulating operations are performed in parallel loop mode.

3.5. Parallel Pixel Encryption

For i = 1, 2,..., MN, perform a bit-wise Xor operation between $Pe(i)$ and the i-th value $K(i)$ in the key sequence **K** to obtain the encrypted pixel value $Pe(i)$, i.e.,

$$Pe(i) = \text{bitxor}(Pe(i), K(i)). \tag{18}$$

The above circulating operations are performed in parallel loop mode. Finally, the pixel sequence $\{Pe(i)\}$ is converted into a matrix of M rows and N columns to obtain the final ciphertext image **C**, i.e., **C** = reshape(**Pe**, M, N).

The operation of the decryption process is the reverse operation of the above encryption process.

4. Experimental Results and Security Analysis

We use MATLAB 2021b to verify the proposed encryption algorithm on a PC with an Intel(R) Core i7-9700 @ 3.00GHz CPU and 16.0 GB memory. The test images are from the famous standard test images databases, CVG-UGR and USC-SIPI. The secret parameters are set as μ = 6.0, e = 0.0001; x_{01} = 0.1323, y_{01} = 0.3217, s_{01} = 0.4126, and t_{01} = 0.8913. For parameters $\{x_{01}, y_{01}, s_{01}, t_{01}\}$, as long as their values are within the range of (0, 1), it is reasonable, while the values of parameters μ and e should meet certain conditions so that system (1) can generate chaos. The encryption results of several test images are shown in Figure 8.

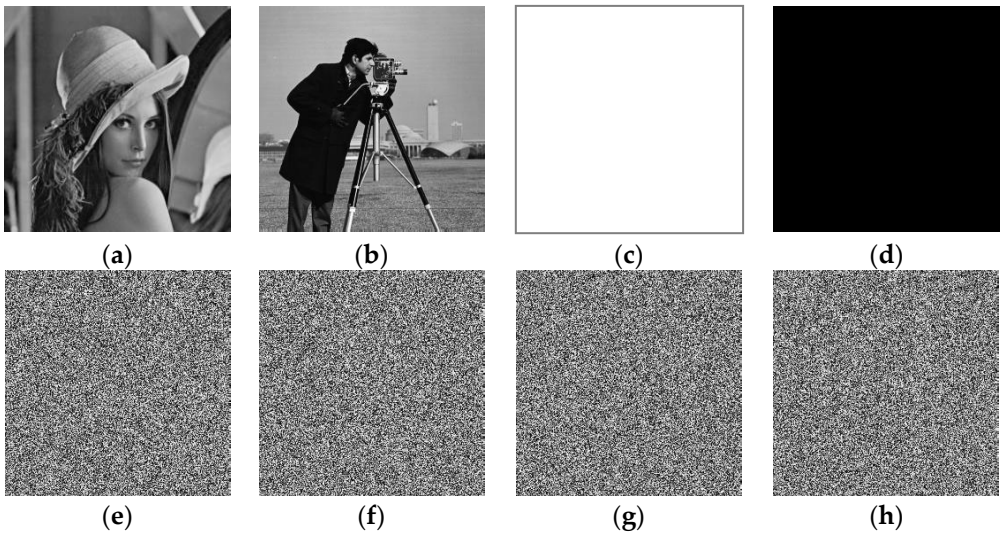


Figure 8. The test images and the encrypted results. (**a**) The image Lena. (**b**) The image Cameraman. (**c**) The all-white image. (**d**) The all-black image. (**e**) The encrypted image Lena. (**f**) The encrypted image Cameraman. (**g**) The encrypted all-white image. (**h**) The encrypted all-black image.

4.1. Key Space Analysis

The performance of a cryptosystem against key exhaustion attacks depends on its key space. The original key of the algorithm in this paper includes the parameters and initial values of the chaotic system. If excluding the system parameter e, the key set contains five double precision parameters $\{b, x_{01}, y_{01}, s_{01}, t_{01}\}$. Each parameter has 15 varied digitals. As a result, the total key space is $10^{15\times5} = 10^{75} > 2^{249}$. At present, a cryptosystem is secure when the key space is greater than 2^{100}, according to Ref. [29]. Therefore, the key space of the proposed scheme is large enough to effectively resist brute force attacks.

4.2. Histogram Analysis

The statistical histogram of an image intuitively shows the distribution of its pixels' values. In fact, the histogram of a real image usually has some kind of non-uniform distribution shape. A good encryption algorithm should ensure the encrypted image has uniformly distributed histograms. Figure 9 shows the statistical histograms of two test images and their encrypted images. By comparison, it can be seen that the encrypted image and its corresponding plaintext image have completely different histograms. In spite of the non-uniform distribution of the original image histogram, the encrypted ciphertext image has a uniform distribution of histograms. Therefore, the encrypted image can effectively resist statistical analysis attacks.

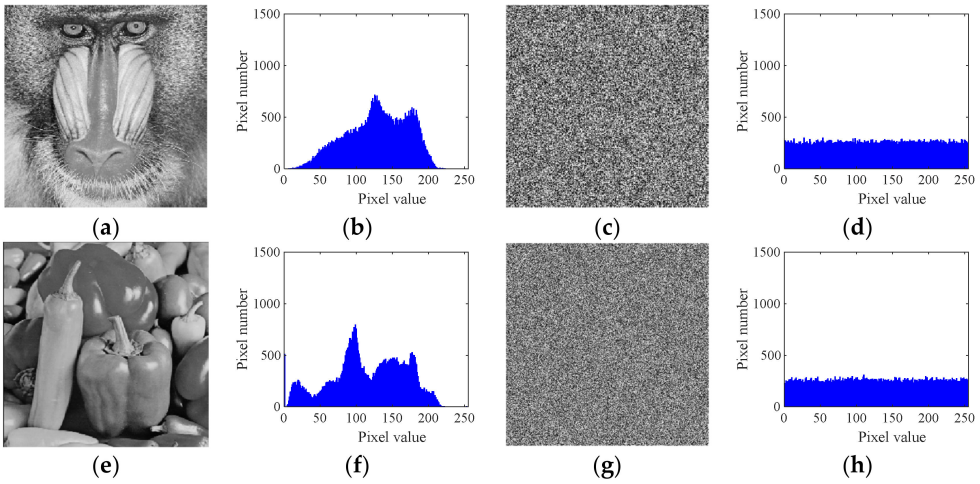


Figure 9. Plaintext/encrypted images and their histograms. (**a**) Plaintext image Baboon. (**b**) Histogram of (**a**). (**c**) Encrypted image Baboon. (**d**) Histogram of (**c**). (**e**) Plaintext image Peppers. (**f**) Histogram (**e**). (**g**) Encrypted image Peppers. (**h**) Histogram of (**g**).

In addition, we can employ the Chi-square test to quantitatively describe the distribution uniformity of the histogram. The Chi-square χ^2 of an image can be calculated by:

$$\chi^2 = \sum_{i=1}^{n} (O_i - E_i)^2 / E_i, \tag{19}$$

where n represents the total gray level of the image, O_i represents the observed occurrence frequency of the i-th gray level, and E_i represents the expected ideal occurrence frequency of the i-th gray level for an image of size $M \times N$, $E_i = M \times N/n$. For a significance level $\alpha = 0.05$, the critical value for an 8-bit gray image ($n = 256$) is equal to $X^2(255, 0.05) = 293.2478$. An encrypted 8-bit gray image should have a value lower than the critical value of 293.2478. We apply the test to several test images and their encrypted images; the test results are listed in Table 3. From Table 3, one can see that all the encrypted images have lower Chi-square values than the critical value, which indicates that the ciphertext images have a uniform distribution. Compared with the results in other works, the images encrypted by this algorithm have a lower Chi-square value in most cases.

Table 3. χ^2 values of images encrypted by different algorithms.

Images	Original Image	This Work	Ref. [30]	Ref. [31]
Lena(256 × 256)	3.0666×10^4	223.2891	230.1484	217.8984
Cameraman(256 × 256)	1.1097×10^5	239.4844	234.3047	219.4609
Lena(512 × 512)	1.5802×10^5	248.8750	239.7539	249.7266
Cameraman(512 × 512)	4.1853×10^5	271.9883	278.0410	261.8965
Barbara(512 × 512)	1.4410×10^5	239.5566	253.9297	227.0996
Mandrill(512 × 512)	1.8760×10^5	244.7559	245.0137	241.0781

4.3. Correlation Analysis

4.3.1. Correlations of Two Adjacent Pixels

For meaningful images, there are very small differences between the adjacent pixels, which means that there is a strong correlation between adjacent pixels. A good encryption algorithm should break this correlation between adjacent pixels. The correlation strength between adjacent pixels can be quantitatively measured by a correlation coefficient. Therefore, the correlation coefficient is introduced in this paper to test the correlation of adjacent pixels of the images, and its calculation formula is as follows:

$$E(\mathbf{x}) = \frac{1}{N_{xy}} \sum_{i=1}^{N_{xy}} x_i \tag{20}$$

$$D(\mathbf{x}) = \frac{1}{N_{xy}} \sum_{i=1}^{N_{xy}} (x_i - E(\mathbf{x}))^2 \tag{21}$$

$$\mathrm{cov}(\mathbf{x}, \mathbf{y}) = \frac{1}{N_{xy}} \sum_{i=1}^{N_{xy}} (x_i - E(\mathbf{x}))(y_i - E(\mathbf{y})) \tag{22}$$

$$r_{\mathbf{xy}} = \mathrm{cov}(\mathbf{x}, \mathbf{y}) / \sqrt{D(\mathbf{x})} \sqrt{D(\mathbf{y})}, \tag{23}$$

where (x_i, y_i) represents the gray value of a group of adjacent pixels in the image, and N_{xy} represents the total number of groups of pixels randomly selected from the image. The smaller the absolute value of the correlation coefficient $r_{\mathbf{xy}}$, the weaker the correlation between the two groups of pixels. Table 4 lists the correlation coefficients between adjacent pixels in different directions of the image encrypted by this algorithm. Table 4 also lists the results of some algorithms published recently. Compared with the results of other algorithms, the proposed algorithm achieves satisfactory results. The method proposed in this paper can achieve better results due to the combination of DNA code scrambling and pixel value encryption. DNA code scrambling not only plays the role of pixel scrambling, but also plays the role of pixel value transformation, which has a stronger effect on breaking the correlation between adjacent pixels.

Table 4. Correlation coefficients of cipher images encrypted by different algorithms.

Algorithm	Image Name	Horizontal	Vertical	Diagonal
This work	5.1.10	0.0090266	−0.0059255	0.0055227
Ref. [32]	5.1.10	−0.002971	−0.000897	0.003682
Ref. [23]	5.1.10	−0.007100	0.008500	0.000200
This work	5.1.11	0.0042465	−0.0074077	−0.0045656
Ref. [32]	5.1.11	0.001757	−0.010444	0.001124
Ref. [23]	5.1.11	−0.004800	−0.001700	0.006800
This work	5.1.12	−0.0041533	−0.0011927	0.0044498
Ref. [32]	5.1.12	0.009575	−0.002502	−0.000582
Ref. [23]	5.1.12	0.005500	−0.004900	0.000100
This work	5.1.13	0.00088813	0.00060857	0.004348
Ref. [32]	5.1.13	0.000347	0.004691	−0.009999
Ref. [23]	5.1.13	0.003800	0.002500	0.003200
This work	5.1.14	−0.0029507	−0.00010549	−0.0016555
Ref. [32]	5.1.14	0.008773	−0.011971	0.000220
Ref. [23]	5.1.14	0.000400	0.000400	0.001200
This work	5.2.08	0.0011946	0.0013694	0.0010658
Ref. [32]	5.2.08	−0.002389	−0.003528	−0.003059
Ref. [23]	5.2.08	0.004100	0.001400	0.000054
This work	5.2.09	−0.0019694	0.00042113	0.0015425
Ref. [32]	5.2.09	0.000783	−0.003316	−0.000207
Ref. [23]	5.2.09	−0.001700	−0.001800	−0.001900

Table 4. *Cont.*

Algorithm	Image Name	Horizontal	Vertical	Diagonal
This work	5.2.10	−0.001072	−0.0014797	0.0051372
Ref. [32]	5.2.10	−0.006168	−0.007614	0.000369
Ref. [23]	5.2.10	0.000007	0.002100	0.001200
This work	7.1.01	0.0024507	−0.0021272	−0.0019601
Ref. [32]	7.1.01	−0.002843	0.000667	0.004116
Ref. [23]	7.1.01	−0.000100	0.001300	−0.001300
This work	7.1.02	−0.001142	-6.4625×10^{-5}	0.0026359
Ref. [32]	7.1.02	−0.003666	−0.001386	−0.001295
Ref. [23]	7.1.02	0.000900	0.001600	0.005700
This work	7.1.03	0.0015743	0.0024114	-4.7521×10^{-5}
Ref. [32]	7.1.03	−0.002931	−0.004124	0.003147
Ref. [23]	7.1.03	0.000100	0.000200	0.003100
This work	7.1.04	−0.001451	-4.903×10^{-5}	0.0035602
Ref. [32]	7.1.04	−0.004028	−0.001065	−0.000901
Ref. [23]	7.1.04	−0.001400	0.000811	−0.003100
This work	7.1.05	−0.005103	0.0019053	0.0020445
Ref. [32]	7.1.05	0.001735	−0.003046	−0.002081
Ref. [23]	7.1.05	−0.002400	−0.000700	0.003400
This work	7.1.06	−0.00022585	0.0040692	0.003082
Ref. [32]	7.1.06	−0.001395	−0.003363	−0.001516
Ref. [23]	7.1.06	0.000832	0.001700	0.001800
This work	7.1.07	−0.0012256	−0.0032455	−0.0044288
Ref. [32]	7.1.07	−0.000608	0.000682	−0.000090
Ref. [23]	7.1.07	0.003900	0.002100	0.002500
This work	5.3.01	-6.562×10^{-5}	-3.2479×10^{-5}	−0.0013003
Ref. [32]	5.3.01	0.000606	0.000090	0.002417
Ref. [23]	5.3.01	0.000400	0.002600	0.001200
This work	5.3.02	0.0011262	−0.00055271	−0.00089431
Ref. [32]	5.3.02	0.000502	0.001669	−0.000435
Ref. [23]	5.3.02	−0.000377	−0.000474	−0.000301

The distribution of adjacent pixel values for image Lena is plotted in Figure 10, which can intuitively display the correlation between adjacent pixels. As can be seen from Figure 10, the adjacent points of the original Lena image are distributed on or near the 45 degree line, indicating that the values of adjacent pixels are equal or close. However, the adjacent points of the ciphertext image are not concentrated on the 45 degree line, indicating that there is a large difference in the values of the adjacent pixels. Therefore, the encryption algorithm effectively breaks the pixel correlation of the image.

4.3.2. Correlations between Original and Cipher-Images

The correlation between various pairs of plaintext-images and ciphertext-images can be described by the 2D correlation coefficients (CC) between the original and encrypted images. The CC are calculated as follows:

$$CC = \frac{\sum_{i=1}^{M_1}\sum_{j=1}^{M_2}(A_{ij}-\overline{A})(B_{ij}-\overline{B})}{\sqrt{\left(\sum_{i=1}^{M_1}\sum_{j=1}^{M_2}(A_{ij}-\overline{A})^2\right)\left(\sum_{i=1}^{M_1}\sum_{j=1}^{M_2}(B_{ij}-\overline{B})^2\right)}}, \quad (24)$$

where $\overline{A} = \frac{1}{M_1 \times M_2}\sum_{i=1}^{M_1}\sum_{j=1}^{M_2} A_{ij}$ and $\overline{B} = \frac{1}{M_1 \times M_2}\sum_{i=1}^{M_1}\sum_{j=1}^{M_2} B_{ij}$. A represents the plaintext-image, B represents the ciphertext-image. M_1 and M_2 are the height and width of the plaintext-image/ciphertext-image, respectively. We have chosen the four traditional images for this

analysis. The *CC* is calculated for the encryption of these images. Table 5 shows the *CC* values of these images. It is clear that the correlation coefficients between various pairs of plaintext-image and ciphertext images are very small. Therefore, there is a great difference between the original images and the encrypted ones.

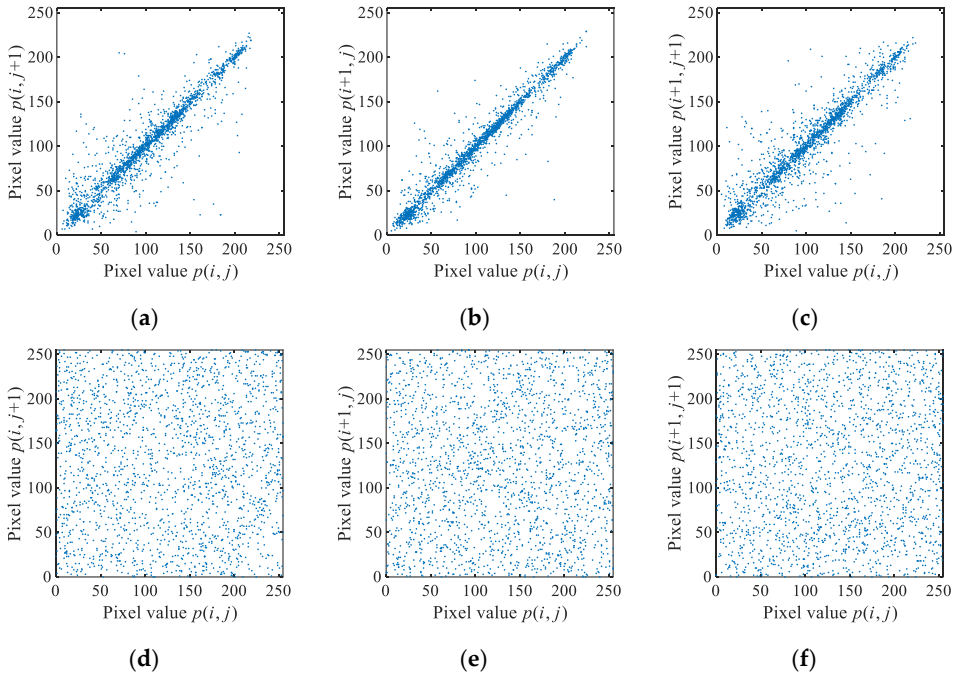


Figure 10. Correlation point diagrams of plaintext image Lena and encrypted image Lena. (**a**) Original Lena in horizontal direction, (**b**) Original Lena in vertical direction, (**c**) Original Lena in diagonal direction, (**d**) Encrypted Lena in horizontal direction, (**e**) Encrypted Lena in vertical direction, (**f**) Encrypted Lena in diagonal direction.

Table 5. The 2D correlation coefficients (CC) with a different approach.

Images	This Paper	Ref. [5]	Ref. [1]	Ref. [33]
Lena	−0.0035	0.002851	−0.0033	0.001744
Baboon	-7.3104×10^{-5}	−0.006632	0.0020	0.005929
Cameraman	0.0026	−0.003558	−0.0025	-
Pepper	−0.0046	−0.001650	0.0043	−0.001332

4.4. MSE and Peak Signal-To-Noise Ratio Analysis

The mean square error (MSE) and the peak signal-to-noise ratio (PSNR) are two parameters for evaluating the reliability of our algorithm. MSE provides the error between an input image and an output image. The *MSE* can be calculated by:

$$MSE = \frac{1}{M_1 \times M_2} \sum_{i=1}^{M_1} \sum_{j=1}^{M_2} [I_o(i,j) - I_e(i,j)]^2, \tag{25}$$

where $I_o(i, j)$ is the pixel value of plain-image, $I_e(i, j)$ is the pixel value of cipher-image, M_1 is the row number and M_2 is the column number. Thus, the *PSNR* is described by the next expression:

$$PSNR = 10 \times \log_{10}[\frac{I_{\max}^2}{MSE}], \tag{26}$$

where $I_{\max}$ is the maximum pixel value of the image. The *PSNR* should be a low value, which corresponds to a great difference between the original image and the encrypted image. To determine the encryption quality, the *PSNR* is computed for the encryption of

the images of Lena, baboon, cameraman and peppers. Table 6 shows the *PSNR* values. The results show that these *PSNR* values are all very low. Therefore, the encryption quality of every image is good.

Table 6. The *MSE* and *PSNR* values of encrypted images with different approach.

Images	MSE				PSNR			
	Ours	**Ref. [5]**	**Ref. [1]**	**Ref. [33]**	**Ours**	**Ref. [5]**	**Ref. [1]**	**Ref. [33]**
lena	9072.7	7760.0	9048.1	8692.3	8.5534	9.2322	8.5652	8.1636
baboon	7188.3	7218.1	7200.6	8325.0	9.5645	9.5466	9.5571	8.9400
cameraman	9400.4	9466.7	9460.6	-	8.3993	8.3688	8.3716	-
peppers	8199.6	8202.4	8093.0	9647.4	8.9929	8.9914	9.0497	7.5889

4.5. Information Entropy Analysis

Information entropy can measure the randomness or uncertainty of an information source. Greater information entropy means that the more random or uncertain the information source is, the more difficult it will be to predict or decipher. The information entropy of an information source can be calculated by the following formula:

$$H(S) = -\sum_{i=1}^{n} p_i \log_2(p_i), \tag{27}$$

where $S = \{s_1, s_2, \ldots, s_n\}$ represents the information source and p_i represents the probability of s_i occurrence. According to the maximum information entropy principle, when each s_i has an equal probability, i.e., $p_i = 1/n$, the information source has the maximum information entropy $\log_2 n$. For the information source of an 8-bit gray-scale image, there are 256 gray levels and n = 256. Therefore, the maximum information entropy that can be reached by the gray-scale image is $\log_2 256 = 8$. Therefore, the closer the information entropy of an encrypted image is to 8, the stronger its uncertainty and the higher its security. The information entropy of various standard test images encrypted by this algorithm and some recently published algorithms are listed in Table 7. The results show that the information entropy of ciphertext images is very close to the ideal value of 8. Moreover, compared with other algorithms, the images encrypted by the proposed algorithm have more advantages in many cases.

Table 7. Information entropy of encrypted images for several different algorithms.

Image Name	Image Size	This Work	Ref. [18]	Ref. [23]	Ref. [32]
5.1.10	256 × 256	7.9976948	7.99720	7.99680	7.99717
5.1.11	256 × 256	7.9970808	7.99730	7.99710	7.96999
5.1.12	256 × 256	7.9975342	7.99540	7.99730	7.99757
5.1.13	256 × 256	7.9971601	7.99630	7.99680	7.99735
5.1.14	256 × 256	7.9972009	7.99730	7.99690	7.99674
5.2.08	512 × 512	7.9993163	7.99920	7.99920	7.99934
5.2.09	512 × 512	7.9993178	7.99900	7.99940	7.99930
5.2.10	512 × 512	7.9992856	7.99870	7.99930	7.99926
7.1.01	512 × 512	7.9992931	7.99800	7.99930	7.99929
7.1.02	512 × 512	7.9992812	7.99490	7.99930	7.99931
7.1.03	512 × 512	7.9992667	7.99830	7.99940	7.99925
7.1.04	512 × 512	7.9992436	7.99850	7.99940	7.99923
7.1.05	512 × 512	7.9993051	7.99880	7.99930	7.99929
7.1.06	512 × 512	7.9993503	7.99900	7.99930	7.99933
7.1.07	512 × 512	7.9993485	7.99870	7.99910	7.99931
7.1.08	512 × 512	7.9993392	7.99880	7.99920	7.99923
7.1.09	512 × 512	7.9993272	7.99850	7.99920	7.99219
elaine.512	512 × 512	7.9992569	7.99930	7.99930	7.99922
5.3.01	1024 × 1024	7.9998174	7.99930	7.99980	7.99983
5.3.02	1024 × 1024	7.9998495	7.99920	7.99990	7.99981
testpat.1k	1024 × 1024	7.9997892	7.98470	7.99980	7.99982

4.6. Sensitivity Analysis

A good encryption algorithm should make ciphertext strongly sensitive to both plaintext and the secret keys. NPCR and UACI are used to measure the sensitivity of a ciphertext image to the plaintext image or secret keys. NPCR and UACI can be calculated by:

$$D(i,j) = \begin{cases} 1, & if\ C(i,j) \neq C'(i,j) \\ 0, & if\ C(i,j) = C'(i,j) \end{cases}, \tag{28}$$

$$\text{NPCR} = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} D(i,j) \times 100\% \tag{29}$$

$$\text{UACI} = \frac{1}{M \times N} \left(\sum_{i=1}^{M} \sum_{j=1}^{N} \frac{|C(i,j) - C'(i,j)|}{255} \right) \times 100\%, \tag{30}$$

where M and N are the row and column numbers of the image. The larger the values of NPCR and UACI, the more sensitive the encryption algorithm is. The ideal values of NPCR and UACI are 99.6094% and 33.4635%, respectively.

For the key sensitivity analysis, each time we tested the NPCR and UACI values between two ciphertext images, their corresponding encryption keys had only one parameter difference of 10^{-15}. The experimental results are listed in Table 8. The experimental results show that the values of NPCR and UACI are very close to the ideal values, indicating that the encryption algorithm is very sensitive to each key parameter. Compared with the results in Ref. [5], the key sensitivity of the proposed method in this paper is stronger.

Table 8. Sensitivity of our proposed method with only one key change 10^{-15}.

The Proposed Method in this Paper			The Method Proposed in Ref. [5]		
Changes of Key	**NPCR**	**UACI**	**Changes of Key**	**NPCR**	**UACI**
$\Delta\mu = 10^{-15}$	99.6124	33.3809	$\Delta x_0 = 10^{-10}$	99.6300	33.5100
$\Delta x_{01} = 10^{-15}$	99.6201	33.6414	$\Delta y_0 = 10^{-10}$	99.6000	33.5100
$\Delta y_{01} = 10^{-15}$	99.6262	33.5600	$\Delta z_0 = 10^{-10}$	99.6000	33.5000
$\Delta s_{01} = 10^{-15}$	99.6552	33.5363	$\Delta w_0 = 10^{-10}$	99.6200	33.5100

For the plaintext sensitivity analysis, we tested the NPCR and UACI values between two ciphertext images each time, and the plaintext images corresponding to each group of ciphertext images only had a pixel difference of 1 bit. We select the first position, the middle position and the last position, respectively, to change one pixel of the plaintext image. The experimental results are listed in Table 9. The experimental results show that the values of NPCR and UACI are very close to the ideal values, indicating that the encryption algorithm is very sensitive to plaintext. Compared with the results in Refs. [1,32], the plaintext sensitivity of the proposed method in this paper is satisfactory.

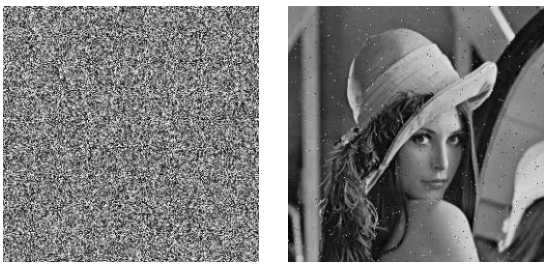
Table 9. Sensitivity of different method for a single pixel change of the original image.

Image	**NPCR**			**UACI**		
	Ours	**Ref. [1]**	**Ref. [32]**	**Ours**	**Ref. [1]**	**Ref. [32]**
Lena	99.6170	98.1644	99.6758	33.4199	31.3312	33.5983
Baboon	99.6399	98.0103	99.6402	33.3027	31.1886	33.6231
Peppers	99.6185	99.9664	-	33.4211	35.6275	-

4.7. Robustness Analysis

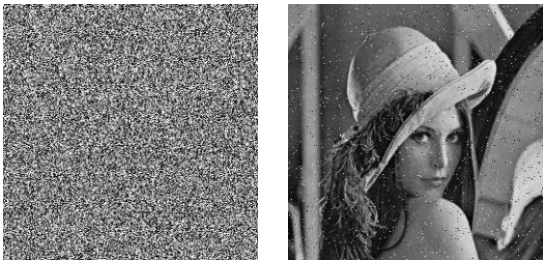
A good encryption algorithm should tolerate partial data loss during the transmission of encrypted images. When the encrypted image is polluted by noise or part of the data is lost, the visually recognizable decrypted image can still be obtained, which means that the encryption algorithm is robust.

To test the performance of the algorithm against noise attack, we added 1%, 3% and 5% salt and pepper noise to the encrypted Lena image. The results of the decrypted images are shown in Figures 11–13. Compared with the algorithm in Ref. [1], when the encrypted image in Ref. [1] suffers from 0.05% salt-and-pepper noise pollution, the decrypted image is worse than that in Figure 13b, indicating that the ability of the proposed algorithm to resist salt-and-pepper noise attack is 100 times that of the algorithm in Ref. [1].



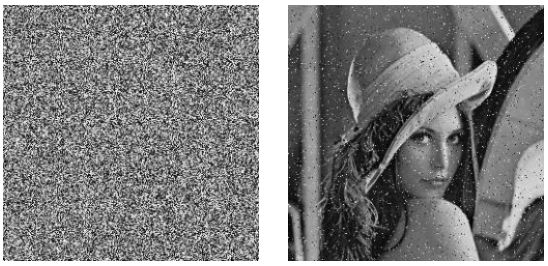
(**a**) (**b**)

Figure 11. Salt and pepper noise intensity of 1%: (**a**) encrypted image and (**b**) decrypted image.



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

Figure 12. Salt and pepper noise intensity of 3%: (**a**) encrypted image and (**b**) decrypted image.



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

Figure 13. Salt and pepper noise intensity of 5%: (**a**) encrypted image and (**b**) decrypted image.

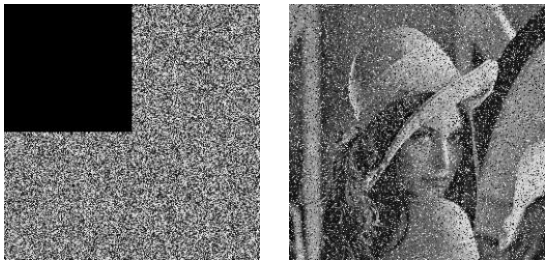
In order to test the performance of the algorithm against data loss, we cut the encrypted image size of 256 × 256 at the upper left corner by 32 × 32, 64 × 64, 128 × 128 sub blocks and then decrypt the cut ciphertext images. The decryption effect is shown in Figures 14–16. The experimental results show that, when the cut area reaches 128 × 128, the algorithm can still restore the original image well. By contrast, the algorithm in Ref. [1] can only tolerate a data loss size of 8 × 16 in ciphertext images at most. When our encrypted image has a data loss size of 128 × 128, the decrypted image is the same as the ciphertext image and the data loss is 8 × 16 of the algorithm in Ref. [1]. It can be seen that the performance of our algorithm against data loss is also much stronger than that of the algorithm in Ref. [1].

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

Figure 14. The removal of a 32 × 32 sub block: (**a**) encrypted image and (**b**) decrypted image.

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

Figure 15. The removal of a 64 × 64 sub block: (**a**) encrypted image and (**b**) decrypted image.



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

Figure 16. The removal of a 128 × 128 sub block: (**a**) encrypted image and (**b**) decrypted image.

4.8. Time Complexity Analysis

The encryption process of the proposed algorithm includes the following five stages: chaotic secret key streams generation, parallel DNA encoding of the image pixels, DNA permutation, parallel DNA decoding of the image pixels, and parallel pixel bit-wise XOR encryption. In the time cost test of the algorithm, the 256 × 256 gray-scale Lena image is used for encryption and decryption. We take the average encryption and decryption time of 10 experiments, and the results are listed in Table 10. Table 10 also lists the time cost data of several algorithms based on DNA coding reported in recently published references. The results demonstrate that the proposed method has faster encryption and decryption speeds than those of the algorithms reported in Refs. [1,20,22,23]. The time complexity data 'are based on the system environment and the software package that is mentioned at the beginning of Section 5.

Table 10. Comparison of encryption and decryption time for 256 × 256 Lena image (second).

Stage	This Work	Ref. [1]	Ref. [20]	Ref. [22]	Ref. [23]
Encryption	0.6007	12.6500	14.8401	1.0770	1.7351
Decryption	0.3804	12.8410	14.9266	1.1144	3.4689

5. Conclusions

In this paper, a new one-dimensional fractional chaotic map is proposed, and an image encryption algorithm based on parallel DNA coding is designed using the chaotic system. For the new chaotic system, we use a series of chaotic performance criteria to prove that it has good chaotic characteristics. Compared with the traditional one-dimensional chaotic systems, the new chaotic system has a larger range of chaotic parameters and more complex chaotic behavior. Aiming at the shortcoming of the image encryption algorithm based on DNA coding, that it consumes a lot of time, an image encryption algorithm based on parallel DNA coding is proposed. Parallel computing can significantly improve the speed of an encryption and decryption algorithm. The initial key of the cryptosystem is designed to be related to the SHA-3 hash value of the plaintext image so that the algorithm can resist the chosen-plaintext attack. Simulation experiments and security analysis results show that the proposed image encryption scheme has a better encryption performance and less time overhead, and has strong robustness to noise and data loss attacks, which indicates that the proposed image encryption scheme has good application potential for secure communication applications.

The algorithm in this paper also has some shortcomings, that is, the number of encryption rounds is a fixed one. If the application scenario has higher requirements for security, it needs to increase the number of encryption rounds. How to set different encryption rounds conveniently and flexibly is worth further study in the future.

Author Contributions: Conceptualization, S.Z. and C.Z.; methodology, W.Z. and X.D.; software, S.Z.; validation, S.Z., W.Z., X.D. and C.Z.; formal analysis, S.Z.; investigation, C.Z.; resources, W.Z.; data curation, S.Z.; writing—original draft preparation, S.Z.; writing—review and editing, C.Z.; visualization, X.D.; supervision, X.D. and W.Z.; project administration, W.Z.; funding acquisition, W.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by National Natural Science Foundation of China (no. 62172441), the Local Science and Technology Developing Foundation Guided by Central Government (Free exploration project 2021Szvup166), and the Natural Science Foundation of Xinjiang Uygur Autonomous Region (no. 2020D01C033).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Zang, H.; Tai, M.; Wei, X. Image Encryption Schemes Based on a Class of Uniformly Distributed Chaotic Systems. *Mathematics* **2022**, *10*, 1027. [CrossRef]
2. Zhang, S.; Liu, L.; Xiang, H. A Novel Plain-Text Related Image Encryption Algorithm Based on LB Compound Chaotic Map. *Mathematics* **2021**, *9*, 2778. [CrossRef]
3. Malik, D.S.; Shah, T. Color multiple image encryption scheme based on 3D-chaotic maps. *Math. Comput. Simul.* **2020**, *178*, 646–666. [CrossRef]
4. Öztürk, İ.; Kılıç, R. Utilizing true periodic orbits in chaos-based cryptography. *Nonlinear Dynam.* **2021**, *103*, 2805–2818. [CrossRef]
5. Zhu, C.X. A novel image encryption scheme based on improved hyperchaotic sequences. *Opt. Commun.* **2012**, *285*, 29–37. [CrossRef]
6. Zhu, S.; Wang, G.; Zhu, C. A Secure and Fast Image Encryption Scheme based on Double Chaotic S-Boxes. *Entropy* **2019**, *21*, 790. [CrossRef] [PubMed]
7. Chai, X.; Fu, J.; Gan, Z.; Lu, Y.; Zhang, Y. An image encryption scheme based on multi-objective optimization and block compressed sensing. *Nonlinear Dynam.* **2022**, *108*, 2671–2704. [CrossRef]
8. Ye, G.; Liu, M.; Wu, M. Double image encryption algorithm based on compressive sensing and elliptic curve. *Alex. Eng. J.* **2022**, *61*, 6785–6795. [CrossRef]
9. Ye, G.; Wu, H.; Liu, M.; Shi, Y. Image encryption scheme based on blind signature and an improved Lorenz system. *Expert Syst. Appl.* **2022**, *205*, 117709. [CrossRef]

10. Liu, L.; Wang, J. A cluster of 1D quadratic chaotic map and its applications in image encryption. *Math. Comput. Simul.* **2023**, *204*, 89–114. [CrossRef]
11. Zhu, H.; Ge, J.; Qi, W.; Zhang, X.; Lu, X. Dynamic analysis and image encryption application of a sinusoidal-polynomial composite chaotic system. *Math. Comput. Simul.* **2022**, *198*, 188–210. [CrossRef]
12. Zhu, S.; Deng, X.; Zhang, W.; Zhu, C. A New One-Dimensional Compound Chaotic System and Its Application in High-Speed Image Encryption. *Appl. Sci.* **2021**, *11*, 11206. [CrossRef]
13. Rong, X.; Jiang, D.; Zheng, M.; Yu, X.; Wang, X. Meaningful data encryption scheme based on newly designed chaotic map and P-tensor product compressive sensing in WBANs. *Nonlinear Dynam.* **2022**, *110*, 2831–2847. [CrossRef]
14. Sun, K.H.; He, S.B.; He, Y.; Yin, L.Z. Complexity analysis of chaotic pseudo-random sequences based on spectral entropy algorithm. *Acta Phys. Sin.* **2013**, *62*, 010501. [CrossRef]
15. Sun, K.H.; He, S.B.; Yin, L.Z.; Li-Kun, A. Application of FuzzyEn algorithm to the analysis of complexity of chaotic sequence. *Acta Phys. Sin.* **2012**, *61*, 130507.
16. Wang, J.; Han, K.; Fan, S.; Zhang, Y.; Tan, H.; Jeon, G.; Pang, Y.; Lin, J. A logistic mapping-based encryption scheme for Wireless Body Area Networks. *Future Gener. Comput. Syst.* **2020**, *110*, 57–67. [CrossRef]
17. Midoun, M.A.; Wang, X.; Talhaoui, M.Z. A sensitive dynamic mutual encryption system based on a new 1D chaotic map. *Opt. Lasers Eng.* **2021**, *139*, 106485. [CrossRef]
18. Alawida, M.; Samsudin, A.; Sen Teh, J.; Alkhawaldeh, R.S. A new hybrid digital chaotic system with applications in image encryption. *Signal Process.* **2019**, *160*, 45–58. [CrossRef]
19. Adleman, L.M. Molecular computation of solutions to combinatorial problems. *Science* **1994**, *266*, 1021–1024. [CrossRef]
20. Hu, T.; Liu, Y.; Gong, L.-H.; Ouyang, C.-J. An image encryption scheme combining chaos with cycle operation for DNA sequences. *Nonlinear Dynam.* **2017**, *87*, 51–66. [CrossRef]
21. Zhang, S.; Liu, L. A novel image encryption algorithm based on SPWLCM and DNA coding. *Math. Comput. Simul.* **2021**, *190*, 723–744. [CrossRef]
22. Lu, Q.; Yu, L.; Zhu, C. A New Conservative Hyperchaotic System-Based Image Symmetric Encryption Scheme with DNA Coding. *Symmetry* **2021**, *13*, 2317. [CrossRef]
23. Yan, X.; Wang, X.; Xian, Y. Chaotic image encryption algorithm based on arithmetic sequence scrambling model and DNA encoding operation. *Multimed. Tools Appl.* **2021**, *80*, 10949–10983. [CrossRef]
24. Chai, X.; Gan, Z.; Yuan, K.; Chen, Y.; Liu, X. A novel image encryption scheme based on DNA sequence operations and chaotic systems. *Neural Comput. Appl.* **2019**, *31*, 219–237. [CrossRef]
25. Hua, Z.; Zhou, B.; Zhou, Y. Sine Chaotification Model for Enhancing Chaos and Its Hardware Implementation. *IEEE Trans. Ind. Electron.* **2019**, *66*, 1273–1284. [CrossRef]
26. May, R.M. Simple mathematical models with very complicated dynamics. *Nature* **1976**, *261*, 459–467. [CrossRef]
27. Zhou, Y.; Bao, L.; Chen, C.L.P. A new 1D chaotic system for image encryption. *Signal Process.* **2014**, *97*, 172–182. [CrossRef]
28. Ma, Y.; Li, C.; Ou, B. Cryptanalysis of an image block encryption algorithm based on chaotic maps. *J. Inf. Secur. Appl.* **2020**, *54*, 102566. [CrossRef]
29. Alvarez, G.; Li, S. Some basic cryptographic requirements for chaos-based cryptosystems. *Int. J. Bifurc. Chaos* **2006**, *16*, 2129–2151. [CrossRef]
30. Elghandour, A.N.; Salah, A.M.; Elmasry, Y.A.; Karawia, A.A. An Image Encryption Algorithm Based on Bisection Method and One-Dimensional Piecewise Chaotic Map. *IEEE Access* **2021**, *9*, 43411–43421. [CrossRef]
31. Lu, Q.; Yu, L.; Zhu, C. Symmetric Image Encryption Algorithm Based on a New Product Trigonometric Chaotic Map. *Symmetry* **2022**, *14*, 373. [CrossRef]
32. Stoyanov, B.; Kordov, K. Image Encryption Using Chebyshev Map and Rotation Equation. *Entropy* **2015**, *17*, 2117–2139. [CrossRef]
33. Es-Sabry, M.; El Akkad, N.; Merras, M.; Saaidi, A.; Satori, K. A new image encryption algorithm using random numbers generation of two matrices and bit-shift operators. *Soft Comput.* **2019**, *24*, 3829–3848. [CrossRef]