Optik - International Journal for Light and Electron Optics 273 (2023) 170342

Contents lists available at ScienceDirect

Optik

journal homepage: www.elsevier.com/locate/ijleo

# Image encryption algorithm based on a new chaotic system with Rubik's cube transform and Brownian motion model

Yibo Zhao [a,b,*], Ruoyu Meng [a,b], Yi Zhang [a,b], Qing Yang [a,b]

[a] School of Electronic & Information Engineering, Nanjing University of Information Science & Technology, Nanjing 210044, China
[b] Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET), Nanjing University of Information Science & Technology, Nanjing 210044, China

ARTICLE INFO

*Keywords:*
Image encryption
Chaotic system
Rubik's cube algorithm
Brownian motion
Security analysis

ABSTRACT

In this paper a new two-dimensional chaotic map based on Logistic map and Chebyshev map is proposed. The performance analysis and evaluation show the proposed map has a very wide range of chaos and good ergodicity. Further, a new image encryption algorithm is designed using the chaotic map with Rubik's Cube transform and Brownian motion model. The encryption process is 'confusion - Rubik's cube transformation - diffusion'. In the confusion stage, a method combining Brownian motion theory and chaotic sequence is proposed. Then combining the advantages of chaotic system and Rubik's cube, a Rubik's Cube transformation algorithm is proposed. The scrambled pixel matrix and the chaotic matrix are spliced into a hexahedron structure, and the Rubik's Cube is dynamically scrambled through a pseudo-random sequence. Finally, the XOR operation is performed on the basis of the two-dimensional Hénon chaotic map. The experimental results show that the algorithm has the advantages of large key space, strong robustness and high security, and can resist common cryptanalysis attacks.

## 1. Introduction

In the post-epidemic era, digital data carrying various kinds of information increasingly interact through the Internet. The network transmission of data brings superior convenience to people but also encounters the risks of data leakage, loss and malicious attacks. In these data transmissions, the digital image is a typical two-dimensional (2D) data with high redundancy and large data volume, which makes the conventional algorithms for text data encryption no longer applicable [1–3]. Therefore, many cryptosystems have been developed to find practical image encryption algorithms. In general, these cryptosystems can be divided into the symmetric structure [4–8] and asymmetric structure [9–12], for instance, DNA encoding [4], chaos theory [5–7], wavelet transform [8] and compressive sensing [9–12], to be named a few. Among them, chaos theory's unpredictability and initial value sensitivity can be well applied to cryptography [13], which shows great potential in the field of image encryption [14].

With the development of chaos theory, chaotic systems have gradually become an essential tool in communication, watermarking and multimedia encryption, etc. The existing chaotic systems can be divided into one-dimensional (1D) chaotic systems [15,16] and high-dimensional (HD) chaotic systems [17–19], among which discrete chaotic systems are also called chaotic maps. Some classic 1D chaotic maps, such as Chebyshev map, Logistic map, Sine map, etc., have relatively simple structures with a single system parameter

* Corresponding author at: School of Electronic & Information Engineering, Nanjing University of Information Science & Technology, Nanjing 210044, China.
*E-mail address:* yibozhaodn@163.com (Y. Zhao).

https://doi.org/10.1016/j.ijleo.2022.170342
Received 10 October 2022; Received in revised form 28 November 2022; Accepted 2 December 2022
Available online 5 December 2022

and variable, which lead to the narrow chaotic range of the parameter. What is worse is that the initial value and system parameter may be predicted by some attack algorithms [20]. These flaws pose security risks to encryption systems. Contrary to 1D systems, HD chaotic systems have more variables and system parameters, which make the system structure relatively complex, and the generated sequence is more random. Thus, HD chaotic systems are more suitable for encryption. However, as the dimension of the system increases, their structure also become more complex, which leads to an increment in the implementation cost and computational complexity of hardware or software. Therefore, to achieve the trade-off between complexity and security, a new two-dimensional (2D) chaotic system is proposed, which strives to have complex chaotic properties while maintaining a simple structure.

In the cryptosystem, there are two basic methods to guide cryptographic design: confusion and diffusion [21]. The confusion-diffusion structure of chaotic image encryption was proposed first in 1998 [22]. Under the guidance of predecessor theories, many algorithms [23–27] based on chaotic image encryption have introduced confusion and diffusion operations, where the pseudo-random sequence generated by the chaotic system is used to diffuse and scramble the original image to eliminate the statistical characteristics and correlation of the encrypted image. Ref. [23] proposed a secure multiple-image encryption method based on 3D bit-scrambling and diffusion operations, in which a layer scrambling is added. An encryption algorithm based on the permutation–diffusion structure is proposed, then the classical permutation is adopted [24]. In Ref. [25], a cross-channel pixel and bit scrambling algorithm is proposed. The first round is pixel-level scrambling, and the second round is bit-level scrambling. Ref. [26] proposed a linear-nonlinear-linear encryption structure to verify the effectiveness of encryption based on the newly designed chaotic system. In Ref. [27], a novel chaos-based image encryption algorithm for color images based on 3D bit-plane permutation is presented. In the scrambling process of the above methods, most of them directly use the pseudo-random sequence generated by the chaotic system for sorting so that the obtained position sequence is used for pixel scrambling, which leads to insufficient flexibility and regular pattern. So, it is necessary to break the convention and build a flexible and powerful random scrambling model on a mature theoretical method. The Brownian motion is an excellent candidate.

Based on the above analysis, a new 2D Logistic-Chebyshev chaotic map (2D-LCCM) is proposed first, which makes up for the shortcomings of some existing chaotic maps, such as narrow parameter range, weak ergodicity, uneven sequence distribution, etc. Then, a new image encryption algorithm is designed, and the Rubik's cube transformation process is added based on confusion-diffusion, aiming at dispersing plaintext information via three-dimension space. During the scrambling phase, a Brownian motion model is constructed to thoroughly shuffle the pixel positions of the image. The pixel trajectory is dynamically controlled by three parameters, which shows more random and unpredictable. After scrambling, the scrambled image matrix and the chaotic matrices are spliced into a hexahedron for Rubik's cube transformation to disperse the original image information. Finally, the image data after Rubik's cube transformation is decomposed into corresponding 8-bit binary. The upper 4-bit and lower 4-bit data are taken as two diffusion variables, and the XOR operation is performed on the basis of the Hénon map. When decrypting, an encrypted image and five cipher matrices are required for complete decryption, which significantly increases the difficulty of deciphering.

The rest of this paper is organized as follows: In Section 2, a new 2D system is proposed after Logistic map and Chebyshev map are briefly reviewed. The detailed encryption process based on the proposed chaotic system is shown in Section 3. Section 4 illustrates the decryption process. Section 5 presents the simulation results and carries out the safety analysis. Conclusion is given in Section 6.

## 2. The 2D Logistic-Chebyshev chaotic map

### 2.1. The definition of 2D-LCCM

Logistic map [28] is a classical 1D chaotic map with complex dynamic behavior, and its mathematical iterative equation is

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

where $\mu \in [0, 4]$ is the control parameter. When $\mu \in (3.569945, 4]$, the system is in chaos.$x_0$ is the initial value and the value range of the chaotic sequence $x_{n+1}$ is [0,1].

Chebyshev map [29] is also a 1D map with only one control parameter, whose mathematical iterative equation is

$$x_{n+1} = \cos(\beta \arccos x_n) \tag{2}$$

where $\beta \in [0, 4]$ is the control parameter. When $\beta > 1$, the system is in chaos. $x_0$ is the initial value and the value range of the chaotic sequence $x_{n+1}$ is $[-1,1]$.

To improve the complexity of the chaotic map and expand the chaotic range of the system, the output of the Logistic map can be considered as the control parameter of the Chebyshev map. Since the chaotic behavior of Chebyshev map appears when the control parameter $\beta$ is greater than 1, and the output of Logistic map is always not less than 0, an exponential function can be introduced to make $\beta$ always greater than or equal to 1. Then, Eqs. (1), (2) and the exponential function are combined to obtain the following equation, which is defined by

$$\left\{ \begin{array}{l} x_{n+1} = \cos(\beta(y_n) \arccos x_n) \\ y_{n+1} = \cos(\beta(x_n) \arccos y_n) \end{array} \right\} \tag{3}$$

where the control parameters $\beta(y_n) = e^{\mu y_n(1-y_n)}$ and $\beta(x_n) = e^{\mu x_n(1-x_n)}$ are dynamic. The outputs $x_{n+1}$ and $y_{n+1}$ are in the range $[-1,1]$. In order to remove negative values, a gain is given to $x_{n+1}$ and $y_{n+1}$, and a floor operator is added to the Eq. (3) to get the final chaotic

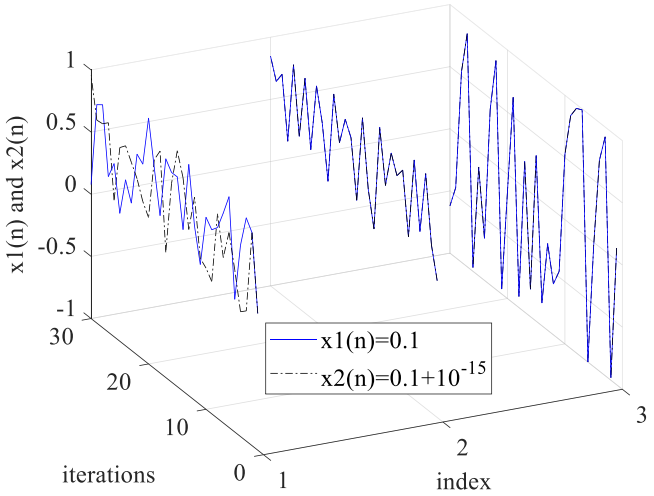


**Fig. 1.** Trajectory curve under different initial values for different chaotic maps (The index 1-the proposed map, The index 2-Logistic map, The index 3-Chebyshev map).

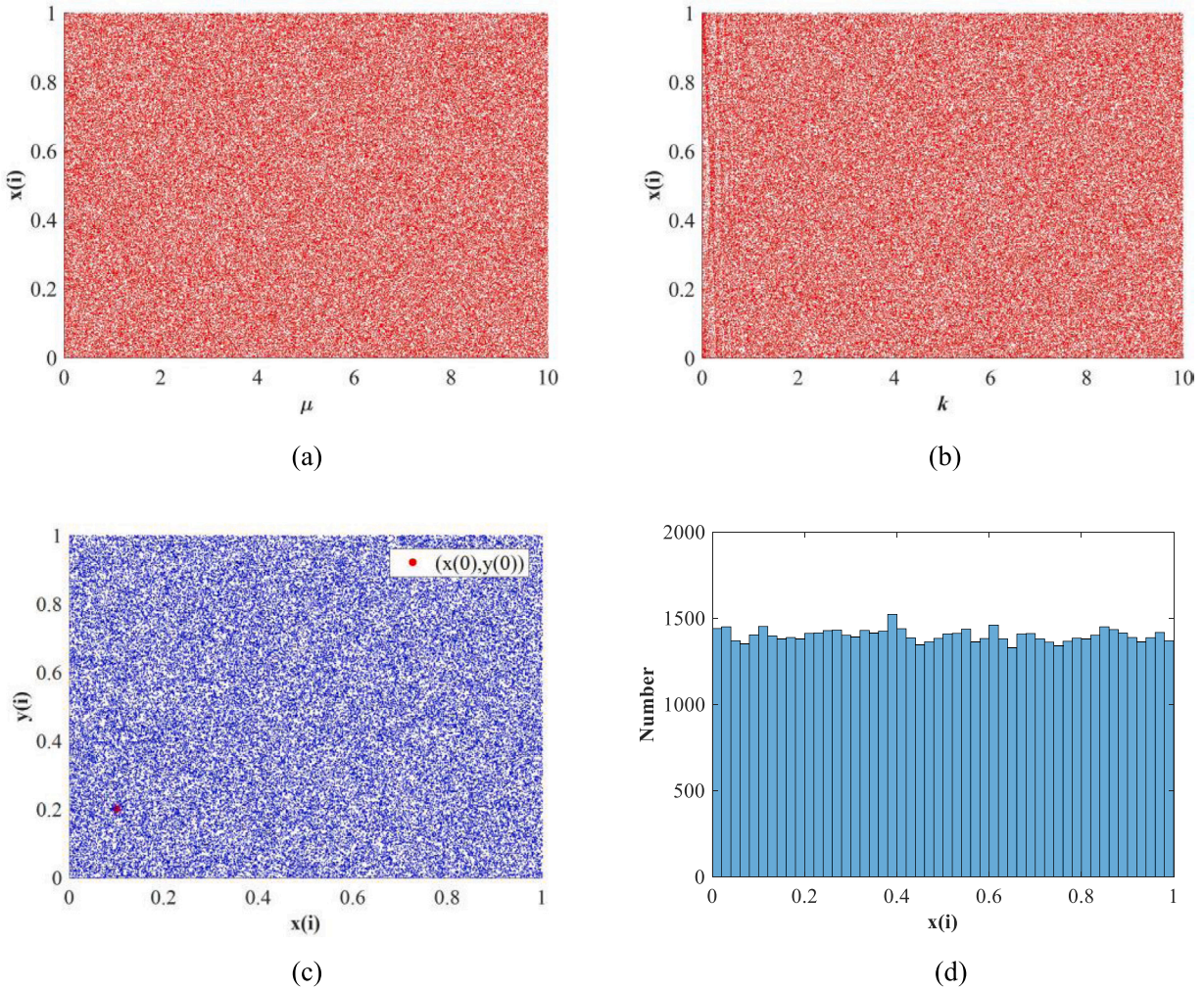


**Fig. 2.** Bifurcation diagram and attractor trajectory of the 2D-LCCM: (a) bifurcation diagram for $x$ against $\mu$; (b) bifurcation diagram for $x$ against $k$; (c) attractor; (d) histogram of (c).

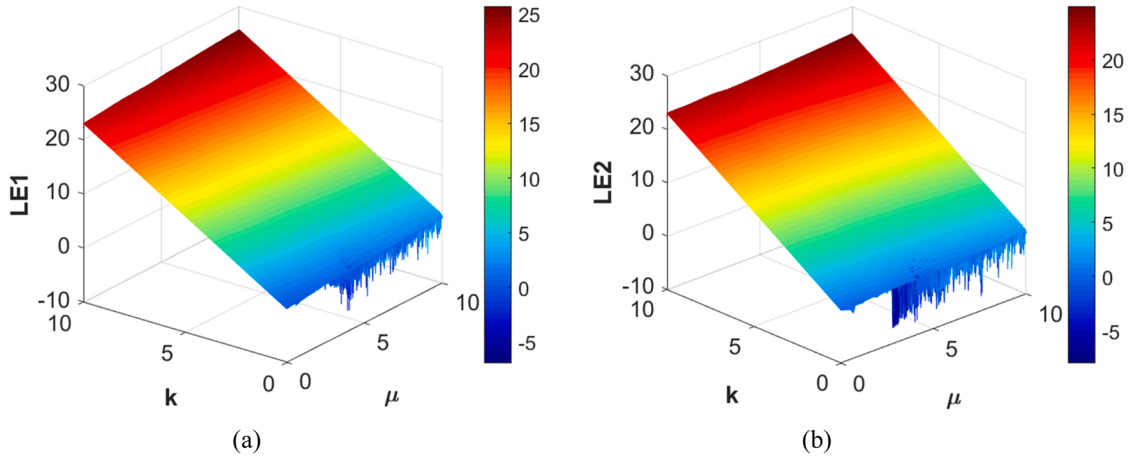


**Fig. 3.** 3D LEs diagram of the 2D-LCCM:(a) LE1; (b) LE2.

system as

$$\left\{\begin{array}{l} x_{n+1} = \cos(\beta(y_n)\arccos x_n) \times 10^k - \\ \quad \text{floor}\left(\cos(\beta(y_n)\arccos x_n) \times 10^k\right) \\ \\ \&y_{n+1} = \cos(\beta(x_n)\arccos y_n) \times 10^k - \\ \quad \text{floor}\left(\cos(\beta(x_n)\arccos y_n) \times 10^k\right) \end{array}\right\} \tag{4}$$

where $\mu$ and $k$ stand for control parameters, whose values range are both $(0, +\infty]$, floor represents the round-down function. The proposed map in this paper is affected by two chaotic systems, it is more complex than 1D system.

### 2.2. Performance analyses and evaluation

#### 2.2.1. Sensitivity analysis

A well-performing chaotic system can be very sensitive to its initial value, and a huge difference can be shown in a very small number of iterations. The sensitivity of the above three mentioned chaotic maps with the same initial value are simulated, as shown in Fig. 1. Blue solid and black dashed lines stand for the trajectories with the initial values $x1(0) = 0.1$and $x2(0) = 0.1 + 10^{-15}$, respectively. In Fig. 1, different indexes represent different chaotic maps, 1 represents the proposed chaotic map, 2 and 3 represent the Logistic map and the Chebyshev map respectively. Blue solid and black dashed lines overlap completely for the Logistic and Chebyshev maps, which means that the two maps have weak sensitivity within the first 30 iterations. For the proposed chaotic map, blue solid and black dashed lines start to separate at the third iteration, which means that the 2D-LCCM is extremely sensitive to small changes in the initial values. The 2D-LCCM is also extremely sensitive to changes in control parameters, which is not described here.

#### 2.2.2. Bifurcation diagram and attractor trajectory

Bifurcation diagram points out the elements of chaotic sequence in terms of control parameter [30]. To simplify, when the system parameters change to a certain value, the system suddenly changes from regularity to irregularity and enters a chaotic state. It is desired that the numbers of fixed points should be diverse rather than fixed in a straight line. Bifurcation diagrams with respect to parameters $\mu$ and $k$ are given in Fig. 2(a) and (b), which shows the characteristics of diversification because of no periodic window. The attractor trajectory and distribution histogram of the 2D-LCCM are shown in Fig. 2(c) and (d), in which the red point represents the coordinate value of the initial iteration. The initial values are set as (0.1, 0.2), and the system control parameters are chosen as $\mu = 1.2$, $k = 8$. After $7 \times 10^4$iterations, the output of the proposed map is randomly distributed over the entire 2D plane of $x,y \in [0,1]$, which shows a state of uniform distribution.

#### 2.2.3. Lyapunov exponent

The Lyapunov exponent (LE) is often used to evaluate the chaotic properties of dynamical systems, and it specifies the separation rate of the nonlinear dynamic system's neighboring orbits. A system with a positive LE is considered chaotic, and a system with two or more positive LEs is considered hyper-chaotic. For the 2D chaotic system, the Jacobian method is used to calculate LEs. The equations of a general 2D chaotic map can be described as

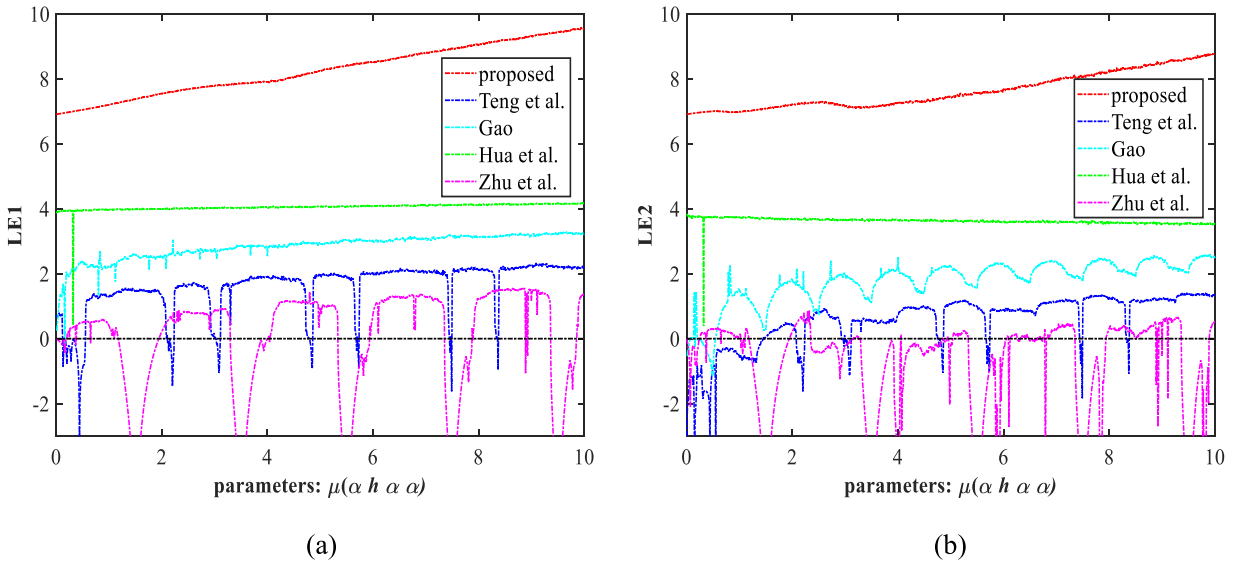


**Fig. 4.** Comparison of LEs between different 2D chaotic systems: (a) comparison of larger LE values of various 2D maps; (b) comparison of smaller LE values of various 2D maps.

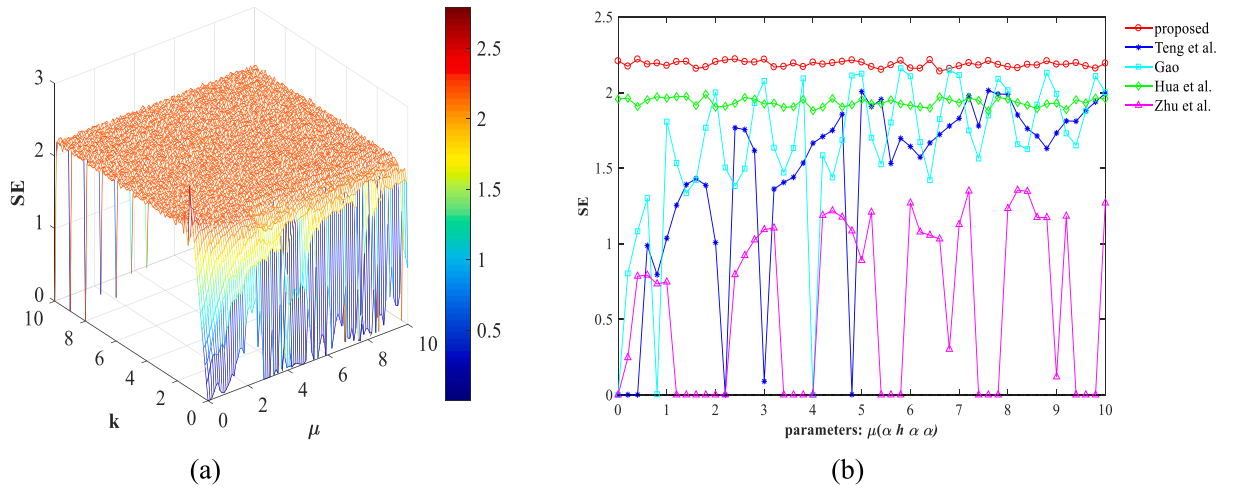


**Fig. 5.** 3D SE illustration of the 2D-LCCM and comparison between different chaotic systems:(a) 3D SE illustration of the 2D-LCCM; (b) Comparison of SE among different chaotic systems.

$$f\left(x,y\right)=\left\{\begin{array}{l}x_{n+1}=f_1(x_n,y_n)\\ \&y_{n+1}=f_2(x_n,y_n)\end{array}\right\} \tag{5}$$

Then, the Jacobian matrix of Eq. (5) is computed by

$$J=\begin{pmatrix}\dfrac{\partial f_1}{\partial x_n} & \dfrac{\partial f_1}{\partial y_n}\\ \dfrac{\partial f_2}{\partial x_n} & \dfrac{\partial f_2}{\partial y_n}\end{pmatrix} \tag{6}$$

After $n$ iterations, $n$ matrices are generated. Let $\lambda_{1j}$, $\lambda_{2j}$ be the eigenvalues of these $n$ matrices, LEs can be quantified by

**Table 1**
NIST test results.

| Test index | P-value | Result |
|---|---|---|
| Frequency | 0.369186 | Pass |
| Block-Frequency | 0.826378 | Pass |
| Cumulative Sums | 0.558760 | Pass |
| Runs | 0.704544 | Pass |
| Longest Runs of Ones | 0.284931 | Pass |
| Rank | 0.529902 | Pass |
| FFT | 0.659591 | Pass |
| Non Overlapping Template(m = 9, template = 000000001) | 0.024041 | Pass |
| Overlapping Template(m = 9) | 0.381364 | Pass |
| Universal | 0.123324 | Pass |
| Approximate Entropy | 0.121465 | Pass |
| Random Excursions(x = −1) | 0.714693 | Pass |
| Random Excursions Variant(x = 1) | 0.855791 | Pass |
| Linear Complexity | 0.177409 | Pass |
| Serial | 0.111984 | Pass |

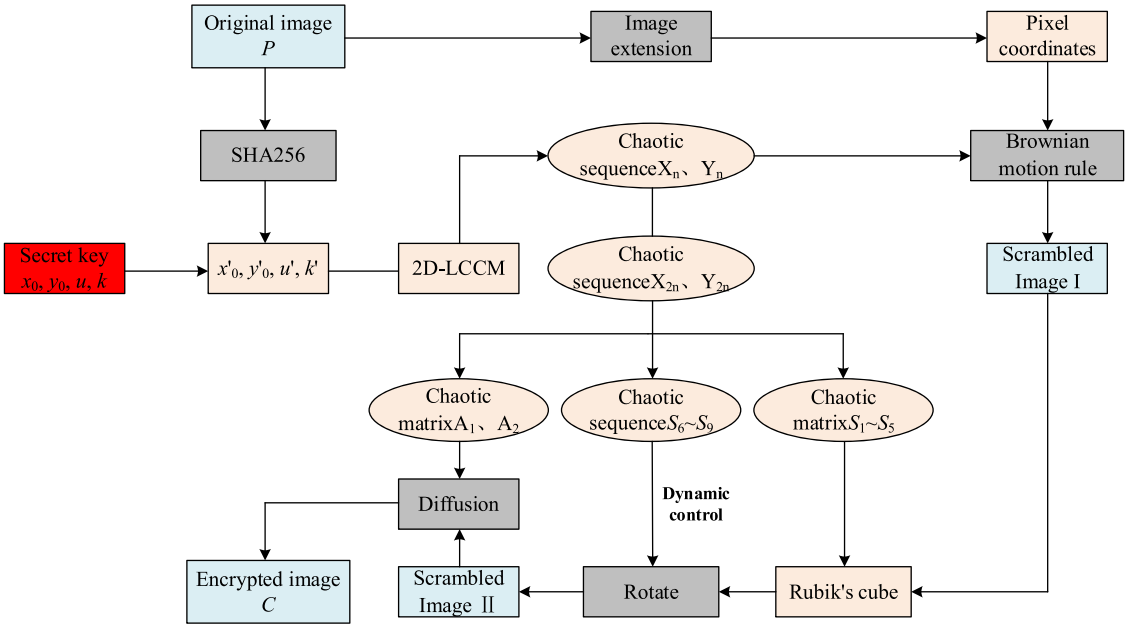


**Fig. 6.** The flow chart of algorithm.

$$\begin{Bmatrix} LE_1 = \lim_{n\to\infty} \frac{1}{n} \sum_{j=1}^{n} \ln\left|\lambda_{1j}\right| \\ LE_2 = \lim_{n\to\infty} \frac{1}{n} \sum_{j=1}^{n} \ln\left|\lambda_{2j}\right| \end{Bmatrix} \tag{7}$$

The two LEs of the 2D-LCCM with respect to $\mu$ and $k$ are shown in Fig. 3. As can be seen from the figure, when $k$ is fixed, LE1 and LE2 are both relatively stable with the increase of $\mu$. Whereas, when $\mu$ is fixed, LE1 and LE2 both increase significantly with the increase of $k$. In most regions, both LE1 and LE2 are greater than 0, which means that the proposed chaotic map has hyperchaotic behavior. The LE values are compared to the last existing chaotic systems in Fig. 4, including the chaotic system proposed by Teng et al. [31], Gao [32], Hua et al. [33] and Zhu et al. [34]. In the comparison, $k$= 3 and $\mu$ is set as the control parameter, and the parameters of other chaotic maps are the default parameters in their papers. The LE simulation results show that both the larger LE values and the smaller LE values of the proposed 2D-LCCM are larger than other chaotic maps. More importantly, large LE values can be obtained as far as possible by adjusting $\mu$ and $k$.

### 2.2.4. Sample entropy

Sample entropy (SE) is a parameter for assessing the complexity of time-series signals [35], which quantitatively defines the

complexity of dynamic systems. The time series {$x(1)$, $x(2)$,., $x(N)$} with length $N$ is expanded to a set of vector sequences {$\boldsymbol{X}_{\mathrm{m}}(1)$, $\boldsymbol{X}_{\mathrm{m}}(2)$,., $\boldsymbol{X}_{\mathrm{m}}(\mathrm{N}\text{-}m+1)$} with dimension $m$, where $\boldsymbol{X}_{\mathrm{m}}(i)$= {$x(i)$, $x(i+1)$,., $x(i+\mathrm{m}\text{-}1)$}, $1 \le i \le N\text{-}m+1$. These vectors represent values from the $i$-th point that starts $m$ consecutive values. Then SE can be computed as

$$SE\left(m, r, N\right) = -\log\frac{A}{B} \tag{8}$$

where $A$ and $B$ are the numbers of vectors. Denote d[$\boldsymbol{X}_{\mathrm{m}}(i)$, $\boldsymbol{X}_{\mathrm{m}}(j)$] as the Chebyshev distance between $\boldsymbol{X}_{\mathrm{m}}(i)$ and $\boldsymbol{X}_{\mathrm{m}}(j)$, and have d[$\boldsymbol{X}_{\mathrm{m}+1}(i)$, $\boldsymbol{X}_{\mathrm{m}+1}(j)$] < $r$ and d[$\boldsymbol{X}_{\mathrm{m}}(i)$, $\boldsymbol{X}_{\mathrm{m}}(j)$] < $r$, $m$ and $r$ usually are set as 2 and 0.2std (*time series*), respectively. The 3D SE variation of the 2D-LCCM with respect to $\mu$ and $k$ is depicted in Fig. 5(a). When $k\in(0.9, 10]$ and $\mu\in(0, 10]$, the SE plane of the 2D-LCCM is greater than 2, and the fluctuation is stable. Fig. 5(b) plots the SE comparison between the 2D-LCCM and other different 2D chaotic systems. In the comparison, the parameters of all 2D chaotic systems are set to the same values as in the LE experiments. It can be seen that the SE value of the 2D-LCCM is larger than that of other 2D chaotic systems, which means that the proposed system produces a higher complexity of time series.

#### *2.2.5. NIST randomness test*

To test the randomness of the sequence generated by the 2D-LCCM, the sequence is examined using the NIST SP 800–22 test suite, which has 15 items. The significance level of each test item is set as $\alpha = 0.01$. If the P-value of each subtest is greater than $\alpha$, the test passes, the sequence is considered almost random, and the confidence level is 99%. The test results are shown in Table 1. As can be seen, all P-values are greater than $\alpha$, which demonstrates that the generated chaotic sequence exhibit good randomness.

## 3. Encryption algorithm

In this section, a new image encryption algorithm based on the 2D-LCCM is proposed. The encryption algorithm flow is shown in Fig. 6. Assume that the original image $P$ to be encrypted is a grayscale image of size $M\times N$. First, $P$ is input to the SHA256 hash function to generate a hash value associated with the plaintext, then the hash value and secret key are used to generate the system parameters and initial values of the 2D-LCCM map. Second, in order to increase the applicability of the algorithm, the original image is extended. The expanded image is scrambled through the Brownian motion model, then the scrambled image matrix and the chaotic matrix are spliced into a virtual hexahedron for Rubik's cube transformation. Finally, the data is diffused by XOR operation. Through the strategy of confusion - Rubik's cube transformation - diffusion, an ordinary image can be encrypted into an unrecognizable password image.

### *3.1. Generation of system parameters and initial values*

To make the key extremely sensitive to changes in the plaintext, the image $P$ is input into the SHA-256 hash function, which outputs a hexadecimal string of length 64. Divide the string into two groups, and record the first 32 as $key_1$ and the last 32 as $key_2$. The $key_1$ and $key_2$ are XORed to obtain $key_3$. The $key_3$ is divided into 32 groups, denoted as $H$= [$H_1$, $H_2$,., $H_{32}$]. Each group $H_i$ ($i$ = 1, 2,., 32) consists of 4 bits and corresponds to the number between 0 and 15 in decimal. The four variables associated with the plaintext are obtained by

$$\left\{\begin{array}{l} \&\varepsilon_1 = \dfrac{H_1 \oplus H_2 \oplus H_3 \oplus H_4 \oplus H_5 \oplus H_6 \oplus H_7 \oplus H_8}{15} \\ \&\varepsilon_2 = \dfrac{H_9 \oplus H_{10} \oplus H_{11} \oplus H_{12} \oplus H_{13} \oplus H_{14} \oplus H_{15} \oplus H_{16}}{15} \\ \&\varepsilon_3 = \dfrac{H_{17} \oplus H_{18} \oplus H_{19} \oplus H_{20} \oplus H_{21} \oplus H_{22} \oplus H_{23} \oplus H_{24}}{15} \\ \&\varepsilon_4 = \dfrac{H_{25} \oplus H_{26} \oplus H_{27} \oplus H_{28} \oplus H_{29} \oplus H_{30} \oplus H_{31} \oplus H_{32}}{15} \end{array}\right\} \tag{9}$$

where $\oplus$ represents the XOR operation. To show how a slight change affects the entire chaotic system, an intermediate variable is introduced, which is defined as

$$\eta = \frac{x_0}{y_0} \times \frac{u}{k} \tag{10}$$

where $x_0, y_0, u, k$ are the initial keys of the cryptosystem. Then, the initial values ($x'_0, y'_0$) and system parameters ($u'$, $k'$) of the 2D-LCCM are calculated as

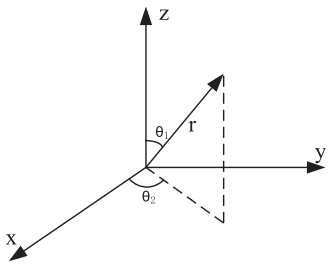


**Fig. 7.** Brownian motion model.

| | $x_2(1)$ | $x_2(2)$ | $x_2(3)$ | $x_2(4)$ | $x_2(5)$ | $x_2(6)$ |
|---|---|---|---|---|---|---|
| L | 2. 2 | 1. 9 | 1. 2 | 1. 8 | 2. 5 | 2. 7 |
| | $I(1)$ | $I(2)$ | $I(3)$ | $I(4)$ | $I(5)$ | $I(6)$ |
| L' | 4 | 3 | 1 | 2 | 5 | 6 |

**Fig. 8.** Generation of permutation position sequence.

$$\begin{cases} \&x_0' = \mathrm{mod}\left(\dfrac{\varepsilon_1+\eta}{x_0+\eta}, 1\right) \\ \&y_0' = \mathrm{mod}\left(\dfrac{\varepsilon_2+\eta}{y_0+\eta}, 1\right) \\ \&u' = u + \dfrac{\varepsilon_3}{\eta} \\ \&k' = k + \dfrac{\varepsilon_4}{\eta} \end{cases} \tag{11}$$

It can be seen from Eq. (11) that a slight change in the plaintext or key can be reflected in the initial values ($x_0'$, $y_0'$) and system parameters ($u'$, $k'$).

### 3.2. Pixel Brownian motion scrambling

To increase the applicability of the algorithm, $P$ is filled with zero pixel value and the size of $P$ is expanded to $K\times K$ ($K$=max{$M$, $N$}). The expanded image is denoted as $Q_1$. The specific steps are as follows:

- *Step* 1, Read all pixel coordinates of the image $Q_1$ in the row direction, where the coordinates of the $i$-th pixel is ($x_1(i)$, $y_1(i)$) ($i$ = 1, 2,., $K^2$). The $x_1(i)$ is the number of rows where the $i$-th pixel is located, and the $y_1(i)$ is the number of columns where the $i$-th pixel is located.
- *Step* 2, Using the initial values and control parameters generated in Section **3.1**, the 2D-LCCM is iterated for $K\times K\times$ $n$ + 1000 times. In order to obtain better randomness, the first 1000 points are discarded, and two subsequences $X_n$ and $Y_n$ are obtained for image scrambling, where $n$ is set as 20.
- *Step* 3, Transform the coordinates obtained in *Step* 1 into ($x_1(i)$, $y_1(i)$, 0) that is taken as the initial point of the $i$-th pixel. Brownian motion is simulated for all pixels of image $Q_1$, and each pixel moves 20 times. Fig. 7 shows a schematic diagram of Brownian motion.

The motion of the pixel is controlled by $\theta_1$, $\theta_2$ and $r$, where $0 \le r \le 100$, $0 \le \theta_1 \le \pi$, $0 \le \theta_2 \le 2\pi$. $\theta_1$, $\theta_2$ and $r$ can be obtained by

$$\begin{cases} \&\theta_1 = \mathrm{mod}(\mathrm{floor}(X_n \times 10^8), 181) \\ \&\theta_2 = \mathrm{mod}(\mathrm{floor}(Y_n \times 10^8), 361) \\ \&r = \mathrm{mod}(\mathrm{floor}((X_n + Y_n) \times 10^8), 101) \end{cases} \tag{12}$$

- *Step* 4, After Brownian motion, the three-dimensional coordinates are mapped to the XOY plane, and then they are normalized to the first quadrant to get a set of coordinates: {($x_2(1)$, $y_2(1)$), ($x_2(2)$, $y_2(2)$),., ($x_2(K^2)$, $y_2(K^2)$)}. The row coordinate array $L = [x_2(1), x_2(2), \ldots, x_2(K^2)]$ is ranked from small to large, and the ranking array $L' = [I(1), I(2), \ldots, I(K^2)]$ is gotten. The process is shown in Fig. 8.

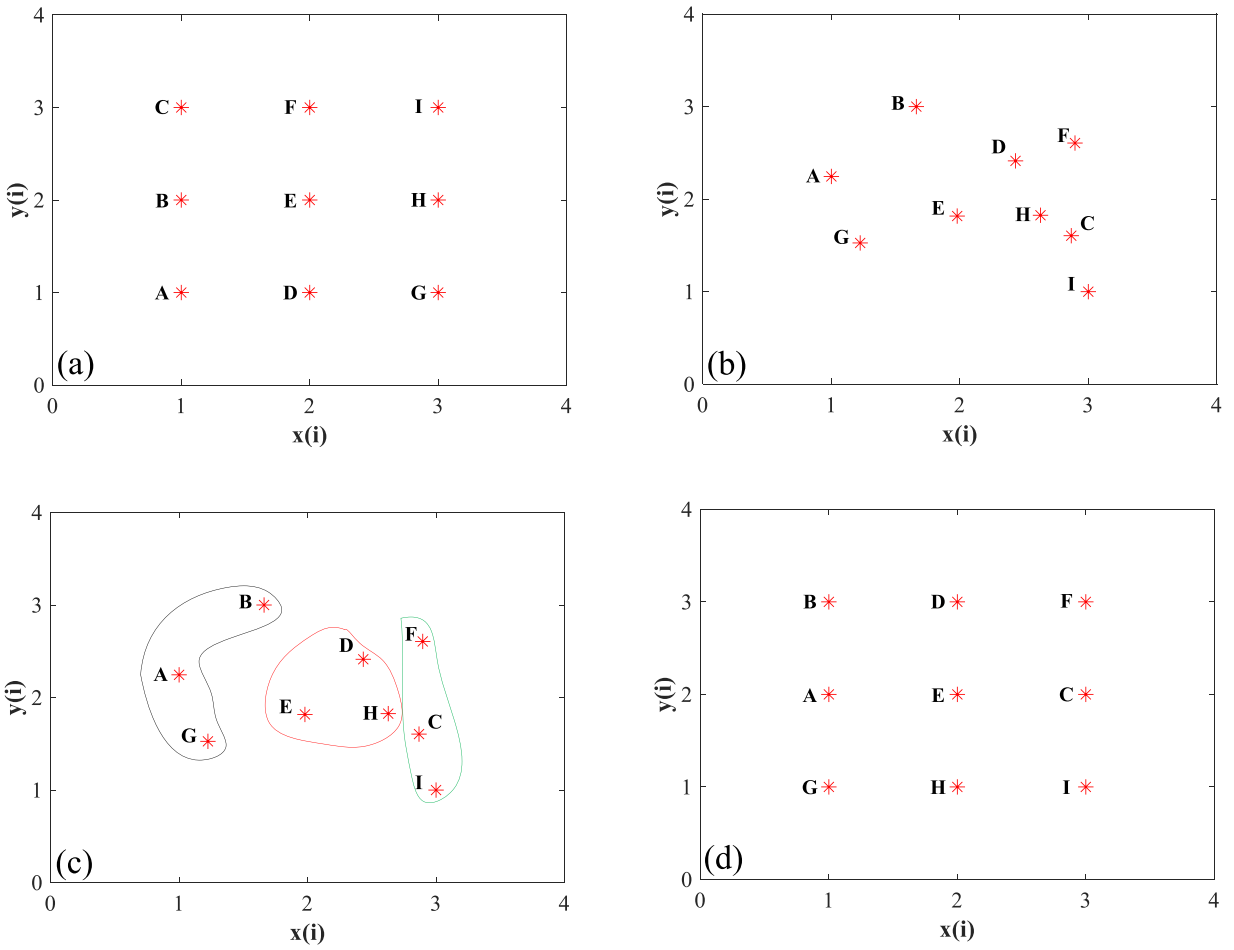


**Fig. 9.** 3 × 3 scrambling example: (a) coordinates before scrambling; (b) coordinates after Brownian motion; (c) area division; (d) coordinates after scrambling.

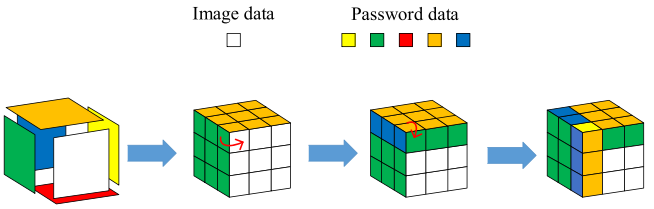


**Fig. 10.** Rubik's Cube transformation process.

- *Step* 5, The scrambled row coordinates $x_3(i)$ are obtained by Eq. (13). The principle of row coordinate rearrangement is to assign values according to the distance between points and the $y$-axis. The row coordinates of the $K$ points closest to the $y$-axis are 1, the row coordinates of the $K$ points second closest to the $y$-axis are 2, and so on. Similarly, the column coordinates corresponding to the $K$ points in the same row are ranked to obtain the scrambled column coordinates $y_3(i)$. An example of 3 × 3 scrambling is shown in Fig. 9.

$$x_3(i) = \begin{Bmatrix} \&1I(i) < K \\ \&\dfrac{I(i)}{K}\bmod(I(i),K) = 0 \\ \&\mathrm{floor}\left(\dfrac{I(i)}{K}\right) + 1I(i) > K, \bmod(I(i),K) \neq 0 \end{Bmatrix} \tag{13}$$

- *Step* 6, $Q_2(x_3(i)\ ,\ y_3(i)) = Q_1(x_1(i)\ ,\ y_1(i))$. $Q_2$ is the scrambled image, $Q_2$ and $Q_1$ have the same size.

### 3.3. Rubik's cube transformation

In order to increase the difficulty of being cracked by the attacker and cover up the information of the original image, a $K$-order Rubik's cube transformation is further introduced to achieve the purpose of dispersing and hiding the information of the original image. The process is shown in Fig. 10. Through this transformation, the image data and the password data are confused with each other, increasing the algorithm complexity.

Specific steps are as follows:

- *Step* 1, Take the first $K^2$ terms of $X_n$ and $Y_n$ to get the subsequences $X_{2n}$ and $Y_{2n}$. Five different chaotic matrices are generated by the five different combinations in Eq. (14), which are denoted as $S_1$, $S_2$, $S_3$, $S_4$ and $S_5$ respectively. $S_1 \sim S_5$ are used for the splicing of the Rubik's cube. Different combination can avoid multiple iterations of chaotic system.

$$\left\{\begin{array}{c} \&S_1 = \mathrm{mod}(\mathrm{floor}\left(\left({X_{2n}}^2 - {Y_{2n}}^2 + 1\right) \times 10^8\right), 256) S_1 = \mathrm{reshape}\left(S_1, K, K\right) \\ S_2 = \mathrm{mod}(\mathrm{floor}\left(\left({X_{2n}}^2 - {Y_{2n}}^2 + 1\right) \times 10^8\right), 256) S_2 = \mathrm{reshape}\left(S_2, K, K\right) \\ \&S_3 = \mathrm{mod}(\mathrm{floor}\left(\left({X_{2n}}^2 - Y_{2n} + 1\right) \times 10^8\right), 256) S_3 = \mathrm{reshape}\left(S_3, K, K\right) \\ S_4 = \mathrm{mod}(\mathrm{floor}\left(\left(X_{2n} - Y_{2n} + 1\right) \times 10^8\right), 256) S_4 = \mathrm{reshape}\left(S_4, K, K\right) \\ \&S_5 = \mathrm{mod}(\mathrm{floor}\left(\left({X_{2n}}^2 - {Y_{2n}}^3 + 1\right) \times 10^8\right), 256) S_5 = \mathrm{reshape}\left(S_5, K, K\right) \end{array}\right\} \tag{14}$$

  where reshape is a function that converts a 1D sequence to a 2D matrix.
- *Step* 2, Since the output of the 2D-LCCM is within [0,1], $X_{2n}$ and $Y_{2n}$ are converted into 8-precision binary sequences, and the length of sequences are $1 \times 8K^2$. Take the first 1000 items of the two sequences to get the pseudo-random sequence, denoted as $S_6$ and $S_7$ which only contains 0 and 1. Then the sequences $S_8$ and $S_9$ are generated by

$$\left\{\begin{array}{c} \&S_8 = \mathrm{mod}(\mathrm{floor}\left(X_{2n}\left(1:1000\right) \times 10^8\right), K) + 1 \\ S_9 = \mathrm{mod}(\mathrm{floor}\left(Y_{2n}\left(1:1000\right) \times 10^8\right), 4) + 1 \end{array}\right\} \tag{15}$$

  The rotation of the Rubik's cube is dynamically controlled by the pseudo-random sequence $S_6$,$S_7$,$S_8$,$S_9$, and their lengths are all $1 \times 1000$. $S_6$ is taken as 0 to represent row rotation, and taken as 1 to represent column rotation. $S_7$ is taken as 0 to indicate that the rotation direction is left/up, and taken as 1 to indicate that the rotation direction is right/down. The value range of $S_8$ is from 1 to $K$. $S_8 = j_1$ represents the $j_1$-th row or $j_1$-th column of the rotating cube. The value range of $S_9$ is from 1 to 4. $S_9 = j_2$ represents the rotation angle is $j_2 \times 90°$. For example, $(S_6(i), S_7(i), S_8(i), S_9(i)) = (0, 0, 10, 3)$ means that the operation of the $i$-th rotation is to rotate the 10th row to the left by 270°.
- *Step* 3, The $Q_2$ image matrix and the chaotic matrix $S_1, S_2, S_3, S_4, S_5$ are spliced into a virtual Rubik's cube, and the cube is scrambled. After 1000 rotations, $Q_2$ becomes $Q_3$, and the chaotic matrices $S_1$, $S_2$, $S_3$, $S_4$, $S_5$ become $S_1^{'}$, $S_2^{'}$, $S_3^{'}$, $S_4^{'}$, $S_5^{'}$.

### 3.4. Diffusion process

In order to ensure the security of encryption, it is necessary to perform a diffusion operation after Rubik's cube transformation. Through the operation, subtle changes in the original image can affect the entire encrypted image, resulting in an avalanche effect. Here, a Hénon map [36] is introduced into the diffusion algorithm, and the current pixel value is influenced by the upper 4-bit and lower 4-bit data of the previous pixel value. The method is more complex and safer than the ordinary XOR operation. The specific diffusion steps are as follows:

- *Step* 1, The 4-bit binary number corresponds to the decimal number in the range of 0–15, so the two chaotic matrices $A_1$ and $A_2$ used in the diffusion process can be quantized by

$$\left\{\begin{array}{c} \&A_1 = \mathrm{mod}(\mathrm{floor}\left(\left(X_{2n} + Y_{2n} + 1\right) \times 10^8\right), 16) A_1 = \mathrm{reshape}\left(A_1, K, K\right) \\ \&A_2 = \mathrm{mod}(\mathrm{floor}\left(\left({X_{2n}}^2 + {Y_{2n}}^2 + 1\right) \times 10^8\right), 16) A_2 = \mathrm{reshape}\left(A_2, K, K\right) \end{array}\right\} \tag{16}$$

- *Step* 2, The gray value of $Q_3$ is decomposed into 8-bit binary, and the upper 4-bit data and lower 4-bit data of each gray value are extracted. The value at the coordinate $(a_1, a_2)$ is denoted as $Q_3(a_1, a_2)$, then $Q_3(a_1, a_2)$ can be expressed as $Q_3(a_1, a_2) = P_8P_7P_6P_5P_4P_3P_2P_1$. $Q_{3\mathrm{H}}(a_1, a_2) = P_8P_7P_6P_5$ and $Q_{3\mathrm{L}}(a_1, a_2) = P_4P_3P_2P_1$ are used to represent the upper 4-bit and the lower 4-bit data at the coordinate $(a_1, a_2)$, respectively.
- *Step* 3, XOR operation on the high and low 4-bit data obtained in *step* 2 is performed. Eqs. (17, 18) are specific operation rules. Diffusion of lower 4-bit data is defined as

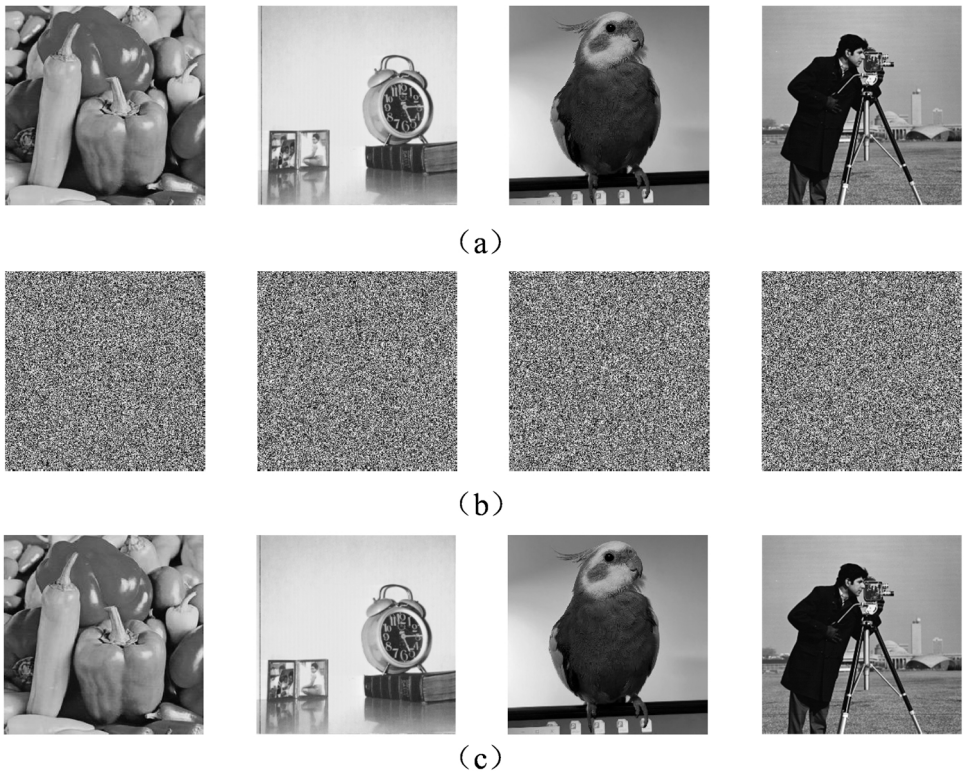


**Fig. 11.** Encryption and decryption results for different images: (a) original images; (b) encrypted images; (c) decrypted images.

$$Q'_{3L}(i,j)=\begin{cases} f\left(i,j\right)\oplus \mathrm{mod}\left(\mathrm{floor}\left(\left(1-1.4\times\left(\frac{k_0}{15}\right)^2+\frac{k_1}{15}\right)\times 10^8\right),16\right), \mathrm{for} i=1, j=1; \\ f\left(i,j\right)\oplus \mathrm{mod}\left(\mathrm{floor}\left(\left(1-1.4\times\left(\frac{Q'_{3L}(i,j-1)}{15}\right)^2+\frac{Q'_{3H}(i,j-1)}{15}\right)\times 10^8\right),16\right), \mathrm{for} j\neq 1; \\ f\left(i,j\right)\oplus \mathrm{mod}\left(\mathrm{floor}\left(\left(1-1.4\times\left(\frac{Q'_{3L}(i-1,K)}{15}\right)^2+\frac{Q'_{3H}(i-1,K)}{15}\right)\times 10^8\right),16\right), \mathrm{for} i\neq 1, j=1. \end{cases} \tag{17}$$

where $Q'_{3L}(i,j)$ is the diffusion result of the lower 4-bit data. $Q'_{3L}(i,j)$ is not only related to the lower 4-bit data of the previous pixel value, but also depends on the upper 4-bit data of the previous pixel value. Diffusion of upper 4-bit data is defined as

$$Q'_{3H}(i,j)=\begin{cases} g(i,j)\oplus \mathrm{mod}\left(\mathrm{floor}\left(0.3\times\frac{k_0}{15}\times 10^8\right),16\right), \mathrm{for} i=1, j=1; \\ g(i,j)\oplus \mathrm{mod}\left(\mathrm{floor}\left(0.3\times\frac{Q'_{3L}(i,j-1)}{15}\times 10^8\right),16\right), \mathrm{for} j\neq 1; \\ g\left(i,j\right)\oplus \mathrm{mod}\left(\mathrm{floor}\left(0.3\times\frac{Q'_{3L}(i-1,K)}{15}\times 10^8\right),16\right), \mathrm{for} i\neq 1, j=1. \end{cases} \tag{18}$$

where $Q'_{3H}(i,j)$is the diffusion result of the upper 4-bit data. The result of $Q'_{3H}(i,j)$ depends on the lower 4-bit of the previous pixel value. The definitions of $f(i,j)$ and $g(i,j)$ used in the above diffusion equation are given by

$$\left\{\begin{array}{l} f(i,j)=Q_{3L}(i,j)\oplus \mathrm{A}_1(i,j)\oplus \mathrm{A}_2(K+1-i,K+1-j) \\ g(i,j)=Q_{3H}(i,j)\oplus \mathrm{A}_1(K+1-i,K+1-j)\oplus \mathrm{A}_2(i,j) \end{array}\right\} \tag{19}$$

In the above Eqs. (17) to (19), $0\le i\le K$, $0\le j\le K$, $k_0$ and $k_1$ are the given initial values. Nonlinear operation is used in the diffusion, which is equivalent to 2D Hénon map with control parameters (*a, b*) of (1.4, 0.3). The method is more complicated than the ordinary XOR operation and cannot be easily cracked.

- *Step* 4, $Q_4$=$Q'_{3H}(i,j)$+$Q'_{3L}(i,j)$, where $Q_4$ is the final encrypted image, and '+ ' indicates the combined process.

**Table 2**
Comparison of key spaces.

| | Algorithm | | | | |
|---|---|---|---|---|---|
| | Proposed | Ref.[38] | Ref.[39] | Ref.[40] | Ref.[41] |
| Key space | $2^{456}$ | $2^{250}$ | $2^{187}$ | $2^{390}$ | $2^{250}$ |

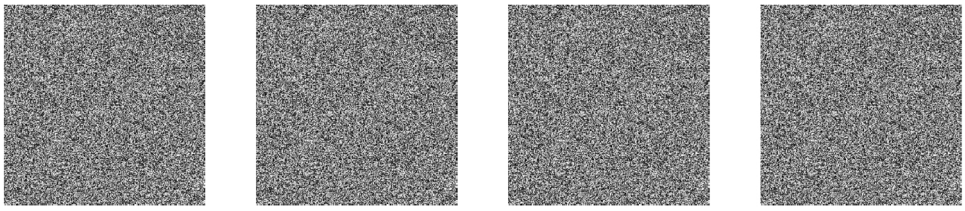


**Fig. 12.** Decryption results with different keys: (a) $key_1$; (b) $key_2$; (c) $key_3$; (d) $key_4$.

## 4. Decryption Process

The proposed encryption algorithm is a asymmetric encryption system. The decryption process is the reverse process of the encryption process, which is briefly described as:

- *Step* 1, the encrypted image $Q_4$, chaos matrix $S_1^{'}, S_2^{'}, S_3^{'}, S_4^{'}, S_5^{'}$, initial key and hash value are input.
- *Step* 2, the reverse of the diffusion algorithm is done to obtain $Q_3$.
- *Step* 3, $Q_3$and the chaotic matrix $S_1^{'}, S_2^{'}, S_3^{'}, S_4^{'}, S_5^{'}$are re-spliced into a Rubik's cube, and rotated in reverse to get $Q_2$.
- *Step* 4, the reverse of the scrambling algorithm is performed on $Q_2$ to get $Q_1$.
- *Step* 5, the pixel values added in the $Q_1$ matrix is deleted to get the original image $P$.

## 5. Experimental results and safety analysis

The experimental hardware environment is 8 GB memory and Windows 10 operating system. The software simulation platform is Matlab 2017a. Set $n = 20$ in the scrambling algorithm, $k_0 = 4$ and $k_1 = 6$ in the diffusion algorithm. The values of the initial security key are set as $x_0 = 0.1$, $y_0 = 0.2$, $u$= 10, $k = 10$. To verify the effectiveness of the proposed encryption scheme, standard grayscale images of size 256 × 256 are selected for encryption and decryption, including Peppers, Clock, Parrot, and Cameraman. The experimental results are shown in Fig. 11. As can be seen from the figures, the encrypted images become noise-like images, and the effective information of the original images cannot be obtained, which makes it difficult to crack. Meanwhile, the decrypted images obtained by the proposed algorithm are lossless.

### 5.1. Key space and sensitivity analysis

The encryption algorithm should contain a large enough key space and be sensitive to changes in the security key. When designing a cryptographic system, the minimum size of the key space must be no less than $2^{100}$ so as to resist brute force cracking [37]. In this paper, the key space of the proposed cryptosystem is divided into two parts. The first part is the initial given values, including $x_0$, $y_0$, $u$, $k$. Assuming the computational precision of the computer is $10^{-15}$, the key space for this part is $10^{15\times4} = 10^{3\times20} \approx 2^{10\times20}$. The second part is the hash value generated by SHA-256, and the key space of this part is $2^{256}$. Therefore, the total key space is about $2^{456}$, which is greater than $2^{100}$. The comparison result of key space between the proposed encryption algorithm and others is shown in Table 2.

Next, to evaluate the sensitivity of the key when decrypting, the slight changes are sequentially made to the four parameters of the key. Five different keys: $key_0$, $key_1$, $key_2$, $key_3$ and $key_4$ are given below.

- $key_0$: $x_0 = 0.1$, $y_0 = 0.2$, $u$= 10, $k = 10$.
- $key_1$: $x_0 = 0.1 + 10^{-15}$, $y_0 = 0.2$, $u$= 10, $k = 10$.
- $key_2$: $x_0 = 0.1$, $y_0 = 0.2 + 10^{-15}$, $u$= 10, $k = 10$.
- $key_3$: $x_0 = 0.1$, $y_0 = 0.2$, $u$= $10 + 10^{-15}$, $k = 10$.
- $key_4$: $x_0 = 0.1$, $y_0 = 0.2$, $u$= 10, $k = 10 + 10^{-15}$.

Where $key_0$ is the correct key, $key_1$, $key_2$, $key_3$, and $key_4$ are the wrong key, and the difference between the correct and wrong keys is only $10^{-15}$. The Clock is encrypted with $key_0$ and decrypted with $key_1$, $key_2$, $key_3$, and $key_4$, respectively. The decrypted results are

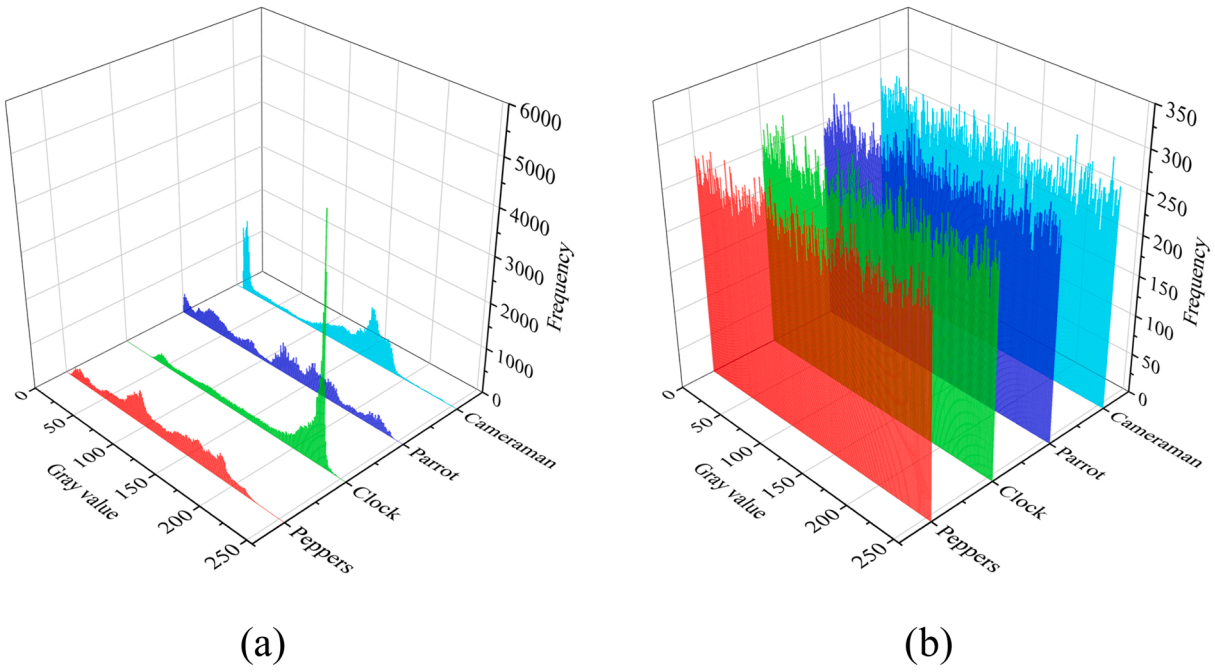


**Fig. 13.** Histograms of original and encrypted images: (a) histogram set of the original images; (b) histogram set of the encrypted images.

**Table 3**
Comparison of $\chi^2$ values for Peppers image.

| Algorithm | Image | $\chi^2$ value | Results |
|---|---|---|---|
| Proposed | **Peppers** | **251.3728** | Pass |
| | Clock | 261.2763 | Pass |
| | Parrot | 226.6904 | Pass |
| | Cameraman | 275.9324 | Pass |
| Ref.[43] | Peppers | 254.5391 | Pass |
| Ref.[44] | Peppers | 254.7510 | Pass |
| Ref.[45] | Peppers | 256.7900 | Pass |
| Ref.[46] | Peppers | 272.6719 | Pass |

shown in Fig. 12. As can be seen from the figures, the decryption can't be obtained correctly by the different wrong keys, which indicates that the proposed algorithm is very sensitive to changes in the key and has high security.

### 5.2. Statistical characteristic analysis

#### 5.2.1. Histogram analysis

To resist statistical attacks, the histogram of the encrypted image must be uniform and completely different from the histogram of the original image. The histograms of the images before and after encryption are shown in Fig. 13. It can be observed that the histograms of encrypted images are uniformly distributed in [0255], which is completely different from the original images. It is difficult for an attacker to predict the original image by employing a statistical analysis method. Therefore, the proposed encryption algorithm can effectively conceal the statistical characteristics of the original image. Furthermore, the regularity of the histogram is quantified by the Chi-square test [42] and compared with other algorithms. The $\chi^2$ value is defined mathematically as

$$\chi^2 = \sum_{j=0}^{255} \frac{(f_j - g)^2}{g} \tag{20}$$

where $j$ is the pixel value, $f_j$ is the frequency with which pixel value $j$ occurs, and $g$ is the theoretically desired value. The gap between the actual observed value and the theoretically desired value is lower the smaller the value of $\chi^2$ is. $\chi^2_{0.05} = 293.2478$ at significance level $\alpha = 0.05$ is given, which means that when $\chi^2_{0.05} > 293.2478$, $\chi^2_{0.05}$ is in the reject domain and the test fails. The test results for different images are given in Table 3 and compared with other algorithms. From the table, it can be observed that all the tested images passed Chi-square test, which means that the histograms of the encrypted images are uniform. And the $\chi^2$ value of Peppers is smaller

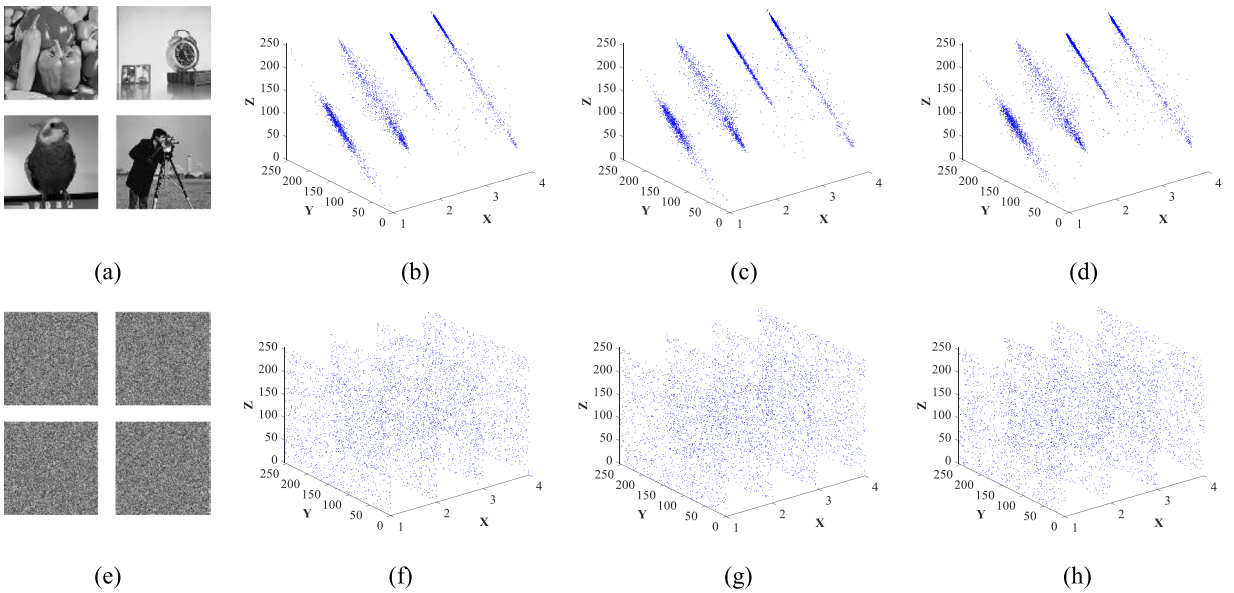


**Fig. 14.** Visualize correlation results:(a) original images; (b) horizontal adjacent pixel pairs of original images; (c) vertical adjacent pixel pairs of original images; (d) diagonal adjacent pixel pairs of original images; (e) encrypted images; (f) horizontal adjacent pixel pairs of encrypted images; (g) vertical adjacent pixel pairs of encrypted images; (h) diagonal adjacent pixel pairs of encrypted images.

**Table 4**
Correlation of adjacent pixels in different directions.

| File name | Size | Original image | | | Encrypted image | | |
|---|---|---|---|---|---|---|---|
| | | Horizontal | Vertical | Diagonal | Horizontal | Vertical | Diagonal |
| 5.1.09 | 256 × 256 | 0.90607 | 0.9437 | 0.90992 | -0.0075024 | -0.0026044 | 0.016095 |
| 5.1.10 | 256 × 256 | 0.91264 | 0.87292 | 0.83229 | -0.0010898 | -0.01921 | -0.0045269 |
| 5.1.11 | 256 × 256 | 0.96679 | 0.92082 | 0.88877 | -0.0010698 | -0.0036067 | -0.014803 |
| 5.1.12 | 256 × 256 | 0.96415 | 0.97777 | 0.95142 | 0.011945 | 0.00020047 | -0.0027988 |
| 5.1.13 | 256 × 256 | 0.8702 | 0.86613 | 0.74521 | -0.02551 | -0.0035202 | 0.0016429 |
| 5.1.14 | 256 × 256 | 0.94901 | 0.90548 | 0.8541 | 0.010285 | -0.0047354 | -0.0025123 |
| 5.2.08 | 512 × 512 | 0.93738 | 0.88665 | 0.87281 | -0.001266 | 0.014353 | -0.0007423 |
| 5.2.09 | 512 × 512 | 0.91979 | 0.83142 | 0.77967 | 0.02621 | 0.00013951 | 0.0036748 |
| 5.2.10 | 512 × 512 | 0.93238 | 0.92034 | 0.89469 | -0.024431 | -0.0008964 | 0.00048733 |
| 7.1.01 | 512 × 512 | 0.96146 | 0.91458 | 0.89391 | -0.0069743 | 0.0044193 | 0.0051545 |

than that of other algorithms.

### 5.2.2. *Adjacent pixel correlation analysis*

There is a high correlation between adjacent pixels in an original image and a pixel often leaks the information of its surrounding pixels. This feature can be often used by attackers to reason and predict the gray value of the next pixel, further to crack the entire original image. Therefore, a good encryption algorithm should have the ability to break the correlation between adjacent pixels. To test the correlation of adjacent pixels, 1000 pairs of adjacent pixels are randomly selected in the horizontal, vertical, and diagonal directions of the original images and encrypted images. The correlation coefficient is calculated by

$$\left\{\begin{array}{c} \&r_{xy} = \dfrac{\mathrm{cov}(x,y)}{\sqrt{D(x)}\sqrt{D(y)}} \\ \\ \&\mathrm{cov}(x,y) = \dfrac{1}{N}\sum_{i=0}^{N}(x_i - E(x))(y_i - E(y)) \\ \\ \&E(x) = \dfrac{1}{N}\sum_{i=0}^{N}x_i, D(x) = \dfrac{1}{N}\sum_{i=0}^{N}(x_i - E(x))^2 \end{array}\right\} \tag{21}$$

The correlations of adjacent pixels in the horizontal, vertical, and diagonal directions of four images (filenames from '1′ to '4′) are plotted in Fig. 14 before and after encryption. In each figure, the value of the X-axis indicates the filenames of the four images, and the values of adjacent pixels are plotted on the *Y-Z* phase plane. It can be seen from Fig. 14(b)~(d) that the adjacent pixel pairs of the four

**Table 5**
Comparison of correlation coefficients for Peppers image.

| Algorithm | Image | Direction | | | Average |
|---|---|---|---|---|---|
| | | Horizontal | Vertical | Diagonal | |
| Proposed | **Peppers** | 0.0058 | -0.0060 | -0.0055 | **0.0057** |
| | Clock | -0.0068 | -0.0005 | 0.0043 | 0.0039 |
| | Parrot | -0.0047 | -0.0091 | -0.0013 | 0.0050 |
| | Cameraman | 0.0003 | -0.0031 | 0.0112 | 0.0049 |
| Ref.[43] | Peppers | 0.0078 | 0.0058 | − 0.0164 | 0.0100 |
| Ref.[44] | Peppers | 0.0028 | 0.0096 | 0.0062 | 0.0062 |
| Ref.[45] | Peppers | 0.0070 | -8.4085e-04 | 1.8576e-04 | 0.0027 |
| Ref.[46] | Peppers | 0.0094 | -0.0046 | 0.0074 | 0.0071 |

**Table 6**
Information entropy of images.

| Images | 5.1.09 | 5.1.10 | 5.1.11 | 5.1.12 | 5.1.13 | 5.1.14 | 5.2.08 | 5.2.09 | 5.2.10 | 7.1.01 |
|---|---|---|---|---|---|---|---|---|---|---|
| Original image | 6.7093 | 7.3118 | 6.4523 | 6.7057 | 1.5483 | 7.3424 | 7.201 | 6.994 | 5.7056 | 6.0274 |
| Encrypted image | 7.9972 | 7.9970 | 7.9977 | 7.9991 | 7.9991 | 7.9976 | 7.9993 | 7.9992 | 7.9993 | 7.9993 |

**Table 7**
Comparison of information entropy for Peppers image.

| Algorithm | Image | Plain | Encrypted |
|---|---|---|---|
| Proposed | **Peppers** | 7.5819 | **7.9992** |
| | Clock | 6.7057 | 7.9991 |
| | Parrot | 7.7477 | 7.9976 |
| | Cameraman | 7.0097 | 7.9986 |
| Ref.[43] | Peppers | | 7.9974 |
| Ref.[44] | Peppers | | 7.9974 |
| Ref.[45] | Peppers | | 7.9971 |
| Ref.[46] | Peppers | | 7.9970 |

original images are mostly on the diagonal or close to the diagonal, indicating that these adjacent pixels exhibit a strong correlation. However, Fig. 14(f)~(h) show that the adjacent pixel pairs of all encrypted images are very randomly distributed on the *Y-Z* phase plane, which means that the proposed algorithm can efficiently decorrelate the high correlations for the original images. Table 4 presents the quantitative results of the correlation of ten classical grayscale images selected from the USC_SIPS image dataset. The results show that the correlation coefficients of adjacent pixels of encrypted images are close to 0. When giving the point graphs of correlation and the correlation coefficient table, the results obtained by different algorithms are also compared and shown in Table 5. In comparison, the correlation coefficients obtained by the proposed algorithm are much closer to 0 than that of other algorithms, except for [45].

### *5.2.3. Information entropy analysis*

Information entropy can be used to measure the randomness of an image. If an image is more random, its information entropy is closer to 8. The grayscale image is divided into 256 grayscale levels. $P(x_i)$ is the probability of occurrence of gray level $i$ in the encrypted image $x$. The calculation equation of information entropy is

$$H(x) = -\sum_{i=1}^{256} P(x_i)\log_2 P(x_i) \tag{22}$$

Table 6 shows the calculation results of the information entropy of different images before and after encryption. It can be seen from the table that the information entropy of all encrypted images is very close to 8, which is a significant improvement over the original image information entropy. Besides, the comparison of information entropy values with various recent studies is presented in Table 7. The obtained information entropy is slightly closer to 8 than other algorithms, which shows the effectiveness of the proposed encryption algorithm.

## *5.3. Analysis of resistance to differential attacks*

Differential attack is the study of how differences in inputs affect the corresponding outputs. Subtle difference is made by the attacker to the original image, and the proposed encryption algorithm is used to encrypt the images before and after the change. By comparing two encrypted images, the relationship is found between the original image and the encrypted image, then to be cracked. Therefore, the ability to resist differential attack is related to the sensitivity to the original image. Number of pixels change rate (NPCR)

**Table 8**
NPCR and UACI scores of 8-bit grayscale images for different image encryption schemes.

| File name | Size | LAS-IES[10,48] | | LSMCL-IEA[34] | | Proposed | |
|---|---|---|---|---|---|---|---|
| | | NPCR (%) | UACI (%) | NPCR (%) | UACI (%) | NPCR (%) | UACI (%) |
| 5.1.09 | 256 × 256 | 99.6064 | 33.4456 | 99.6094 | 33.5253 | 99.6273 | 33.5725 |
| 5.1.10 | 256 × 256 | 99.6154 | 33.4946 | 99.6155 | 33.5115 | 99.6105 | 33.4133 |
| 5.1.11 | 256 × 256 | 99.6244 | 33.5541 | 99.6094 | 33.5174 | 99.6127 | 33.5579 |
| 5.1.12 | 256 × 256 | 99.5703 | 33.4302 | 99.5758 | 33.4202 | 99.6144 | 33.4214 |
| 5.1.13 | 256 × 256 | 99.6109 | 33.4438 | 99.6170 | 33.5019 | 99.6135 | 33.4023 |
| 5.1.14 | 256 × 256 | 99.6364 | 33.4655 | 99.6353 | 33.4939 | 99.6089 | 33.4242 |
| 5.2.08 | 512 × 512 | 99.5870 | 33.4008 | 99.6151 | 33.4766 | 99.6064 | 33.4961 |
| 5.2.09 | 512 × 512 | 99.6260 | 33.4804 | 99.6094 | 33.4528 | 99.6166 | 33.5413 |
| 5.2.10 | 512 × 512 | 99.6124 | 33.4563 | 99.6166 | 33.3925 | 99.6094 | 33.4477 |
| 7.1.01 | 512 × 512 | 99.5992 | 33.5037 | 99.5872 | 33.5017 | 99.6111 | 33.5027 |
| 7.1.02 | 512 × 512 | 99.6075 | 33.4237 | 99.6109 | 33.4415 | 99.6040 | 33.4952 |
| 7.1.03 | 512 × 512 | 99.6079 | 33.4291 | 99.6147 | 33.4455 | 99.6180 | 33.5064 |
| 7.1.04 | 512 × 512 | 99.5988 | 33.4739 | 99.6174 | 33.4772 | 99.6237 | 33.4317 |
| 7.1.05 | 512 × 512 | 99.6170 | 33.4362 | 99.6212 | 33.4615 | 99.6236 | 33.4458 |
| 7.1.06 | 512 × 512 | 99.6272 | 33.3954 | 99.6475 | 33.5036 | 99.6194 | 33.5017 |
| 7.1.07 | 512 × 512 | 99.5931 | 33.4073 | 99.6101 | 33.3952 | 99.5960 | 33.4418 |
| 7.1.08 | 512 × 512 | 99.6094 | 33.4332 | 99.6063 | 33.4682 | 99.5867 | 33.4133 |
| 7.1.09 | 512 × 512 | 99.6162 | 33.4177 | 99.6105 | 33.3940 | 99.6067 | 33.4218 |
| 7.1.10 | 512 × 512 | 99.6045 | 33.4344 | 99.5907 | 33.5565 | 99.6024 | 33.4333 |
| ruler.512 | 512 × 512 | 99.6120 | 33.4262 | – | – | 99.6106 | 33.5327 |
| gray21.512 | 512 × 512 | 99.6022 | 33.4608 | 99.5949 | 33.4314 | 99.6164 | 33.5694 |
| boat.512 | 512 × 512 | 99.6154 | 33.4654 | 99.5998 | 33.4519 | 99.6169 | 33.3942 |
| Mean | | 99.6091 | 33.4490 | 99.6102 | 33.4676 | **99.6166** | **33.4712** |

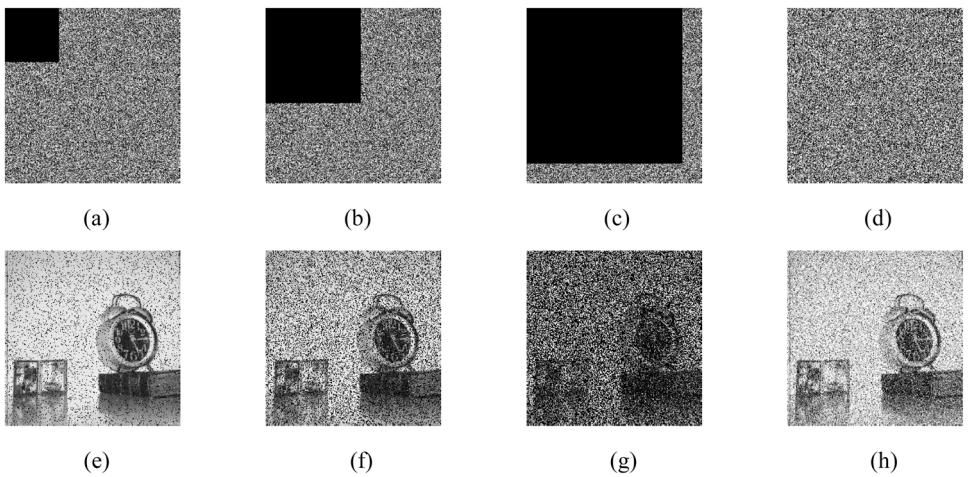


**Fig. 15.** Image decryption effect when subjected to clipping attack and noise pollution: (a)10% clipping;(b)30% clipping; (c)80% clipping; (d) Gaussian noise with mean 0 and variance 0.05; (e) Decrypted image with 10% clipping; (f) Decrypted image with 30% clipping; (g) Decrypted image with 80% clipping; (h) Decrypted image with Gaussian noise.

and unified average change intensity (UACI) are the most commonly used metrics to measure original image sensitivity. For two plain images $P_1$ and $P_2$ with only one pixel value difference, which are denoted as $C_1$ and $C_2$ after encryption, respectively. The NPCR value and UACI value between the two encrypted images can be calculated by

$$\begin{cases} NPCR\left(C_1, C_2\right) = \dfrac{\sum_{i,j} D(i,j)}{MN} \times 100\% \\ D\left(i,j\right) = \begin{cases} 1, C_1(i,j) \neq C_2(i,j) \\ 0, C_1(i,j) = C_2(i,j) \end{cases} \\ UACI = \dfrac{1}{MN} \dfrac{\sum_{i,j} |C_1(i,j) - C_2(i,j)|}{255} \times 100\% \end{cases} \tag{23}$$

**Table 9**
Time comparison of different algorithms.

| Algorithm | Image | Encryption time (s) | Hardware parameters |
|---|---|---|---|
| Proposed | Lena (256 ×256) | 3.0739 | i5/2.3 GHZ CPU, 8 G RAM |
| Ref.[45] | Lena (256 ×256) | 1.7351 | i5/2.3 GHZ CPU, 8 G RAM |
| Ref.[49] | Baboon (256 ×256) | 5.7855 | i7/3.4 GHZ CPU, 8 G RAM |

For 8-bit grayscale images, the ideal value for NPCR is 99.6094%, and the ideal value for UACI is 33.4635% [47]. The closer NPCR and UACI are to ideal values, the stronger the sensitivity of the encryption algorithm to original images and the more secure the encryption algorithm. 22 grayscale images are selected from the Miscellaneous image dataset in USC_SIPS. A pixel value somewhere in the original image is changed randomly, and the NPCR and UACI values of the encrypted image before and after the change are calculated in Table 8. Among all the three image encryption schemes, the means of NPCR and UACI obtained by the proposed scheme are greater than the ideal values of 99.6094% and 33.4635%, slightly better than the other two schemes. Therefore, it can conclude that the proposed algorithm has good resistance to differential attacks.

### *5.4. Robustness analysis*

Digital images are susceptible to noise and data loss during network transmission and storage on physical media. A good image encryption algorithm should have the ability to resist these anomalies. The robustness of the algorithm can be improved by introducing Rubik's cube transformation. In order to verify the decryption effect of the proposed encryption algorithm when disturbed, the encrypted image will be clipped in different degrees, and the Gaussian noise with mean 0 and variance 0.05 will be added for decryption. The corresponding experimental results are shown in Fig. 15. When the encrypted image is lost by 10% and 30%, the decrypted image still has good visual quality, and some valid information can still be recovered when noise is encountered. The decryption results under two abnormal states show that the proposed encryption algorithm has good anti-interference performance.

### *5.5. Speed performance*

The time of proposed encryption algorithm is mainly divided into three parts. The first part is the time consumed by Brownian motion, and the time cost of this part depends on the number of motions. The second part is the time consumed by rotating the cube, and the time cost depends on the number of rotations of the cube. The third part is the time consumed by diffusion. Taking the Lena image (256 ×256) as an example, the number of movements in the Brownian motion model is set to 20, and the scrambling time is slightly longer, about 1.32 s. In the Rubik's cube transformation, the number of cube rotations is set to 1000 times, and the time is about 0.075 s. In the last stage, the encryption time is about 1.6789 s. The total time of the proposed encryption algorithm is about 3.0739 s. The comparison of the proposed algorithm and other algorithms in terms of time is shown in Table 9. The comparison results show that the proposed algorithm has not yet reached an excellent level, but it is generally acceptable. In addition, there are many factors that affect the running time, such as emulation software, CPU frequency, RAM size, and so on. It is not appropriate to measure the quality of the algorithm simply by time. Here only a reference result is given.

## 6. Conclusion

In this paper, a new 2D-LCCM map is proposed to make a trade-off between computational cost and cryptographic security. The design inspiration of this system is that the Logistic map and the exponential function are combined to construct a dynamic parameter to control Chebyshev map. The dimension is extended from 1D to 2D to enhance the complexity. Experiment results such as attractor trajectories, bifurcation, LE, SE, and NIST tests prove that 2D-LCCM has excellent ergodicity, wide hyperchaotic range, and strong randomness, and performs well when compared with other 2D chaotic systems. Besides, an image encryption algorithm combining Brownian motion and Rubik's cube is designed based on 2D-LCCM. The security analysis results show that the encrypted image can resist statistical analysis very well, which are illustrated by the chi-square test, correlation and information entropy results. The NPCR and UACI are all close to their ideal values. What is more, when the encrypted image is subject to 80% clipping attack and noise attack, the receiver can still obtain valid visual information. Therefore, the proposed image encryption algorithm can resist various attacks, including differential attack, statistical attack, clipping attack, and noise attack.

## CRediT authorship contribution statement

**Yibo Zhao:** Methodology, Writing – original draft. **RunYu Meng:** Methodology, Writing – original draft, Software. **Yi Zhang:** Software, Simulation. **Qing Yang:** Simulation.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

This work was in part supported by a research grant provided by a Project Funded III by the Priority Academic Program Development of Jiangsu Higher Education Institutions, National Natural Science Foundation of China (61871230).

## References

[1] C. Wang, et al., Image description with polar harmonic Fourier moments, IEEE Trans. Circuits Syst., Video Technol. 30 (12) (2020) 4440–4452.
[2] M.A. Chenaghlu, M. Ali Balafar, M.R.F. Derakhshi, A novel image encryption algorithm based on polynomial combination of chaotic maps and dynamic function generation, Signal Process. 157 (2019) 1–13.
[3] K.U. Shahna, A. Mohamed, Novel hyper chaotic color image encryption based on pixel and bit level scrambling with diffusion, Signal Process.: Image Commun. 99 (2021), 116495.
[4] C. Zou, et al., A novel image encryption algorithm based on DNA strand exchange and diffusion, Appl. Math. Comput. 430 (2022), 127291.
[5] L. Liu, J. Wang, A cluster of 1D quadratic chaotic map and its applications in image encryption, Math. Comput. Simul. 204 (2022) 89–114.
[6] J. Wang, et al., Image encryption based on Logistic-Sine self-embedding chaotic sequence, Optik 271 (2022), 170075.
[7] K. Jain, A. Aji, P. Krishnan, Medical image encryption scheme using multiple chaotic maps, Pattern Recognit. Lett. 152 (2021) 356–364.
[8] M. Zarebnia, R. Parvaz, Dynamical 2D and 3D image encryption method by hybrid system based on cat map and wavelet transform, Optik 219 (2020), 165148.
[9] P. Rakheja, P. Singh, R. Vig, An asymmetric image encryption mechanism using QR decomposition in hybrid multi-resolution wavelet domain, Opt. Lasers Eng. 134 (2020), 106177.
[10] X. Huang, et al., Meaningful image encryption algorithm based on compressive sensing and integer wavelet transform, Front. Comput. Sci. 17 (3) (2023), 173804.
[11] E. Kumari, et al., Asymmetric color image encryption and compression based on discrete cosine transform in Fresnel domain, Results Opt. 1 (2020), 100005.
[12] G. Ye, M. Liu, M. Wu, Double image encryption algorithm based on compressive sensing and elliptic curve, Alex. Eng. J. 61 (2022) 6785–6795.
[13] S. Wang, L. Hong, J. Jiang, An image encryption scheme using a chaotic neural network and a network with multistable hyperchaos, Optik 268 (2022), 169758.
[14] Y. Peng, K. Sun, S. He, Dynamics analysis of chaotic maps: From perspective on parameter estimation by meta-heuristic algorithm, Chin. Phys. B 29 (3) (2020), 030502.
[15] M.A. Midoun, X. Wang, M.Z. Talhaoui, A sensitive dynamic mutual encryption system based on a new 1D chaotic map, Opt. Lasers Eng. 139 (2021), 106485.
[16] X. Wang, C. Liu, D. Jiang, An efficient double-image encryption and hiding algorithm using a newly designed chaotic system and parallel compressive sensing, Inf. Sci. 610 (2022) 300–325.
[17] J. Dai, et al., Quantum multi-image compression-encryption scheme based on quantum discrete cosine transform and 4D hyper-chaotic Henon map, Quantum Inf. Process. 20 (7) (2021) 246.
[18] L. Gong, et al., A NEw 4D chaotic system with coexisting hidden chaotic attractors, Int. J. Bifurc. Chaos 30 (10) (2020) 2050142.
[19] L. Gong, et al., New 4D chaotic system with hidden attractors and self-excited attractors and its application in image encryption based on RNG, Phys. A 591 (2022), 126793.
[20] C. Li, D. Arroyo, K.T. Lo, Breaking a chaotic cryptographic scheme based on composition maps, Int. J. Bifurc. Chaos 20 (8) (2010) 2561–2568.
[21] C.E. Shannon, Communication theory of secrecy systems, Bell Syst. Tech. J. 28 (4) (1949) 656–715.
[22] J. Fridrich, Symmetric ciphers based on two-dimensional chaotic maps, Int. J. Bifurc. Chaos 8 (06) (1998) 1259–1284.
[23] M. Demirtaş, A novel multiple grayscale image encryption method based on 3D bit-scrambling and diffusion, Optik 266 (2022), 169624.
[24] D. Herbadji, et al., Color Image Encryption Scheme Based on Enhanced Quadratic Chaotic Map, IET Image Process 14 (2020) 40–52.
[25] M. Demirtaş, A new RGB color image encryption scheme based on cross-channel pixel and bit scrambling using chaos, Optik 265 (2022), 169430.
[26] C. Pak, L. Huang, A new color image encryption using combination of the 1D chaotic map, Signal Process. 138 (2017) 129–137.
[27] Z. Gan, et al., A chaotic image encryption algorithm based on 3-D bit-plane permutation, Neural Comput. Appl. 31 (2019) 7111–7130.
[28] R.M. May, Simple mathematical models with very complicated dynamics, Nature 261 (5560) (1976) 459–467.
[29] L. Liu, et al., 2D Logistic-Adjusted-Chebyshev map for visual color image encryption, J. Inf. Secur. Appl. 60 (2021), 102854.
[30] U. Erkan, A. Toktas, Q. Lai, 2D hyperchaotic system based on Schaffer function for image encryption, Expert Syst. Appl. 213 (2023), 119076.
[31] L. Teng, et al., Color image encryption based on cross 2D hyperchaotic map using combined cycle shift scrambling and selecting diffusion, Nonlinear Dyn. 105 (2021) 1859–1876.
[32] X. Gao, Image encryption algorithm based on 2D hyperchaotic map, Opt. Laser Technol. 142 (2021), 107252.
[33] Z. Hua, et al., Color image encryption using orthogonal Latin squares and a new 2D chaotic system, Nonlinear Dyn. 104 (2021) 4505–4522.
[34] H. Zhu, et al., 2D logistic-modulated-sine-coupling-logistic chaotic map for image encryption, IEEE Access 7 (2019) 14081–14098.
[35] J. Richman, J.R. Moorman, Physiological time-series analysis using approximate entropy and sample entropy, AJP Heart Circ. Physiol. 278 (6) (2000) 2039–2049.
[36] J.A. Gallas, Structure of the parameter space of the Hénon map, Phys. Rev. Lett. 70 (18) (1993) 2714–2717.
[37] G. Alvarez, S. Li, Some basic cryptographic requirements for chaos-based cryptosystems, Int. J. Bifurc. Chaos 16 (8) (2006) 2129–2151.
[38] P. Ping, et al., Meaningful encryption: generating visually meaningful encrypted images by compressive sensing and reversible color transformation, IEEE Access 7 (2019) 170168–170184.
[39] X. Chai, et al., An efficient visually meaningful image compression and encryption scheme based on compressive sensing and dynamic LSB embedding, Opt. Lasers Eng. 124 (2020), 105837.
[40] X. Zhang, L. Zhang, Multiple-image encryption algorithm based on chaos and gene fusion, Multim. Tools Appl. 81 (2022) 20021–20042.
[41] A.A. Alarood, et al., IES: Hyper-chaotic plain image encryption scheme using improved shuffled confusion-diffusion, Ain Shams Eng. J. 13 (3) (2022), 101583.
[42] H. Ghazanfaripour, A. Broumandnia, Designing a digital image encryption scheme using chaotic maps with prime modular, Opt. Laser Technol. 131 (2020), 106339.
[43] S. Zhou, A real-time one-time pad DNA-chaos image encryption algorithm based on multiple keys, Opt. Laser Technol. 143 (2021), 107359.
[44] C. Li, X. Yang, An image encryption algorithm based on discrete fractional wavelet transform and quantum chaos, Optik 260 (2022), 169042.
[45] X. Yan, et al., Chaotic image encryption algorithm based on arithmetic sequence scrambling model and DNA encoding operation, Multimed. Tools Appl. 80 (2021) 10949–10983.
[46] X. Wang, R. Si, A new chaotic image encryption scheme based on dynamic L-shaped scrambling and combined map diffusion, Optik 245 (2021), 167658.
[47] C. Fu, et al., A chaos-based digital image encryption scheme with an improved diffusion strategy, Opt. Express 20 (3) (2012) 2363–2378.
[48] Z. Hua, Y. Zhou, Image encryption using 2D Logistic-adjusted-Sine map, Inf. Sci. 339 (2016) 237–253.
[49] Z. Huang, N. Zhou, Image encryption scheme based on discrete cosine Stockwell transform and DNA-level modulus diffusion, Opt. Laser Technol. 149 (2022), 107879.