

An image encryption algorithm based on spatiotemporal chaos and middle order traversal of a binary tree

Xingyuan Wang(王兴元), Yining Su(苏怡宁), and Shujuan Lin(林淑娟)
Citation: Chin. Phys. B, 2022, 31 (**11**): 110503. DOI: 10.1088/1674-1056/ac6861

What follows is a list of articles you may be interested in

Exponential sine chaotification model for enhancing chaos and its hardware implementation

Rui Wang(王蕊), Meng-Yang Li(李孟洋), and Hai-Jun Luo(罗海军)
Chin. Phys. B, 2022, 31 (**8**): 080508. DOI: 10.1088/1674-1056/ac6335

Control of chaos in Frenkel-Kontorova model using reinforcement learning

You-Ming Lei(雷佑铭) and Yan-Yan Han(韩彦彦)
Chin. Phys. B, 2021, 30 (**5**): 050503. DOI: 10.1088/1674-1056/abd74f

A novel pseudo-random coupled LP spatiotemporal chaos and its application in image encryption

Xingyuan Wang(王兴元), Yu Wang(王宇), Siwei Wang(王思伟), Yingqian Zhang(张盈谦), Xiangjun Wu(武相军)
Chin. Phys. B, 2018, 27 (**11**): 110502. DOI: 10.1088/1674-1056/27/11/110502

Effects of abnormal excitation on the dynamics of spiral waves

Min-Yi Deng(邓敏艺), Xue-Liang Zhang(张学良), Jing-Yu Dai(戴静娱)
Chin. Phys. B, 2016, 25 (**1**): 010504. DOI: 10.1088/1674-1056/25/1/010504

Secure communication based on spatiotemporal chaos

Ren Hai-Peng (任海鹏), Bai Chao (白超)
Chin. Phys. B, 2015, 24 (**8**): 080503. DOI: 10.1088/1674-1056/24/8/080503

An image encryption algorithm based on spatiotemporal chaos and middle order traversal of a binary tree

Yining Su(苏怡宁)[†], Xingyuan Wang(王兴元)[‡], and Shujuan Lin(林淑娟)

School of Information Science and Technology, Dalian Maritime University, Dalian 116026, *China*

(Received 12 December 2021; revised manuscript received 10 April 2022; accepted manuscript online 20 April 2022)

This paper proposes an image encryption algorithm based on spatiotemporal chaos and middle order traversal of a binary tree. Firstly, other programming software is used to perform the middle order traversal, and the plaintext image is sorted according to the middle order traversal sequence on the permutation. Secondly, the chaotic sequence is generated using the coupled map lattice to set the chaotic interference value. Finally, the XOR operation between the adjacent pixel values of the replacement image is completed to generate the ciphertext matrix. The simulation and experimental results show that the proposed algorithm can resist typical attacks and has good robustness.

Keywords: spatiotemporal chaos, image encryption, middle order traversal, coupled map lattice

PACS: 05.45.Gg, 07.05.Pj, 05.45.Ra **DOI:** 10.1088/1674-1056/ac6861

1. Introduction

With the rapid development of network and communication technology, information based on audio, video and image storage is distributed on various public platforms. Therefore, information security has become an important topic.[1–4] The key to researching information security is to preserve information security during the process of information transmission.

Chaos has cryptographic characteristics such as parameter sensitivity.[5–7] Recently, many image encryption schemes have been proposed, based on DNA,[8,9] neural networks,[10–12] chaos,[13–16] substitution boxes,[17–20] etc. These encryption schemes are based on the sensitivity of chaos to the initial conditions. In the past 10 years they have attracted widespread attention and achieved positive results in the field of image encryption. Wang *et al.* proposed the anti-dynamic degradation theorem, which proved that a chaotic stream cipher system is theoretically secure.[21] Therefore, chaotic cryptography has entered a new stage. The recently proposed spatiotemporal chaotic system caused fluctuations in chaotic cryptography with its excellent chaotic dynamics.[21–25] However, for the image encryption algorithm proposed in this paper, space–time chaos alone is not sufficient and it does not meet the standards of information security. Therefore, different mechanisms are needed to jointly resist the destruction of and tampering with information in the transmission process.

Middle order traversal of a binary tree is a method for traversing a binary tree in a data structure. Its traversal method is to traverse the left subtree first, then access the root node and finally traverse the right subtree. The sequence generated by this method is not sequential nor is it periodic, like cat mapping, and is easy to implement. In this paper, sequential traversal of a binary tree is applied to replace image encryption, which effectively disturbs the original position of plaintext pixels.

The encryption algorithm proposed in this paper is based on a combination of spatiotemporal chaos and sequential traversal of a binary tree. The key is generated by the plaintext image through the SHA-512 hash function, which increases the sensitivity of the plaintext. The chaotic dynamics of spatiotemporal chaos and the permutation method of order traversal in binary trees effectively increase the image confusion.

The rest of the paper is summarized as follows. Section 2 introduces the preparation work before the algorithm is implemented. Section 3 introduces the implementation of the encryption algorithm in detail. Section 4 shows the simulation results of the encryption algorithm. Section 5 is a security analysis of the algorithm. Section 6 gives a performance analysis of the algorithm. Section 7 contains our conclusions.

2. Related work

2.1. Coupled map lattices

A coupled map lattice (CML) is a dynamic system with discrete time, discrete space and a continuous state. CMLs are widely used to generate spatiotemporal chaos. A CML consists of a non-linear map located on a grid point called a local map. Each local map is coupled to other local maps according to a certain coupling rule. In fact, in computer implementations, any chaotic system must have periodicity with limited precision, but the actual period of a CML may be large enough to protect information security. CMLs were proposed

[†]Corresponding author. E-mail: 2081604413@qq.com

[‡]Xingyuan Wang has passed away.

© 2022 Chinese Physical Society and IOP Publishing Ltd

by Kaneko[26] and can be described as

$$x_{n+1}(i) = (1-\varepsilon)f(x_n(i)) + \frac{\varepsilon}{2}(f(x_n(i-1)) + f(x_n(i+1))), \tag{1}$$

where n is the time index, i $(1,2,\ldots,L)$ is the grid index and $\varepsilon \in (0,1)$ is the coupling constant. $x_n(i)$ indicates the shape of the site with time n $(1,2,\ldots)$ state variable.

The local mapping function $f(x)$ is chosen to be the logistic map:

$$x_{n+1}(i) = ux_n(1-x_n), \tag{2}$$

when $u \in (3.5699456,4)$ and $x_n \in (0,1)$ the system is in a chaotic state.

2.2. Middle order traversal of a binary tree

Middle order traversal is a kind of binary tree traversal, also called middle root traversal and middle order travel. In a binary tree, the in-order traversal first traverses the left subtree, then accesses the root node and finally traverses the right subtree. The order traversal process in the binary tree is shown in Fig. 1.

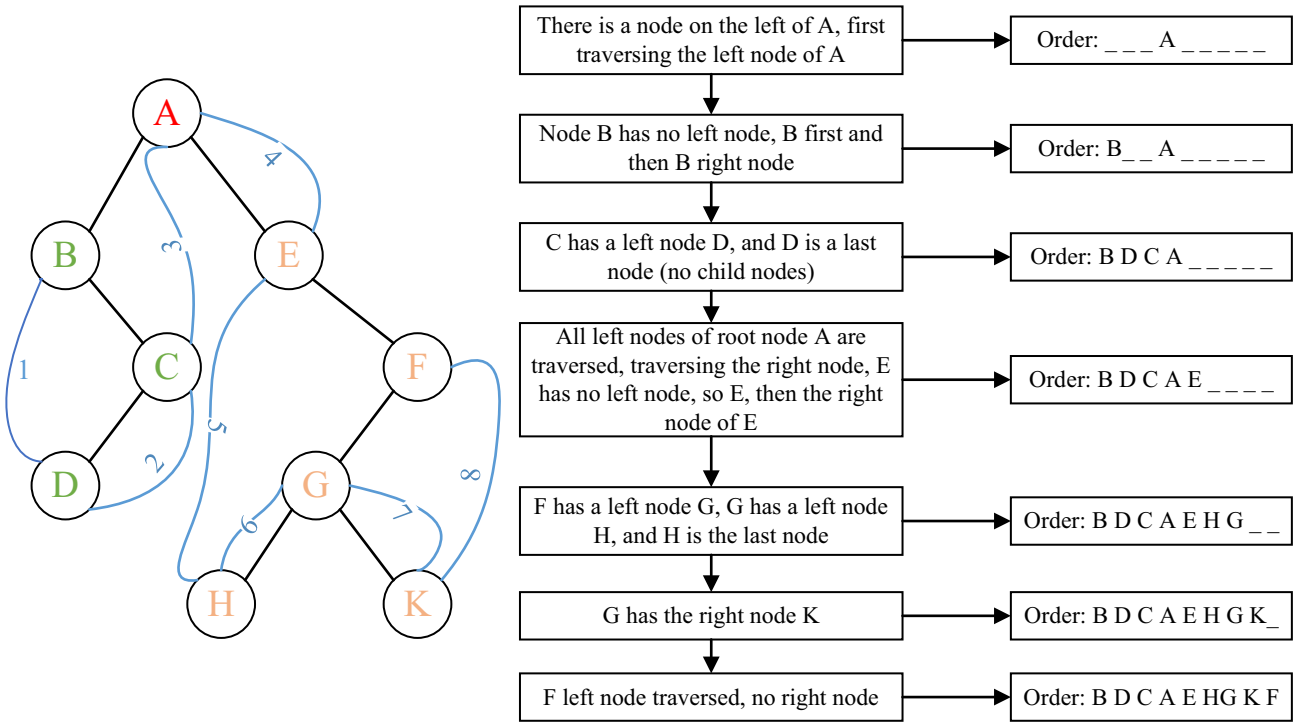


Fig. 1. Middle order traversal of a binary tree.

2.3. Replacement method based on middle order traversal of a binary tree

In this paper, middle order traversal of a binary tree is applied to the permutation of the image, and the chaos of the ordered traversal sequence in the binary tree is adopted. Due to the large number of pixel values of the image, constructing a binary tree and performing a middle order traversal operation on the pixel values of the image may be difficult to implement and increase the running speed and time of the algorithm. Therefore, this paper adopts the middle order traversal 1–65536 through VC++ 6.0 to generate the sequence.txt document, which stores the middle order traversal sequence of 1–65536 and then traverses the image pixel values according to the order. Sorting is done to complete the image replacement operation. The pseudo code that generates the middle order traversal sequence is as follows:

```
void f(int i)
{ if (i > 256 × 256) return;
  else
  { f(i × 2);
    a[k++] = i;
    f(i × 2 + 1);}
}
int main( )
{ FILE *fp = NULL;
  fp = open("sequence.txt","w");
  charstr[20];
  itoa(a[(i − 1) × 256 + j], str, 10);
  fpr int f(fp, str);
  fpr int f(fp," ");
}
```

3. Encryption algorithm

3.1. Key structure

The key system structure of this paper is shown in Fig. 2 and consists of four parts. Here ε and u are the control parameters of the CML and a_1 and a_2 are the initial values of the two coupled maps. In this paper, the key k is generated by the plaintext image through the SHA-512 hash function, which is divided into 8-bit blocks and converted into 64 decimal numbers $k_1, k_2, \ldots, k_{64}$.

$$\begin{aligned}
\varepsilon &= \mathrm{mod}((\mathrm{sum}(k_1, k_2, \ldots, k_{16})/256), 1) + a_0,\\
u &= \mathrm{mod}((\mathrm{sum}(\mathrm{k}_{17}, \mathrm{k}_{18}, \ldots, \mathrm{k}_{32})/256), 1) + a_0,\\
a_1 &= \mathrm{mod}((\mathrm{mean}(k_{33}, k_{34}, \ldots, k_{48})/256), 1) + b_0,\\
a_2 &= \mathrm{mod}((\mathrm{mean}(k_{49}, k_{50}, \ldots, k_{64})/256), 1) + b_0, \qquad (3)
\end{aligned}$$

where a_0 and b_0 are control parameters added to increase the sensitivity of the key. The function uses parameters within the range of $u \in (3.89, 4]$. u is obtained by the following transformation u:

$$u = 3.89 + 0.11 \times u. \qquad (4)$$

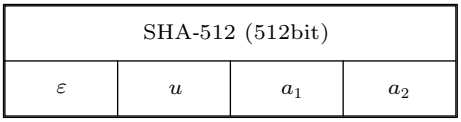


Fig. 2. Key structure.

3.2. Encryption process

The encryption process based on spatiotemporal chaos and middle order traversal of a binary tree is as follows:

Step 1: A 512-bit key k is generated by the plaintext image through the SHA-512 hash function.

Step 2: The sequence.txt file generated in Subsection 2.3 is converted into matrix I according to

$$I = \mathrm{importdata}(\mathrm{sequence.txt}). \qquad (5)$$

Step 3: Since the matrix I is a middle order traversal sequence, the plaintext image P is converted into a one-dimensional sequence, P_1, a sequence traversal sequence is used for sorting P_1 according to Eq. (6) and a permutation matrix B is generated,

$$B(i, j) = P_1(I(i, j)). \qquad (6)$$

Step 4: Equation (1) is iterated $M \times N + 500$ times according to the key in Subsection 3.1, cancelling the first 500 values and avoiding transient reaction. Finally, chaotic sequences S_1, S_2 are generated.

Step 5: According to Eq. (7), S_1, S_2 are integerized to generate S, which is used for the chaotic interference value of the diffusion,

$$S(i) = \mathrm{mod}(\mathrm{floor}(\mathrm{double}(S_1(i) + S_2(i)) \times 256), 256). \qquad (7)$$

Step 6: A ciphertext matrix C is generated by performing a XOR operation between the pixel value and the adjacent pixel value according to

$$C_1 = B_1 \oplus S_1, \quad C_{i+1} = B_{i+1} \oplus C_i \oplus S_i. \qquad (8)$$

3.3. Decryption process

The ciphertext image C is transmitted to the receiver through the common channel, and the keys are transmitted to the receiver through the key exchange protocol.[27] The decryption process is the reverse of the encryption process. Specific steps are as follows:

Step 1: The chaotic sequences S_1, S_2 are obtained by substituting the key iterative chaotic system. Sequence S is obtained according to Step 5 in Subsection 3.2.

Step 2: The inverse XOR operation is performed according to Eq. (9) to obtain B,

$$B_1 = C_1 \oplus S_1, \quad B_{i+1} = C_{i+1} \oplus C_i \oplus S_i. \qquad (9)$$

Step 3: The sequence I generated in Subsection 2.3 is used to inverse scramble B to get P_1. Finally, P_1 is recombined into plaintext image P.

4. Simulation results

In this paper, Matlab 2017 is used as a simulation tool to test the binary images 'Lena', 'Cameraman', and 'House' and a colored 'Lena' image with the encryption method described in this article. The simulation results are shown in Fig. 3. Figures 3(a)–3(e) show different plaintext images, figures 3(f)–3(j) are the corresponding ciphertext images, figures 3(k)–3(o) are the corresponding decrypted images, and figures 3(p)–3(r) illustrate the encryption and decryption process for the colored image 'Lena'. We can see that the ciphertext image has completely covered the plaintext information, and the decrypted image is no different from the plaintext image.

5. Security analysis

5.1. Key space analysis

A good encryption algorithm is extremely sensitive to the key, and the key space is large enough to resist typical attacks. In the algorithm described in this paper, the keys used are SHA-512 generated hash value and given keys a_0, b_0. When the accuracy of the key reaches 10^{-14}, the key space reaches $2^{512} \times 10^{28} \approx 2^{512} \times 2^{93} = 2^{605}$, which is much larger than 2^{200}. Table 1 compares the key space between this algorithm and other algorithms, and it can be seen from the table that the key space of this algorithm is larger than that of most algorithms. So the key space is large enough to resist a variety of typical attacks.

Table 1. Comparison of the key spaces of different algorithms.

Algorithm	This work	Ref. [28]	Ref. [29]	Ref. [30]	Ref. [31]	Ref. [32]
Key space	2^{605}	2^{161}	2^{175}	2^{156}	10^{56}	0.3×2^{158}

5.2. Key sensitivity analysis

Four sets of experiments were performed to test the sensitivity of the keys, and the color and grayscale 'Lena' images were tested separately. Figure 4 shows the test results. Figures 4(a) and 4(e) show the use of the correct key to decrypt the encrypted image; figures 4(b)–4(d) and figures 4(f)–4(h) show the respective use of the wrong key to decrypt the encrypted image. The wrong key is just a minor change to the correct key. When the plaintext image is not decrypted using the wrong key, the algorithm is sensitive to the key.

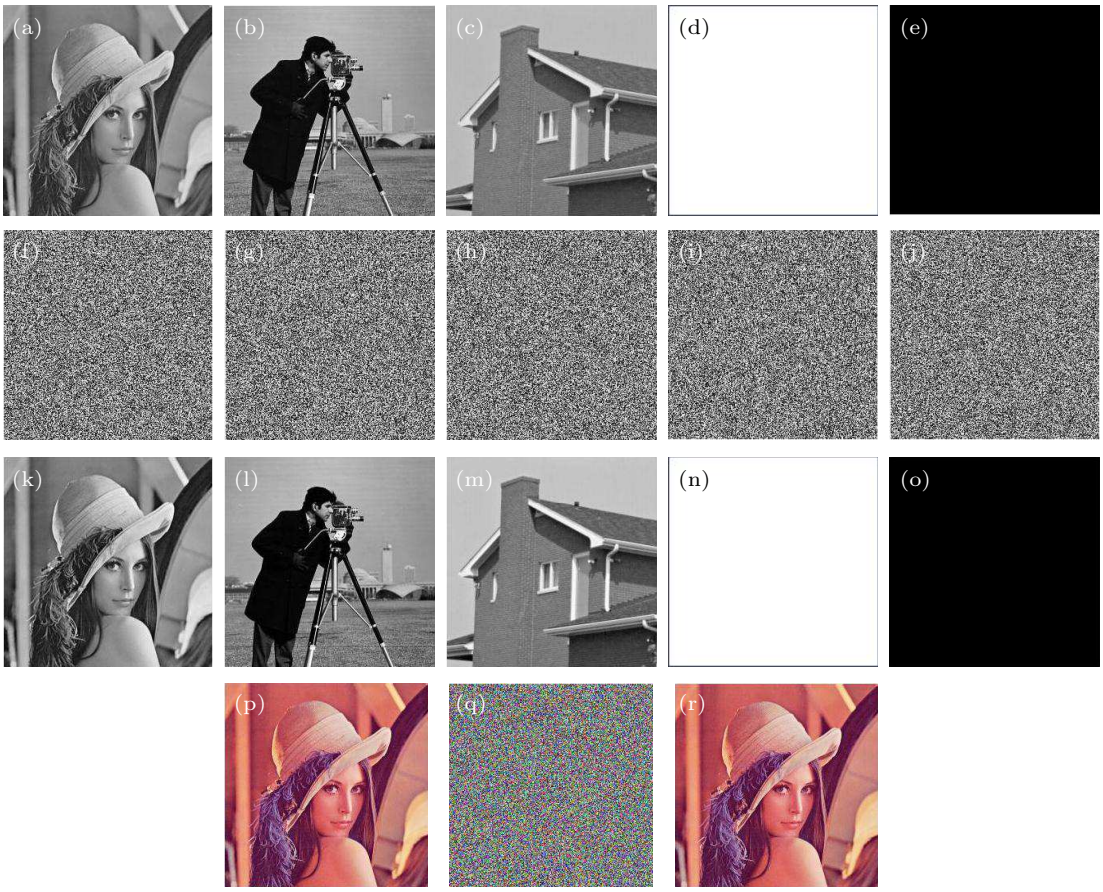


Fig. 3. Encrypted image and decrypted image of plaintext image: (a) plaintext 'Lena'; (b) plaintext 'Cameraman'; (c) plaintext 'House'; (d) plaintext white; (e) plaintext black; (f) ciphertext of 'Lena'; (g) ciphertext of 'Cameraman'; (h) ciphertext of 'House'; (i) ciphertext of white; (j) ciphertext of black; (k) decryption of 'Lena'; (l) decryption of 'Cameraman'; (m) decryption of 'House'; (n) decryption of white; (o) decryption of black; (p) plaintext color 'Lena'; (q) ciphertext of color 'Lena'; (r) decryption of color 'Lena'.

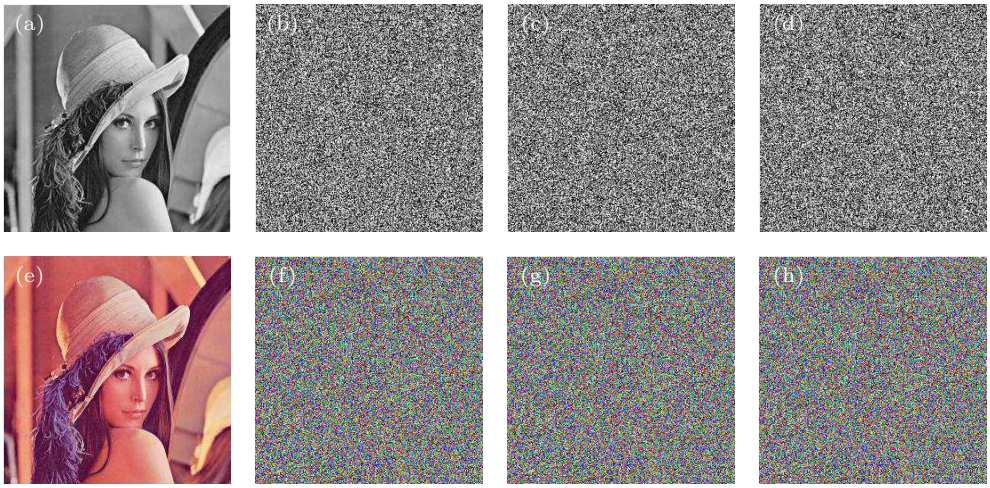


Fig. 4. Key sensitivity analysis: (a) decrypted image with the correct key; (b) decrypted image with $\varepsilon=\varepsilon+10^{-16}$; (c) decrypted image with $u=u+10^{-16}$; (d) decrypted image with $a_1=a_1+10^{-16}$; (e) decrypted image with the correct key; (f) decrypted image with $a_2=a_2+10^{-16}$; (g) decrypted image with $\varepsilon=\varepsilon+10^{-14}$; (h) decrypted image with $u=u+10^{-14}$.

6. Performance analysis

6.1. Histogram analysis

A histogram is a graph that shows the frequency at which gray values of a digital image appear. In order to hide the information of the plaintext image, the histogram of the encrypted image tends to be straight. Figure 5 shows the histogram of the plain image and the ciphertext image of the grayscale images 'Lena', 'Cameraman', and 'House' and figure 6 shows the plaintext images of the R, G, and B channels of the color image 'Lena' and the histogram of the ciphertext image. It can be seen from Figs. 5 and 6 that the histograms of the encrypted images are very similar. In order to better express the encrypted image histograms, we use the chi-square test. The critical values for the 5% and 1% probability of 255 degrees of freedom are 293.2478 and 310.457, respectively. It can be seen from Table 2 that significance levels of 5% and 1% were accepted. Obviously, the histograms of all ciphertext images tend to be flat, so it is difficult to obtain pure image information through statistical analysis.

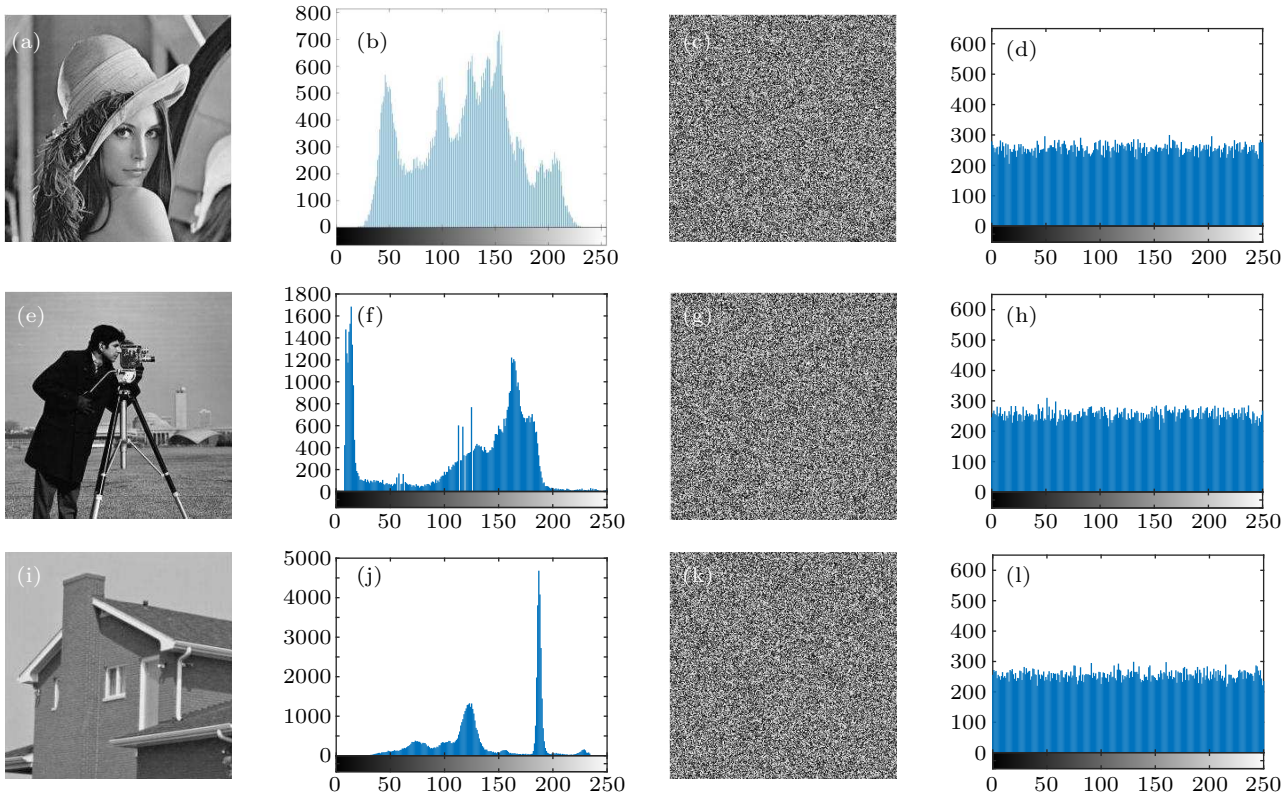


Fig. 5. Histogram analysis: (a) plaintext 'Lena'; (b) histogram of 'Lena'; (c) ciphertext of 'Lena'; (d) histogram of ciphered 'Lena'; (e) plaintext 'Cameraman'; (f) histogram of 'Cameraman'; (g) ciphertext of 'Cameraman'; (h) histogram of ciphered 'Cameraman'; (i) plaintext 'House'; (j) histogram of 'House'; (k) ciphertext of 'House'; (l) histogram of ciphered 'House'.

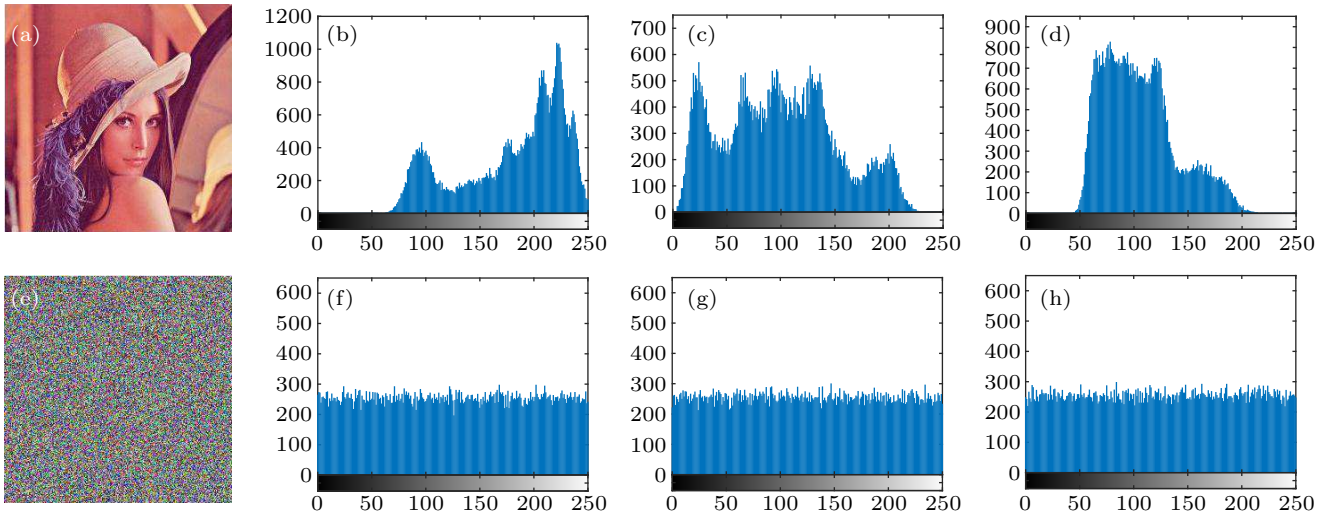


Fig. 6. Histogram analysis: (a) plaintext color 'Lena'; (b) histogram for plain R component; (c) histogram for plain G component; (d) histogram for plain B component; (e) ciphertext of color 'Cameraman'; (f) histogram for encrypted R component; (g) histogram for encrypted G component; (h) histogram for encrypted B component.

6.2. Correlation analysis

Adjacent pixels have a high correlation between plaintext pixels. In order to hide the information between the plaintext images, the correlation between the pixels is greatly reduced. The formula for calculating the correlation between pixels is as follows:

$$r_{xy} = \frac{\operatorname{cov}(x,y)}{\sqrt{D(x)}\sqrt{D(y)}}, \tag{10}$$

where

$$\operatorname{cov}(x,y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))(y_i - E(y)),$$

$$D(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))^2, \quad E(x) = \frac{1}{N}\sum_{i=1}^{N}x_i.$$

Here x and y are the gray values of two adjacent pixels. In this paper, 2000 pairs of pixels were selected to test the 'Lena', 'Cameraman' and 'House' gray images and the 'Lena' color image in clear images with Eq. (10). Figures 7–12 show the correlation between ciphertext images in the horizontal, vertical and diagonal directions.

Table 3 shows the correlation coefficients of the three plaintext images 'Lena', 'Cameraman', and 'House' and the three directions of the ciphertext image. When the correlation coefficient is close to zero, the proposed algorithm can resist statistical analysis. Experiments show that the algorithm is feasible.

Table 4 shows the plaintext image of the three channels of the 'Lena' color image and the correlation coefficients in three directions of the ciphertext image. When the correlation coefficient is close to zero, the proposed algorithm can resist statistical analysis. Experiments show that the algorithm is feasible.

Table 2. The χ^2 evaluation results of ciphertext images.

Images	'Lena'	'Cameraman'	'House'	'Lena'		
				R	G	B
χ^2	286.1406	254.0781	276.4141	232.0078	267.4688	240.8828

Table 3. Correlations of the plain-image (PI) and the corresponding cipher image (CCI) between adjacent pixels.

Image	Correlation coefficient	Direction of adjacent pixels		
		Horizontal	Vertical	Diagonal
'Lena'	PI	0.9384	0.9682	0.9129
	CCI	0.0018	−0.0020	−0.0016
'Cameraman'	PI	0.9335	0.9591	0.9083
	CCI	0.0025	0.0008	−0.0010
'House'	PI	0.9778	0.9650	0.9479
	CCI	−0.0016	−0.0021	0.0009
'Lena' in Ref. [12]	CCI	−0.0005	−0.0006	0.0002
'Lena' in Ref. [16]	CCI	0.0049	0.0037	0.0089
'Lena' in Ref. [28]	CCI	−0.0036	−0.0028	−0.0021
'Lena' in Ref. [29]	CCI	0.0041	0.0030	0.0011
'Lena' in Ref. [32]	CCI	0.0022	0.0013	0.0008
'Lena' in Ref. [34]	CCI	−0.0059	−0.0064	−0.0003

Table 4. Correlations of the color plain-image (CPI) and the corresponding cipher image (CCI) between adjacent pixels.

Image	Channel	Correlation coefficient	Direction of adjacent pixels		
			Horizontal	Vertical	Diagonal
'Lena'	Red	CPI	0.9440	0.9701	0.9208
		CCI	−0.0032	0.0028	0.00004
	Green	CPI	0.9192	0.9549	0.8978
		CCI	0.0023	−0.0015	0.0005
	Blue	CPI	0.8725	0.9119	0.8525
		CCI	−0.0001	−0.0020	−0.0008

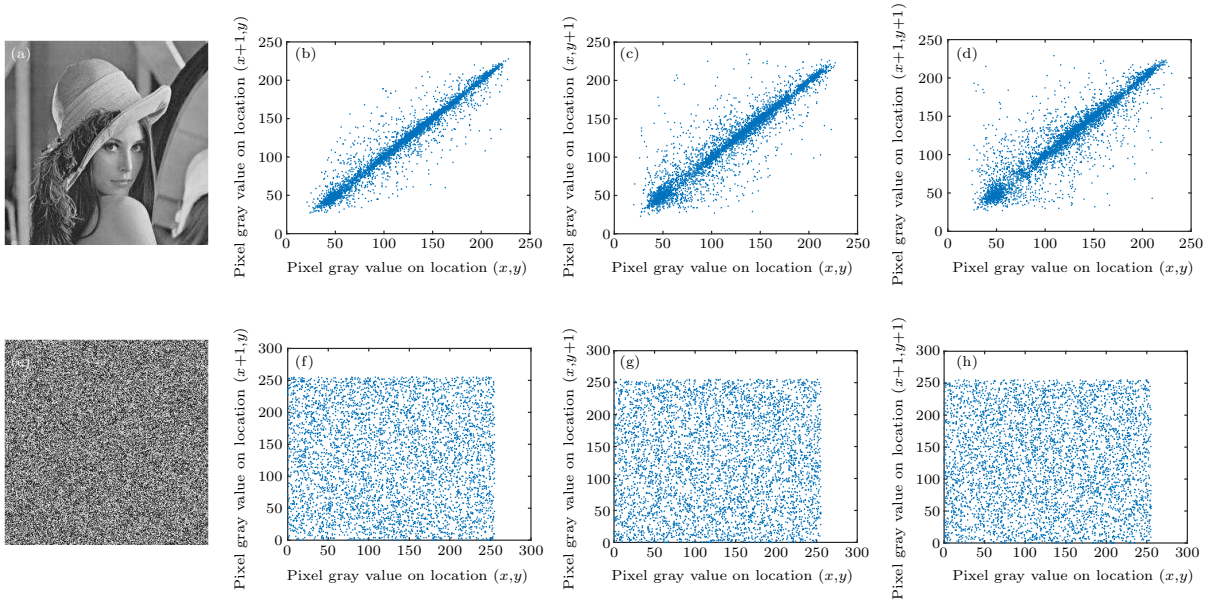


Fig. 7. Correlation analysis: (a) plaintext 'Lena'; (b) horizontal correlation of plain image; (c) vertical correlation of plain image; (d) diagonal correlation of plain image; (e) ciphertext of 'Lena'; (f) horizontal correlation of ciphered image; (g) horizontal correlation of ciphered image; (h) diagonal correlation of ciphered image.

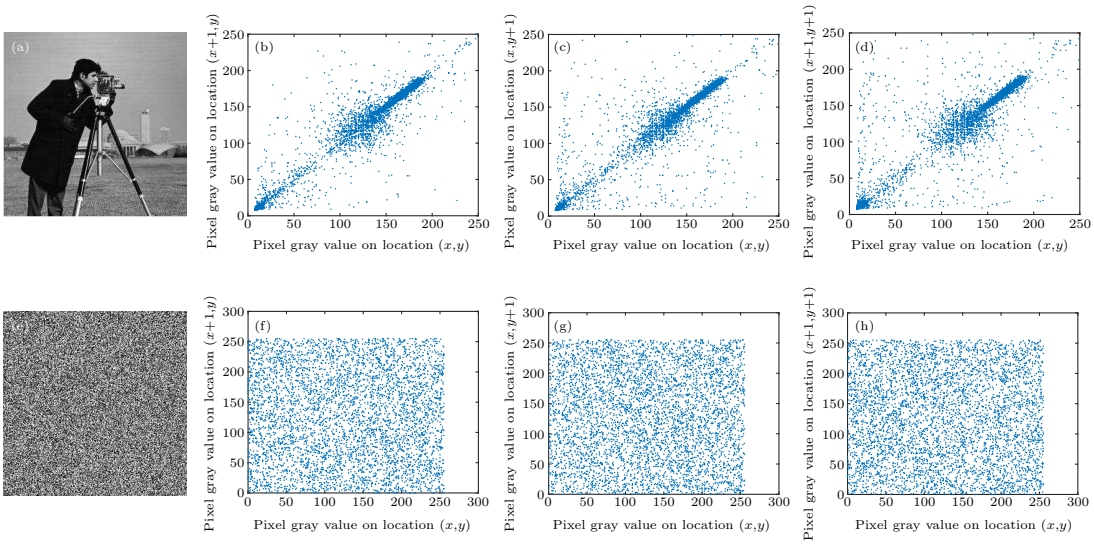


Fig. 8. Correlation analysis: (a) plaintext 'Cameraman'; (b) horizontal correlation of plain image; (c) vertical correlation of plain image; (d) diagonal correlation of plain image; (e) ciphertext of 'Cameraman'; (f) horizontal correlation of ciphered image; (g) horizontal correlation of ciphered image; (h) diagonal correlation of ciphered image.

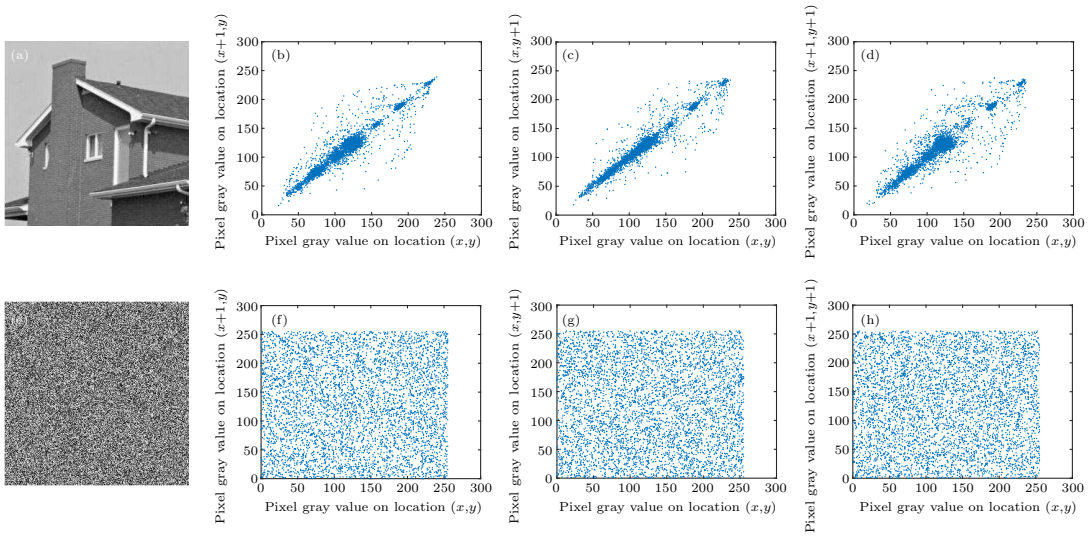


Fig. 9. Correlation analysis: (a) plaintext 'House'; (b) horizontal correlation of plain image; (c) vertical correlation of plain image; (d) diagonal correlation of plain image; (e) ciphertext of 'House'; (f) horizontal correlation of ciphered image; (g) horizontal correlation of ciphered image; (h) diagonal correlation of ciphered image.

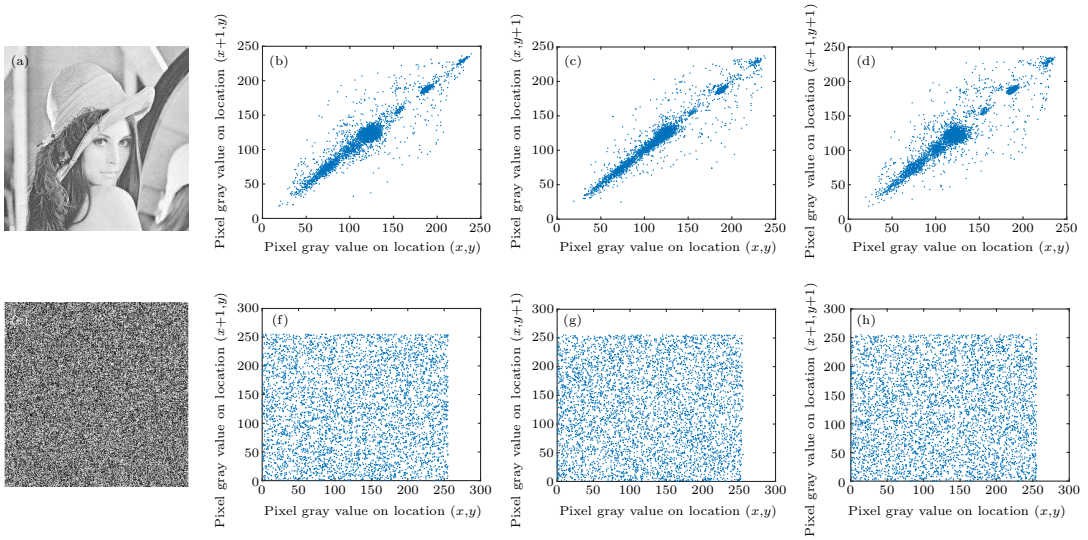


Fig. 10. Correlation analysis: (a) plaintext of R component; (b) horizontal distribution for R component; (c) vertical distribution for R component; (d) diagonal distribution for R component; (e) ciphertext of R component; (f) horizontal distribution for ciphered R component; (g) vertical distribution for ciphered R component; (h) diagonal distribution for ciphered R component.

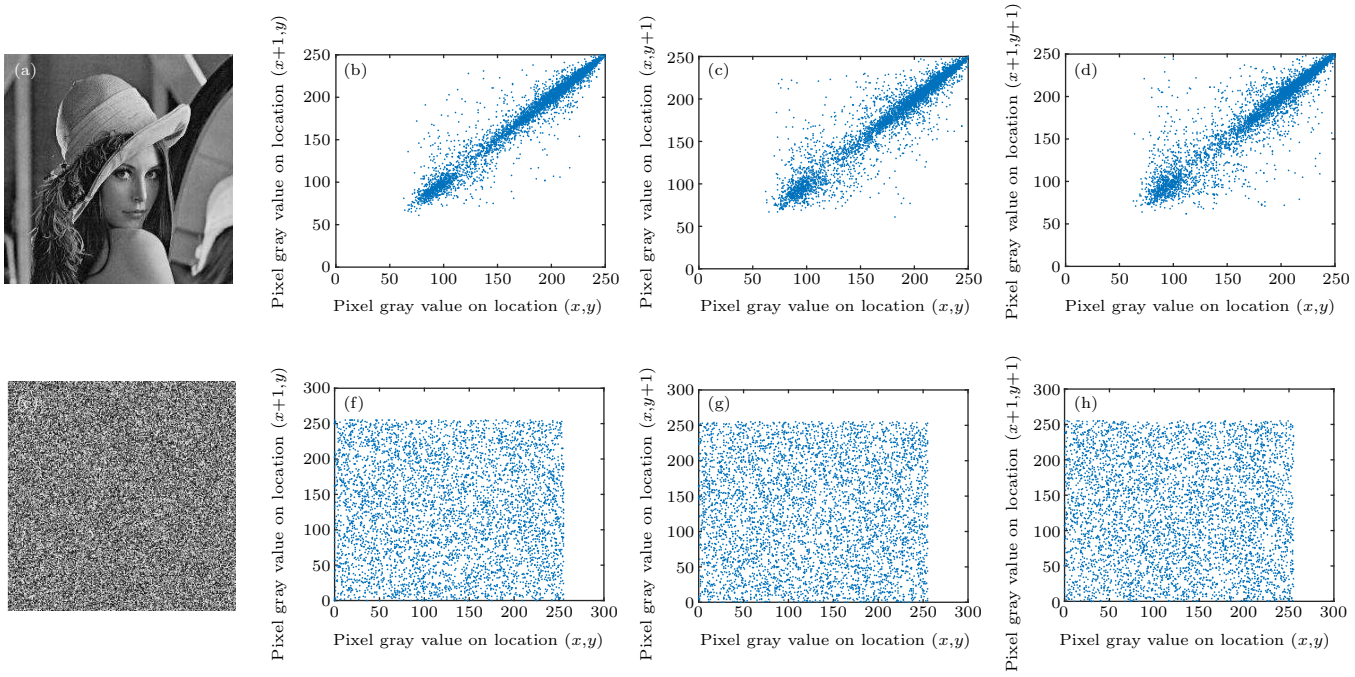


Fig. 11. Correlation analysis: (a) plaintext of G component; (b) horizontal distribution for G component; (c) vertical distribution for G component; (d) diagonal distribution for G component; (e) ciphertext of G component; (f) horizontal distribution for ciphered G component; (g) vertical distribution for ciphered G component; (h) diagonal distribution for ciphered G component.

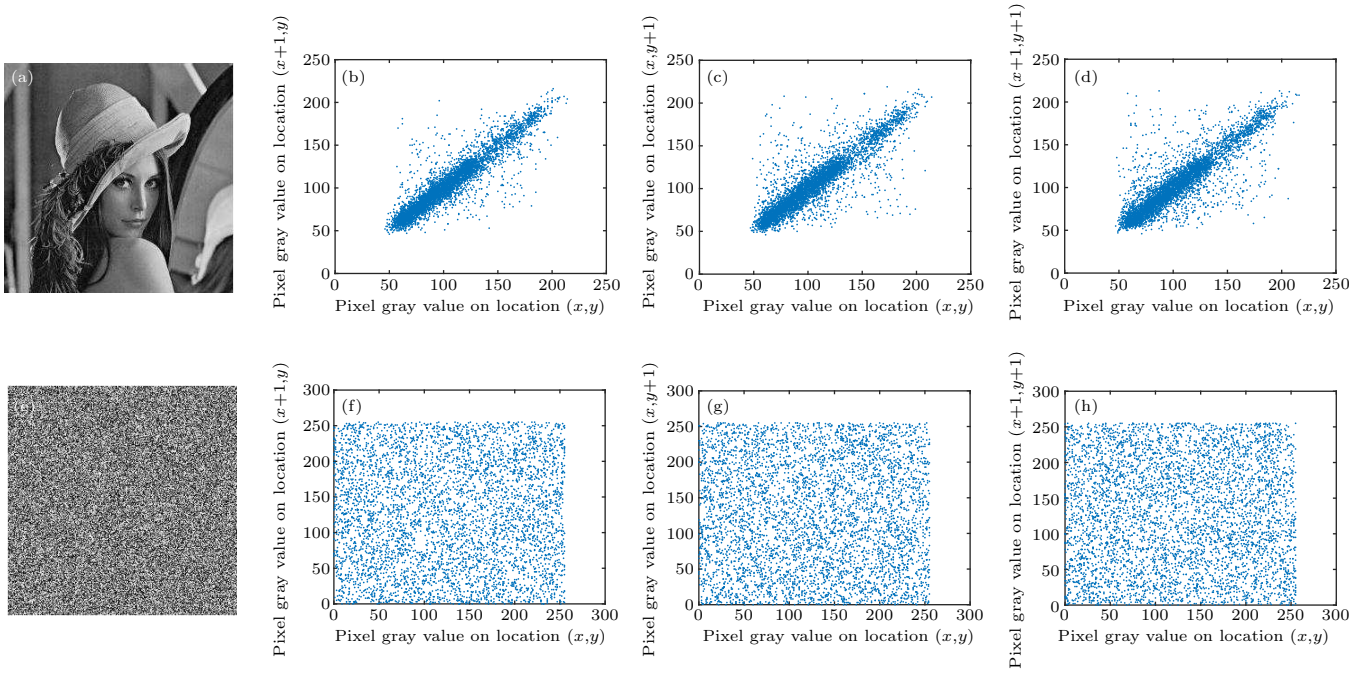


Fig. 12. Correlation analysis: (a) plaintext of B component; (b) horizontal distribution for B component; (c) vertical distribution for B component; (d) diagonal distribution for B component; (e) ciphertext of B component; (f) horizontal distribution for ciphered B component; (g) vertical distribution for ciphered B component; (h) diagonal distribution for ciphered B component.

6.3. Information entropy analysis

Information entropy is used to reflect the degree of confusion in an image. It is calculated as follows:

$$H(s) = \sum_{i=0}^{2^L-1} p(s_i) \log_2 \frac{1}{p(s_i)}, \tag{11}$$

where $p(s_i)$ represents the probability of occurrence of s_i. For an image, the ideal value for information entropy is 8. Table 5 shows the plaintext images of the three grayscale images 'Lena', 'Cameraman' and 'House' and the color image 'Lena' and the information entropy of the ciphertext images. Experiments show that the information entropy of all ciphertext images is close to 8, which means that the cipher image of the

algorithm has good randomness.

6.4. Differential attack analysis

The number of pixels change rate (NPCR) and unified average changing intensity (UACI) randomness tests are commonly used to evaluate the ability of an encrypted image to resist differential attacks. NPCR and UACI are calculated as follows:

$$\mathrm{NPCR} = \frac{\sum_{i,j} D(i,j)}{W \times H} \times 100, \tag{12}$$

$$\mathrm{UACI} = \frac{1}{W \times H}\left[\sum_{i,j} \frac{|c_1(i,j) - c_2(i,j)|}{255}\right] \times 100, \tag{13}$$

here W and H represent, respectively, the width and height of the image and c_1 and c_2 are the two ciphertext images after the original plaintext image changes by one pixel value. If $c_1(i,j) \neq c_2(i,j)$, then $D(i,j) = 1$, otherwise $D(i,j) = 0$. Table 6 shows the NPCR and UACI values for the three grayscale images 'Lena', 'Cameraman' and 'House' and the color image 'Lena'. The ideal value of NPCR is 99.6094% and for UACI it is 33.4635%. It can be seen from Table 7 that the NPCR and UACI values of this algorithm are closer to ideal values than those of most algorithms.

6.5. Robust analysis

Images are vulnerable to hijacking, tampering or destruction during transmission, so we usually use noise-adding methods to evaluate whether the encryption algorithm is robust.[34] Figure 13 shows the decryption effect after adding noise to the encrypted image. Experiments show that the image information can still be obtained after adding noise, which shows that the proposed algorithm has good robustness.

Table 5. Information entropy of the plain-image (PI) and the cipher-image (CI).

Image		Information entropy
'Lena'	PI	7.4445
	CI	7.9974
'Cameraman'	PI	6.9718
	CI	7.9973
'House'	PI	6.4970
	CI	7.9972
'Lena' (red)	PI	7.2530
	CI	7.9972
'Lena' (green)	PI	7.5923
	CI	7.9973
'Lena' (blue)	PI	6.9673
	CI	7.9973
'Lena' in Ref. [12]	CI	7.9980
'Lena' in Ref. [16]	CI	7.9967
'Lena' in Ref. [28]	CI	7.9974
'Lena' in Ref. [29]	CI	7.9968
'Lena' in Ref. [30]	CI	7.9970
'Lena' in Ref. [33]	CI	7.9971

Table 6. The average NPCR and UACI values with various images.

Image	'Lena'	'Cameraman'	'House'	'Lena' (R)	'Lena' (G)	'Lena' (B)
NPCR (%)	99.6109	99.9606	99.6154	99.9607	99.9586	99.9581
UACI (%)	33.4332	33.5634	33.3476	33.3419	33.3478	33.3486

Table 7. Comparison of NPCR and UACI for different algorithms for the image 'Lena'.

Algorithms	Ref. [12]	Ref. [16]	Ref. [28]	Ref. [29]	Ref. [32]	Ref. [33]
NPCR (%)	99.6154	99.5827	99.6142	99.5700	99.6093	99.6197
UACI (%)	33.4647	33.4791	33.4016	33.4100	33.4596	33.0443



Fig. 13. Robust analysis. (a) The encrypted image with 0.01 salt and pepper noise. (b) The encrypted image with 0.05 salt and pepper noise. (c) The encrypted image with 0.01 Gaussian noise. (d) The encrypted image with 0.05 Gaussian noise. (e) The encrypted image with 0.01 speckled noise. (f) The decrypted image with 0.01 salt and pepper noise. (g) The decrypted image with 0.05 salt and pepper noise. (h) The decrypted image with 0.01 Gaussian noise. (i) The decrypted image with 0.05 Gaussian noise. (j) The decrypted image with 0.01 speckled noise.

6.6. Time analysis

Encryption time is also a necessary condition for evaluating encryption algorithms. The experimental environment is MATLAB R2016a with AMD Ryzen 5 3500H CPU and 8 GB RAM. We encrypt the 'Lena' image 50 times, take the average value and compare it with other algorithms to get Table 8. It can be seen from the table that the encryption time of this algorithm is shorter than that of most of the other algorithms, which shows that encryption of this image is faster.

Not only experimental time analysis but also theoretical time complexity analysis should be carried out. Assuming the size of the image is $M \times N$, Step 1 generates the hash value whose time complexity is $O(512)$. The required binary tree sequence is obtained in Step 2, with the time complexity of $O(M \times N)$. In Step 3, the binary tree sequence is used to scramble the image, whose time complexity is $O(M \times N)$. Step 4 is to iterate the chaotic system and generate two chaotic sequences, whose time complexity is $2O(M \times N)$. In Step 5, the required XOR sequence is generated, whose time complexity is $O(M \times N)$. In Step 6 the ciphertext image is obtained from the XOR scrambled image, and its time complexity is $O(M \times N)$. Therefore, the time complexity of the algorithm is $2O(M \times N)$. Table 9 compares the time complexity between the present algorithm and other algorithms, and shows the present algorithm is superior to most of the other algorithms.

Table 8. Time comparisons for different algorithms.

Algorithms	This work	Ref. [33]	Ref. [35]	Ref. [36]	Ref. [37]
Time (s)	0.4409	0.3381	0.5994	1.3005	1.3107

Table 9. Comparison of time complexity for different algorithms.

Algorithms	This work	Ref. [35]	Ref. [36]	Ref. [37]
Time (s)	$2O(M \times N)$	$4O(M \times N \log MN)$	$5O(M \times N)$	$2O(M \times N \log MN)$

7. Conclusion

The encryption algorithm proposed in this paper is based on a combination of spatiotemporal chaos and sequential traversal of a binary tree. The key is generated by the plaintext image through the SHA-512 hash function, which increases the sensitivity of the plaintext. The chaotic dynamics of spatiotemporal chaos and the permutation method of order traversal in binary trees effectively increase the image confusion. Diffusion uses an adjacent or exclusive XOR based on chaotic interference values, which makes ciphertext images resistant to differential attacks.

Acknowledgments

Project supported by the National Natural Science Foundation of China (Grant No. 61672124), the Password Theory Project of the 13th Five-Year Plan National Cryptography Development Fund (Grant No. MMJJ20170203), Liaoning Province Science and Technology Innovation Leading Talents Program Project (Grant No. XLYC1802013), Key Research and Development Projects of Liaoning Province, China (Grant No. 2019020105-JH2/103), and Jinan City '20 universities' Funding Projects Introducing Innovation Team Program (Grant No. 2019GXRC031).

References

[1] Wang X, Wang X, Ma B, Li Q and Shi Y Q 2021 *IEEE Signal Process. Lett.* **28** 1125
[2] Wang X Y, Liu C and Jiang D H 2021 *Inf. Sci.* **574** 505
[3] Xian Y J, Wang X Y and Teng L 2021 *IEEE Trans. Circuits Syst. Video Technol.*
[4] Li Q, Wang X Y, Ma B, Wang X Y, Wang C P, Gao S and Shi Y Q 2021 *IEEE Trans. Circuits Syst. Video Technol.*
[5] Liu S T and Wang P 2018 *Fractal Control Theory* (Singapore: Springer)
[6] Liu S T, Zhang Y P and Liu C A 2020 *Fractal Control and Its Applications* (Singapore: Springer)
[7] Liu S T, Wang Y P, Bi Z M and Wang Y 2021 *Mathematical Principle and Fractal Analysis of Mesoscale Eddy* (Singapore: Springer)
[8] Girdhar A and Kumar V 2018 *Multimed. Tools Appl.* **77** 27017
[9] Zhang Y Q, Wang X Y, Liu J and Chi Z L 2016 *Opt. Lasers Eng.* **82** 95
[10] Wang X Y and Gao S 2020 *Inf. Sci.* **507** 16
[11] Ratnavelu K, Kalpana M, Balasubramaniam P, Wong K and Raveendran P 2017 *Signal Process.* **140** 87
[12] Norouzi B and Mirzakuchaki S 2017 *Multimed. Tools Appl.* **76** 13681
[13] Wang X Y and Gao S 2020 *Inf. Sci.* **539** 195
[14] Wang X Y and Zhang M Z 2021 *Inf. Sci.* **579** 128
[15] Wang X Y and Liu P B 2022 *IEEE Trans. Circuits Syst. I: Regul. Pap.* **69** 1291
[16] Mahmud M, Lee M and Choi J Y 2020 *Opt. Laser Technol.* **121** 105818
[17] Liu H, Zhao B and Huang L 2019 *Entropy* **21** 343
[18] Wang X, Cavusoglu U, Kacar S, Akgul A, Pham V T, Jafari S, Alsaadi F E and Nguyen X Q 2019 *Appl. Sci.* **9** 781
[19] Zhu C, Wang G and Sun K 2018 *Symmetry* **10** 399
[20] Zhang Y 2018 *Inf. Sci.* **450** 361
[21] Wang M X, Wang X Y, Zhao T T, Zhang C, Xia Z Q and Yao N M 2021 *Inf. Sci.* **544** 1
[22] Zhang Y Q, Wang X Y, Liu L Y and Liu J 2018 *Int. J. Bifurcation Chaos* **28** 1850020
[23] Wang X Y, Wang Y, Wang S W, Zhang Y Q and Wu X J 2018 *Chin. Phys. B* **27** 110502
[24] Guo S F, Liu Y, Gong L H, Yu W Q and Gong Y L 2018 *Multimed. Tools Appl.* **77** 21109
[25] Wang X Y and Yang J J 2021 *Inf. Sci.* **569** 217
[26] Kaneko K 1989 *Physica D* **34** 1
[27] Wang X Y (U.S. Patent) 9742558 B2 2017-08-22
[28] Firdous A, Rehman A U and Missen M M S 2021 *IEEE Access* **9** 11675
[29] Nardo L G, Nepomuceno E G, Arias-Garcia J and Butusov N D 2019 *Chaos Solitons Fractals* **123** 69
[30] Ullah A, Jamal S S and Shah T 2018 *Nonlinear Dyn.* **91** 359
[31] Zhang Y Q, He Y, Li P and Wang X Y 2020 *Opt. Lasers Eng.* **128** 106040
[32] Chen C, Sun K and He S 2020 *Signal Process.* **168** 107340
[33] Jun W J and Fun T S 2021 *IEEE Access* **9** 120596
[34] Wang X Y and Su Y N 2020 *Sci. Rep.* **10** 18556
[35] Hua Z and Zhou Y 2016 *Inf. Sci.* **339** 237
[36] Mondal B, Kumar P and Singh S 2018 *Multimed. Tools Appl.* **77** 31177
[37] Muhammad K, Hamza R, Ahmad J, Lloret J, Wang H X and Baik S W 2018 *IEEE Trans. Ind. Inform.* **14** 3679