Optik - International Journal for Light and Electron Optics 171 (2018) 891–903

Contents lists available at ScienceDirect

Optik

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

Original research article

A novel image encryption algorithm based on self-orthogonal Latin squares

Ming Xu[a,b], Zihong Tian[a,*]

[a] College of Mathematics and Information science, Hebei Normal University, Shijiazhuang, China
[b] Department of Mathematics and Physics, Shijiazhuang Tiedao University, Shijiazhuang, China

ARTICLE INFO

Keywords:
Image encryption
Self-orthogonal Latin squares
Security
Efficiency

ABSTRACT

This paper proposes a novel image encryption algorithm based on self-orthogonal Latin squares. A self-orthogonal Latin square can generate a 2D map and some 1D maps which can be used for permutations of image. It can also provide a pseudo-random sequence which can be used for substitutions of image. Simulation results show that the proposed algorithm has not only a desirable level of security, but also high efficiency, so the algorithm is suitable for practical application.

1. Introduction

Accompanying with the acceleration of globalization and development of communication technology, prodigious amounts of image information, public or private, are transmitted via public communication channels. How to transmit the great deal of image information securely and efficiently has become an important problem which attracts more and more attention.

Due to the specific characteristics of image data, such as relatively large block length and higher dimension, the traditional block cipher is somewhat unsuitable for digital images. Therefore, a variety of encryption algorithms specific to image have been proposed, such as schemes based on chaos [6,9,11,14,15,20,22,23,25,26,30], schemes based on DNA [4,7,10,16,19], and schemes based on genetic algorithm [1,17]. However, many of them have some defects, and then have been broken by chosen-plaintext attack or known-plaintext attack [27,28,29].

Latin square is a specific type of square matrix with uniformity. The relation between Latin square and cryptography was pointed out first by Shannon in his classical paper [13]. Latin square has some good characteristics which are very suitable for image encryption:

1) The number of Latin squares is huge. For example, the number of Latin squares at order 10 is about 10^{37}. When the order increases, the number of Latin squares will rise sharply. It implies that we can generate a large number of Latin squares to encrypt images with arbitrary size. The key space is huge, so it is secure against brute-force attack;
2) Latin squares have uniform histogram. It implies that the ciphertext image after Latin square whitening is secure against statistical analysis;
3) Compared with chaotic sequences which are used extensively in image encryption, Latin squares are directly defined over integer. They don't need discretization;
4) Latin squares have matrix form, which is coincident with the image data.

* Corresponding author.
E-mail address: tianzh68@163.com (Z. Tian).

https://doi.org/10.1016/j.ijleo.2018.06.112
Received 10 January 2018; Received in revised form 16 May 2018; Accepted 27 June 2018

$$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 3 & 0 & 1 \\ 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 4 & 3 & 2 & 1 \\ 2 & 1 & 0 & 4 & 3 \\ 4 & 3 & 2 & 1 & 0 \\ 1 & 0 & 4 & 3 & 2 \\ 3 & 2 & 1 & 0 & 4 \end{bmatrix}$$

Fig. 1. Examples of Latin squares of orders 3, 4, 5.

In view of these good characteristics, some image encryption algorithms based on Latin squares have been proposed. Wu et al. introduce a Latin square image cipher (LSIC) for both grayscale and color images in [18]. They design a new loom-like substitution-permutation network which can produce good confusion and diffusion effects. Panduranga et al. utilize chaotic map and Latin square to construct an image encryption algorithm in [12]. However, it is broken later by Ahmad et al. in [2]. Recently, Hu et al. use Latin squares to construct a 3D lookup table in [5]. This lookup table is used to perform bit-level permutation dynamically, i.e. the permutation operation is plaintext-related. In this paper, we design an image encryption algorithm based on self-orthogonal Latin square(SOLS). Compared with ordinary Latin square, SOLS has some particular properties which will be introduced detailedly in Section 2.

Most of the existing image ciphers are based on Fridrich architecture which is first introduced by Fridrich in [30]. Some improvements [23–25] have also been proposed. In this paper, we adopt a "permutation-substitution-permutation" architecture in one round encryption, and use one SOLS to perform the whole encryption process. The permutation operations at top level and bottom level can protect the substitution operation at middle level, considering the XOR operations in substitution stage are easily broken by differential attack. Moreover, the diffusion effect is contributed not only by the substitution stage, but also by the permutation stage at bottom level. Therefore the algorithm can achieve a good diffusion effect (Fig. 1).

The rest of this paper is organized as follows: In Section 2, we introduce some basic definitions and conclusions which will be used in the proposed algorithm. The concrete procedures of encryption algorithm and decryption algorithm are described in Section 3. Some simulation results and security analysis are given in Section 4. Finally, we conclude the paper in Section 5.

2. Related work

2.1. Latin squares and self-orthogonal Latin squares

A *Latin square* of side n (or order n) is an $n \times n$ array in which each cell contains a single symbol from an n-set S, such that each symbol occurs exactly once in each row and exactly once in each column. In this paper, we let $\boldsymbol{S} = \boldsymbol{Z}_n = \{0,1,\ldots, n-1\}$, and use $L(i, j)$ to denote the element in the ith row and jth column of Latin square L.

The *transpose* of Latin square L, denoted by L^T, is the Latin square which results from L when the roles of rows and columns are exchanged(that is, $L^T(i, j) = L(j, i)$).

Two Latin squares L and L' of the same order are *orthogonal* if $L(a, b) = L(c, d)$ and $L'(a, b) = L'(c, d)$, implies $a = c$ and $b = d$.

A *self-orthogonal Latin square* (SOLS) is a Latin square that is orthogonal to its transpose.

The existence of self-orthogonal Latin square(SOLS) is guaranteed for every order $n \in \boldsymbol{N}$, except for $n = 2, 3, 6$. Notable existence proof and construction results for SOLS appear in [3].

Using SOLS we can construct a 2D permutation in the plane. For example, using the SOLS of order 4 in Fig. 2 and its transpose, we can obtain a new matrix C as follows:

$$C = \begin{bmatrix} (0,0) & (1,3) & (2,1) & (3,2) \\ (3,1) & (2,2) & (1,0) & (0,3) \\ (1,2) & (0,1) & (3,3) & (2,0) \\ (2,3) & (3,0) & (0,2) & (1,1) \end{bmatrix}$$

According to the orthogonality, each ordered pair (i, j) $(i, j = 0, 1, 2, 3)$ occurs exactly once in matrix C. Then matrix C can produce a 2D permutation satisfying $(0, 0) \rightarrow (0, 0)$, $(0, 1) \rightarrow (1, 3)$, $(0, 2) \rightarrow (2, 1)$, …, $(3, 3) \rightarrow (1, 1)$.

Compared with the permutation method based on ordinary Latin squares, such as [18], the permutation method based on SOLS is

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 3 & 2 & 1 & 0 \\ 1 & 0 & 3 & 2 \\ 2 & 3 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 2 & 1 & 4 & 3 \\ 3 & 1 & 4 & 0 & 2 \\ 4 & 3 & 2 & 1 & 0 \\ 2 & 4 & 0 & 3 & 1 \\ 1 & 0 & 3 & 2 & 4 \end{bmatrix} \quad \begin{bmatrix} 0 & 2 & 1 & 4 & 3 & 6 & 5 \\ 3 & 1 & 6 & 0 & 5 & 4 & 2 \\ 4 & 5 & 2 & 6 & 0 & 1 & 3 \\ 5 & 6 & 4 & 3 & 2 & 0 & 1 \\ 6 & 3 & 5 & 1 & 4 & 2 & 0 \\ 1 & 0 & 3 & 2 & 6 & 5 & 4 \\ 2 & 4 & 0 & 5 & 1 & 3 & 6 \end{bmatrix}$$

Fig. 2. Examples of SOLS of orders 4, 5 and 7.

more efficient. One ordinary Latin square can only produce some 1D maps. To permute the 2D image using these 1D maps, we need to permute the image row by row and column by column. It is time consuming. However, using the 2D permutation generated from SOLS, we can permute the image directly. The simulation results in Section 4 show that the effect of one round encryption in the proposed algorithm is very close to the effect of 8 rounds encryption in [18].

2.2. Logistic map

In the proposed algorithm, we use Logistic map to produce a sequence. Then this sequence will be used to construct a SOLS. The Logistic map is described by Eq. (1).

$$x_{n+1} = \mu x_n(1 - x_n)\ n = 0, 1, 2, 3... \tag{1}$$

where μ is a system parameter, $0 \le \mu \le 4$. x_n is a floating number in (0,1). When $\mu > 3.573815$, this system exhibits chaotic characteristics.

3. The proposed image encryption algorithm

In the rest of this paper, N denotes a prime power. P denotes a $N \times N$ plaintext image, and C denotes corresponding ciphertext image. L denotes a keyed SOLS of order N, and K denotes the encryption key.

3.1. The generation of SOLS

We use Algorithm 1 to generate a SOLS. Compared with other generation algorithms, such as search algorithm, the algorithm constructs SOLS directly on the finite field F_N, thus it is more efficient.

Algorithm 1. Generation of a SOLS of order N ($L = \mathrm{SLG}(key_0, \mu_0, t)$)

Input: key_0 is initial value of Logistic map, μ_0 is system parameter of Logistic map, t is a parameter satisfying $t \in F_N$, $t \notin \{0, 1\}$ and $2t \neq 1$.

Output: SOLS L of order N.

Step 1: Generate a chaotic sequence $x = (x_0, x_1, ...,x_{N-1})$ through Logistic map with initial value key_0 and system parameter μ_0.

Step 2: Prepare the chaotic sequence x as follows:

$$[lx, fx] = sort(x) \tag{2}$$

where [,] = sort() is the sequencing index function. fx is the new sequence after ascending to x, lx is the index value of fx. Denote lx as $lx = \{c_0, c_1, ...,c_{N-1}\}$.

Step 3: Construct a finite field F_N on $lx = \{c_0, c_1, ...,c_{N-1}\}$, that is, redefine "+" and "×" on lx to meet the definition of finite field. Then construct SOLS L of order N as follows:

```
for i=0:1:N−1 do
    for j=0:1:N−1 do
        L(i,j)←t×ci+(1−t)×cj        (3)
    end for.
end for.
```

where "×", "+" and " - " denote the multiplication、addition and subtraction respectively in finite field F_N. $L(i, j)$ is the element in the generated SOLS.

3.2. Image encryption

The proposed encryption algorithm includes three parts. Firstly, the original image P is scrambled by SOLS L and its transpose as described in Section 2.1. Secondly, we perform XOR operation between the scrambled image and L to change the distribution of gray-level histogram, meanwhile, achieve the diffusion effect. At last, permute the diffused image once more by the bijections generated from the rows and columns of L. In the last stage, we also add some operations to strengthen the diffusion effect. The schematic block diagram is shown in Fig. 3.

The encryption process is described detailedly by Algorithm 2.

Algorithm 2. The proposed encryption algorithm ($C = \mathrm{Œ}(P, K, t)$)

Input: P is a $N \times N$ grayscale image, $K = (key_0, \mu_0, key_1, \mu_1)$ is the secret key, t is a parameter satisfying $t \in F_N$, $t \notin \{0, 1\}$ and $2t \neq 1$.

Output: C is the ciphertext image.

Step 1: Use Algorithm 1, key_0, μ_0, and t to generate SOLS $L = \mathrm{SLG}(key_0, \mu_0, t)$.

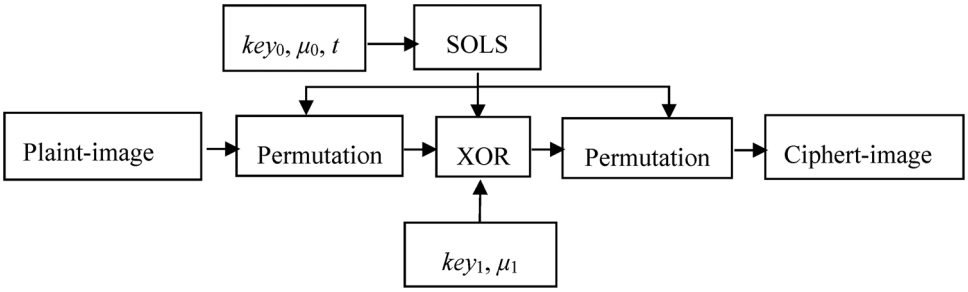

Fig. 3. Block diagram for the image encryption algorithm.

Step 2: Use L to scramble P following Eq. (4), and obtain the scrambled image P_1.

$$P_1[i,j] \longleftarrow P[L[i,j],\ L[j,i]] \quad 0 \le i,j < N \tag{4}$$

Step 3: Scan all the pixels of P_1 from the upper left corner to the lower right corner to generate a sequence $\{p[i]\}_{i=0}^{N^2-1}$, and scan all the elements of L in the same manner to generate a sequence $\{L[i]\}_{i=0}^{N^2-1}$. Each pixel value of P_1 is altered sequentially by Eq. (5), then a sequence $\{m[i]\}_{i=0}^{N^2-1}$ is obtained. Rearrange sequence $\{m[i]\}_{i=0}^{N^2-1}$ into an $N \times N$ image P_2.

$$\begin{cases} m'[i] = m[i]/1000 \\ f(m'[i]) = \mu_1 \times (m'[i]) \times (1 - m'[i]) \\ M[i+1] = (f(m'[i]) \times 1000 + L[i+1]) \bmod 256 \\ m[i+1] = p[i+1] \oplus M[i+1] \end{cases} \tag{5}$$

In Eq. (5), the initial value $m[-1] = \{[\mu_1 \times (key_1) \times (1 - key_1)] \times 10^{14}\} \bmod 256$. $M[i]$ and $m'[i]$ are temporary variables.

Step 4: Use L to generate N row bijections $\{r_i(k)\}_{i=0}^{N-1}$ and N column bijections $\{c_j(k)\}_{j=0}^{N-1}$, where $r_i(k) \triangleq L[i,\ k]$ and $c_j(k) \triangleq L[k,\ j]$. Firstly permute the pixels of each row in P_2 using the N row bijections, and obtain a temporary image P_3. Secondly, permute the pixels of each column in P_3 using the N column bijections, and then obtain the final ciphertext image C. The specific process is described as follows:

Let k_1=0, k_2=0;
for i=0: 1: $N-1$ do
  for j=0:1: $N-1$ do
    $P_3[i,j] \leftarrow P_2[i, r_{k_1}[j]]$; (6)
  end for.
  $k_1 \leftarrow P_3[i, N-1]$;
end for.
for j=0:1: $N-1$ do
  for i=0:1: $N-1$ do
    $C[i,j] \leftarrow P_3[c_{k_2}[i], j]$; (7)
  end for.
  $k_2 \leftarrow C[N-1, j]$;
  end for.

The decryption process is an inverse process of encryption. It is described detailedly by Algorithm 3.

Algorithm 3. The proposed decryption algorithm $P = \mathbb{D}(C, K, t)$

Input: C is a $N \times N$ ciphertext image, $K = (key_0, \mu_0, key_1, \mu_1)$ is the secret key, t is a parameter satisfying $t \in F_N$, $t \notin \{0, 1\}$ and $2t \neq 1$.
Output: P is the recovered plaintext image.
Step 1: Use Algorithm 1, key_0, μ_0, and t to generate SOLS $L = \mathrm{SLG}(key_0, \mu_0, t)$.
Step 2: Use L to generate N row bijections $\{r_i(k)\}_{i=0}^{N-1}$ and N column bijections $\{c_j(k)\}_{j=0}^{N-1}$, where $r_i(k) \triangleq L[i,\ k]$ and $c_j(k) \triangleq L[k,\ j]$. Firstly, permute the pixels of each column in C using the N column bijections, and obtain a temporary image P_3. Secondly, permute

the pixels of each row in P_3 using the N row bijections, and obtain image P_2. The specific process is described as follows:

Let k_1=0, k_2=0;
for j=0:1: $N-1$ do
  for i=0:1: $N-1$ do
    $P_3[c_{k_2}[i], j] \leftarrow C[i, j]$; (8)
  end for.
  $k_2 \leftarrow C[N-1, j]$;
end for.

for i=0: 1: $N-1$ do
  for j=0:1: $N-1$ do
    $P_2[i, r_{k_1}[j]] \leftarrow P_3[i, j]$; (9)
  end for.
  $k_1 \leftarrow P_3[i, N-1]$;
 end for.

Step 3: Scan all the pixels of P_2 from the upper left corner to the lower right corner to generate a sequence $\{m[i]\}_{i=0}^{N^2-1}$, and scan all the elements of L in the same manner to generate a sequence $\{L[i]\}_{i=0}^{N^2-1}$. Each pixel value of P_2 is altered sequentially by Eq. (10), then a sequence $\{p[i]\}_{i=0}^{N^2-1}$ is obtained. Rearrange sequence $\{p[i]\}_{i=0}^{N^2-1}$ into an $N \times N$ image P_1.

$$\begin{cases} m'[i] = m[i]/1000 \\ f(m'[i]) = \mu_1 \times (m'[i]) \times (1 - m'[i]) \\ M[i+1] = (f(m'[i]) \times 1000 + L[i+1]) \bmod 256 \\ p[i+1] = m[i+1] \oplus M[i+1] \end{cases} \quad (10)$$

In Eq. (10), the initial value $m[-1] = \{[\mu_1 \times (key_1) \times (1 - key_1)] \times 10^{14}\} \bmod 256$. $M[i]$ and $m'[i]$ are temporary variables.

Step 4: Use L to scramble image P_1 by Eq. (11), and then obtain the plaintext image P.

$$P[L[i, j],\ L[j, i]] \longleftarrow P_1[i, j] \quad 0 \le i, j < N \quad (11)$$

In the proposed algorithm, the size of image is restricted to prime power. It does not make much sense to further generalize the encryption algorithm to images with arbitrary size. Because an image is usually compressed for convenience before it is encrypted, and we have to pad the compressed image to the block size anyway. However, if we have to do so, we only need to pad the image with chaotic sequence or cut the image into fixed size.

4. Simulation results and security analysis

The simulation results and security analysis of the proposed algorithm are given in this section. Meanwhile, we give comparisons between our algorithm and four representative algorithms [9,16,18,26].

All the tests are performed under C#2010, running on Microsoft Windows XP with intel core i5-3320 M 2.60 GHz processor and 3.22 GB RAM. In these tests, we use different 256 × 256 images for encryption and decryption. The secret key K of one round encryption is set as: $key_0 = 0.12345678$, $key_1 = 0.23456789$, $\mu_0 = \mu_1 = 3.99999$. The parameter is set as $t = 3$.

4.1. Key space and sensitivity analysis

4.1.1. Key space analysis

The two initial values and two system parameters of Logistic map are set as the secret key K of the proposed algorithm. If the four values all have a computational precision of 10^{-14}, the key space can reach $10^{54} \approx 2^{180}$, which is large enough to resist the brute-force attack for the moment.

Considering that the computing power is increasing year by year, to resist possible brute-force attack in the future, many image encryption algorithms with huge key space have been proposed [5,22,24]. For example, the algorithm in [22] adopts a linear hyperbolic chaotic system of partial differential equations with many parameters and initial values, thus its key space can reach 2^{250} which is a very large number. In fact, our algorithm can also reach a large key space in theory. From section 3, we can see that the

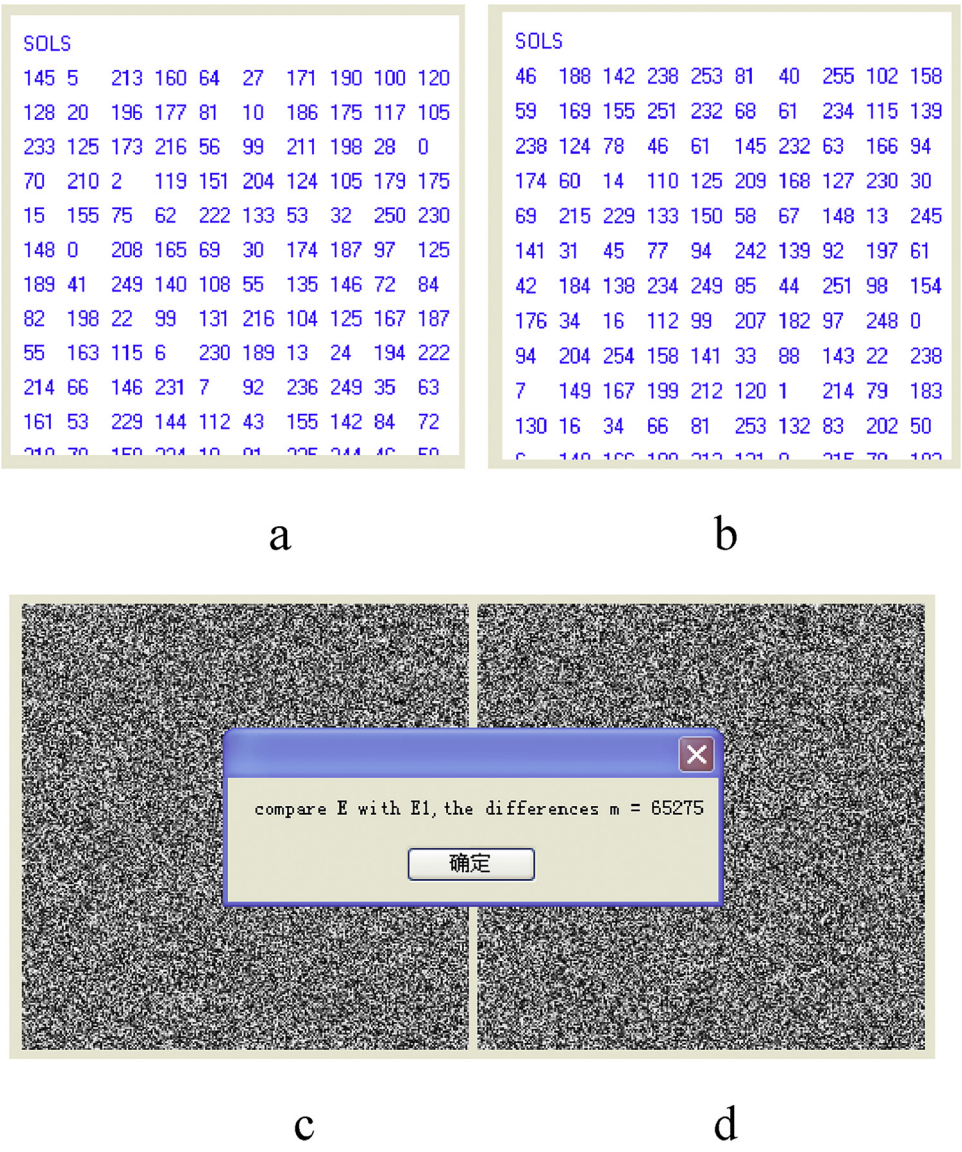

Fig. 4. (a) Part of SOLS corresponding to initial value 0.12345678.
(b)Part of SOLS corresponding to initial value $0.12345678 + 10^{-14}$.
(c)Cipher-image of Lena corresponding to initial value 0.12345678.
(d)Cipher-image of Lena corresponding to initial value $0.12345678 + 10^{-14}$.

algorithm is mainly based on the SOLS generated from Algorithm 1. In Algorithm 1, the SOLS is constructed by a sequence lx with length N. Then for an 256×256 image, there are at most 256! SOLS available for the cryptosystem. It means that the key space can be enlarged flexibly to satisfy the actual needs in the future.

4.1.2. Key sensitivity analysis

(1) Key sensitivity analysis in encryption stage

Firstly, we modify key_0 slightly by adding 1 to the 14th digit after the decimal point, i.e. change the key from (0.12345678, 3.99999, 0.23456789, 3.99999) to $(0.12345678 + 10^{-14}, 3.99999, 0.23456789, 3.99999)$. Then we compare the two corresponding SOLS and two corresponding cipher-images. The two SOLS are shown partly in Fig. 4(a) and (b), and the two cipher-images of Lena are depicted in Fig. 4(c) and (d). To give quantitative results, we compute the percentage of different pixels between the two cipher-images, which is equal to $\frac{65275}{65536} \approx 99.6017\%$. The simulation results demonstrate the high sensitivity of the encryption key.

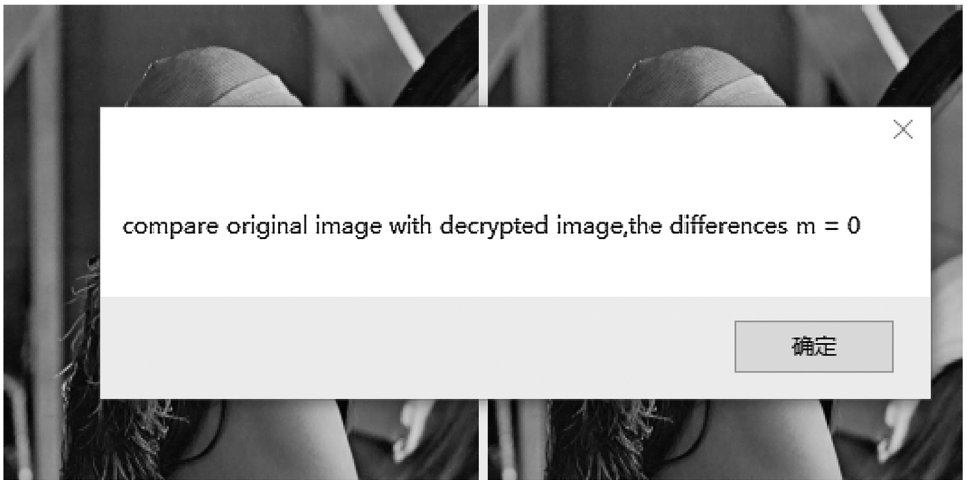

Fig. 5. Comparison of original image and the decrypted image using the correct key.

(2) Key sensitivity analysis in decryption stage

We use $K = (key_0, \mu_0, key_1, \mu_1) = (0.12345678, 3.99999, 0.23456789, 3.99999)$ to encrypt image Lena, and obtain the corresponding ciphertext image E_0. Firstly, we use K to decrypt E_0, the comparison of original image and the decrypted image is shown in Fig. 5. The number of different pixels between the two images is 0. Secondly, we modify key_0 slightly by adding 1 to the 14th digit after the decimal point, i.e. change the secret key from (0.12345678, 3.99999, 0.23456789, 3.99999) to $(0.12345678 + 10^{-14}, 3.99999, 0.23456789, 3.99999)$. Then we decrypt E_0 using the modified key. The comparison of original image and the decrypted image is shown in Fig. 6. The percentage of different pixels between the two images in Fig. 6 is equal to $\frac{65299}{65536} \approx 99.6284\%$. From the simulation results, we can see that a tiny change in the decryption key will result in a totally different decrypted image. It demonstrates the high sensitivity of the decryption key.

4.2. Histogram analysis

The histogram describes the distribution of different pixel values within an image. A uniform histogram is necessary for an ideal ciphertext image to resist histogram analysis. Fig. 7 displays the histograms of some plaintext images and their ciphertext images. From the histograms, we can see that different pixel values are distributed uniformly after the encryption.

4.3. Correlation analysis

As is known, the adjacent pixels in a meaningful image are highly correlated, whereas in a well confused image the adjacent pixels

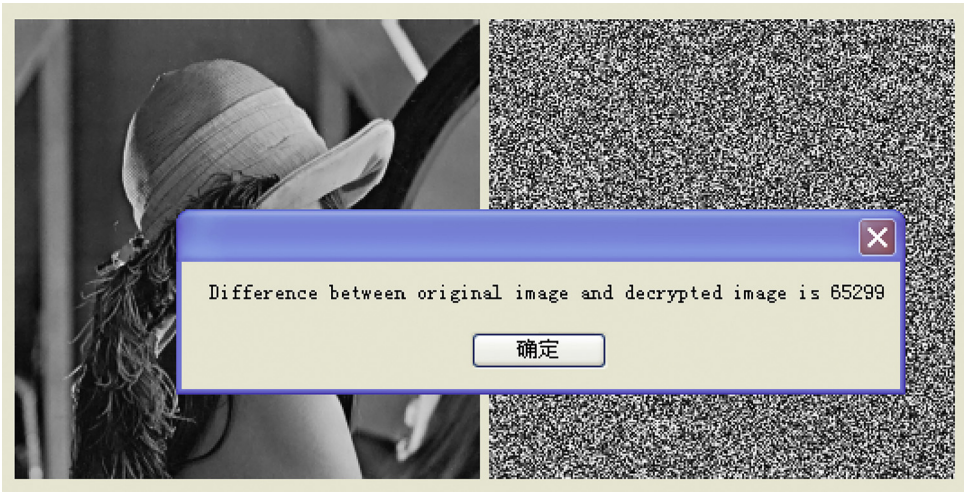

Fig. 6. Comparison of original image and the decrypted image using the modified key.

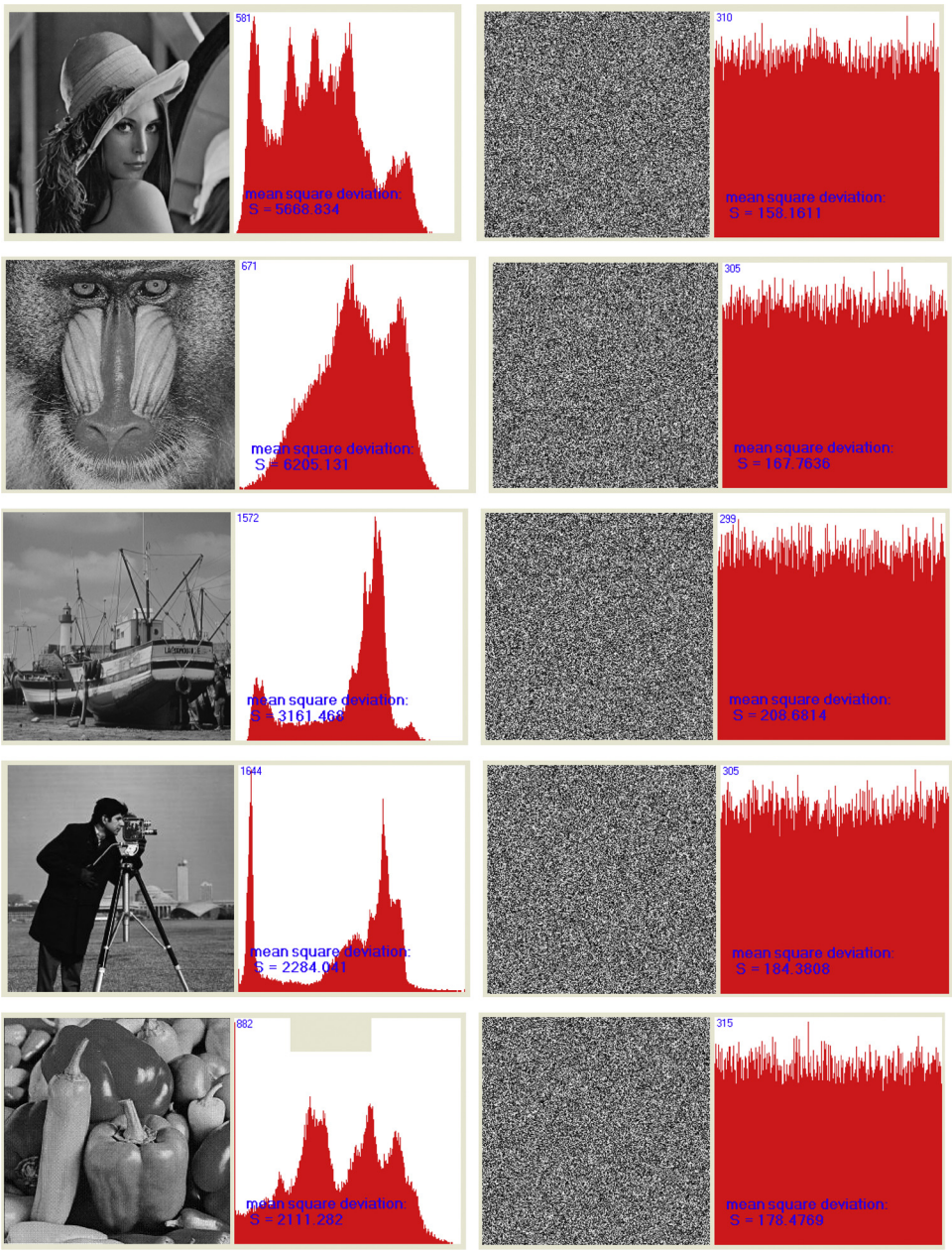

Fig. 7. Sample results of encrypted images using the proposed algorithm. The 1st column: plaintext images(images from top to bottom are Lena, Baboo, Boat, Cameraman, Peppers); 2nd column: histograms of plaintext images; 3rd column: ciphertext images; 4th column: histograms of ciphertext images.

are weakly correlated. To appraise the confusion effect of an encryption algorithm, we compute the correlation coefficients as follows:

$$r_{x,y} = \frac{E\{[x - E(x)][y - E(y)]\}}{\sqrt{D(x)}\sqrt{D(y)}} \tag{12}$$

$$E(x) = \frac{1}{S}\sum_{i=1}^{S} x_i \, D(x) = \frac{1}{S}\sum_{i=1}^{S} [x_i - E(x)]^2 \tag{13}$$

In the above equations, x and y denote two adjacent pixels(vertically adjacent, horizontally adjacent and diagonally adjacent respectively) in the image; $E(x)$ is the expectation of x, and $D(x)$ is the variance; S denotes the total number of samples which are selected randomly in the image. The average coefficient of the three directions(vertical, horizontal and diagonal) is calculated by Eq.

Table 1
Correlation coefficients of different images in the proposed algorithm.

Image	Testing direction			Average value
	Vertical	Horizontal	Diagonal	
Lena	0.9411	0.9652	0.9217	0.9426
Cipher-image of Lena	0.0220	0.01792	7E-06	0.0133
Baboo	0.6916	0.6205	0.6111	0.6411
Cipher-image of Baboo	−0.0062	0.0233	0.0214	0.0169
Boats	0.9140	0.9448	0.8786	0.9125
Cipher-image of Boats	0.02012	0.0117	−0.0182	0.0166
Cameraman	0.9317	0.9591	0.9038	0.9315
Cipher-image of cameraman	−0.0261	−0.0153	−0.0118	0.0177
Peppers	0.8979	0.9271	0.8810	0.9020
Cipher-image of peppers	−0.0127	−0.0103	0.0022	0.0084

(14):

$$averagecoefficient = \frac{|a| + |b| + |c|}{3} \tag{14}$$

where a, b, c denote the correlation coefficients of the three directions respectively.

The correlation coefficients of some plaint-images and their cipher-images in the proposed algorithm are listed in Table 1. To compare the proposed algorithm with others, we also list the correlation coefficients of different encryption schemes in Table 2(plaint-image is Lena). Furthermore, to display the correlation visually, the correlation distributions of different images in the proposed algorithm are plotted in Fig. 8. We can see that in the plaintext images the dots are located along the diagonal, whereas in the ciphertext images the dots are scattered over the entire plane. All the simulation results demonstrate the good confusion effect of the proposed algorithm.

4.4. Information entropy analysis

Information entropy is an index to measure the uncertainty of a cryptosystem. For a message source m, the information entropy $H(m)$ can be computed by Eq.(15).

$$H(m) = \sum_{i=0}^{2^M-1} P(m_i)\log\frac{1}{P(m_i)} \tag{15}$$

where M is the number of bits to represent a symbol $m_i \in m$ and $P(m_i)$ is the occurrence probability of m_i. A secure cryptosystem should produce a cipher-image with equiprobable grey levels. It means that for an ideal cipher-image with 256 grey levels, the information entropy should be close to 8 as much as possible.

Entropy values of the five images in the proposed algorithm are listed in Table 3. Entropy values of Lena in the five different algorithms are listed in Table 4.

From the tables, we can see that the entropy values in the proposed algorithm are very close to the ideal value 8. It means that the proposed algorithm is safe enough against entropy attack.

4.5. Differential attack analysis

By comparing two correlative cipher-images, an attacker may trace the relationship between the plaintext image and its cipher-image. This type of attack is called differential attack. In general, we use two criteria, *NPCR*(Number of Pixels Change Rate) and *UACI* (Unified Average Changing Intensity), to appraise the ability of resisting differential attack. *NPCR* and *UACI* are defined as follows:

Table 2
Correlation coefficients of Lena and its cipher-images in different schemes.

Image	Testing direction			Average value
	Vertical	Horizontal	Diagonal	
Cipher-image in proposed algorithm	0.0220	0.01792	7E-06	0.0133
Cipher-image in [9]	0.0580	0.0024	0.0170	0.0258
Cipher-image in [16]	0.0034	0.0044	0.0020	0.0033
Cipher-image in [18] after 8 rounds	−0.0028	0.0053	0.0017	0.0033
Cipher-image in [26]	0.0025	0.0035	0.0011	0.0024

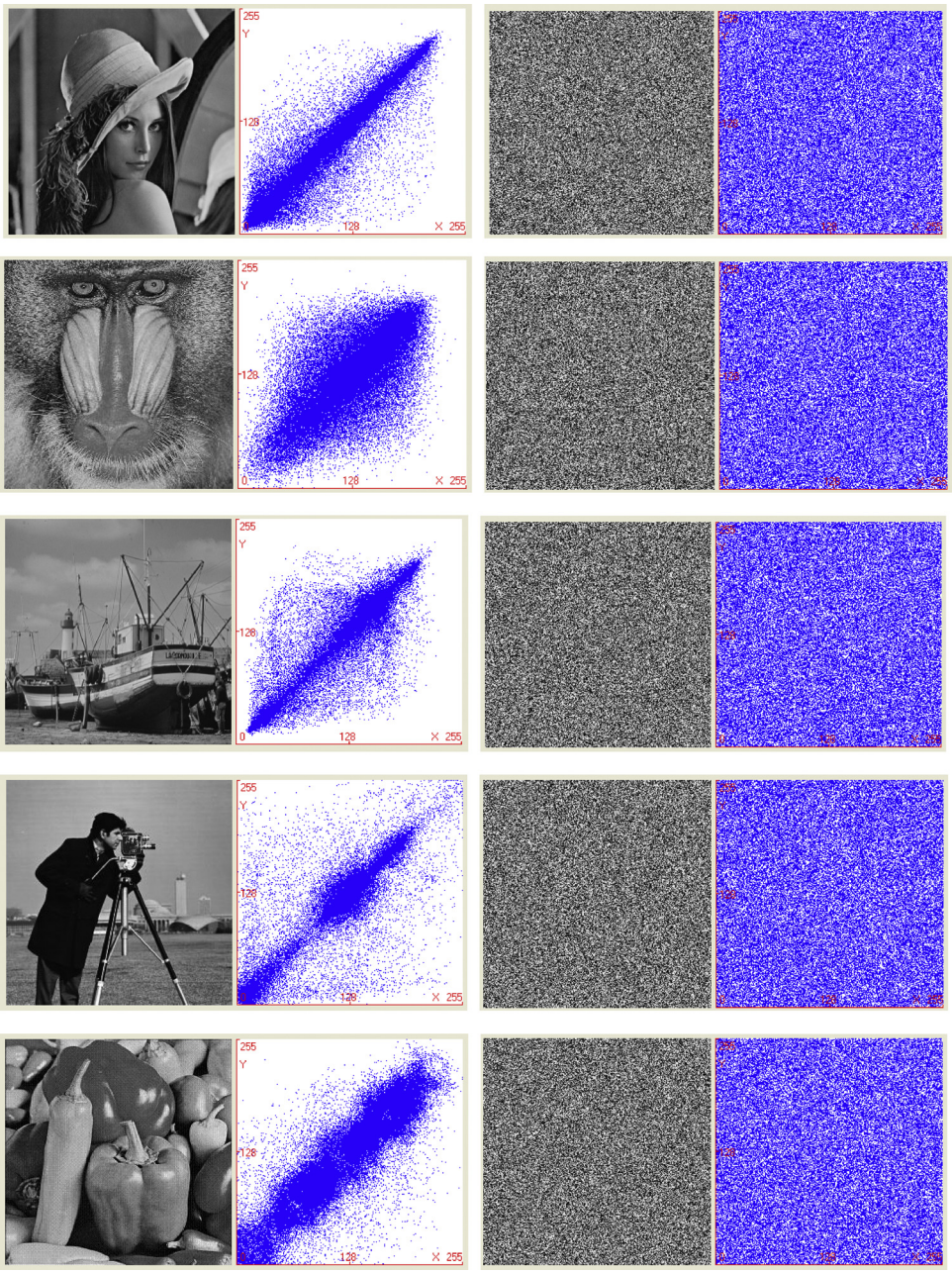

Fig. 8. Correlation distributions of encrypted images using the proposed algorithm. The 1st column: plaintext images(images from top to bottom are Lena, Baboo, Boat, Cameraman, Peppers); 2nd column: correlation distributions of plaintext images; 3rd column: ciphertext images; 4th column: correlation distributions of ciphertext images.

Table 3
Entropy values of different images in the proposed algorithm.

Plain-image	Lena	Baboo	Boats	Cameraman	Peppers
Entropy	7.9970	7.9973	7.9967	7.9970	7.9969

Table 4
Entropy values of Lena in the five different algorithms.

Algorithm	Proposed algorithm	[9]	[16]	[18] 8 rounds encryption	[26]
Entropy	7.9970	7.9871	7.9967	7.9971	7.9973

Table 5
The critical *NPCR* scores for 256 × 256 grey image with 256 levels.

	$\alpha = 0.05$	$\alpha = 0.01$	$\alpha = 0.001$
N_α^*	99.5693%	99.4960%	99.4588%

Table 6
The critical *UACI* scores for 256 × 256 grey image with 256 grey levels.

	$\alpha = 0.05$	$\alpha = 0.01$	$\alpha = 0.001$
u_α^{*-}	33.2824%	33.2255%	33.1594%
u_α^{*+}	33.6447%	33.7016%	33.7677%

Table 7
NPCR and *UACI* values of different images in the proposed algorithm.

Image	NPCR(%)	UACI(%)
Lena	99.6107	33.4232
Baboo	99.6268	33.3149
Boats	99.6013	33.4472
Cameraman	99.5988	33.4025
Peppers	99.6196	33.4150

Table 8
NPCR and *UACI* values of Lena in different algorithms.

Algorithm	NPCR(%)	UACI(%)
Proposed algorithm	99.6107	33.4232
Algorithm [9]	99.6062	33.8981
Algorithm [16]	99.2173	33.4055
Algorithm [18] after 8 rounds	99.6689	33.4936
Algorithm [26]	99.6041	33.4198

Table 9
The average running time of the four different algorithms.

Algorithm	Proposed algorithm	Algorithm in [9]	Algorithm in [16]	Algorithm in [18] after 8 rounds
Time (second)	0.425	≥1	≥10	≥1

$$NPCR = \frac{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} D(i,j)}{M \times N} \times 100\% \tag{16}$$

where $D(i,j) = \begin{cases} 1 & C_1(i,j) \neq C_2(i,j) \\ 0 & C_1(i,j) = C_2(i,j) \end{cases}$

$$UACI = \frac{1}{255 \times M \times N} \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} |C_1(i,j) - C_2(i,j)| \times 100\% \tag{17}$$

In the above formulas, C_1 and C_2 are two cipher-images satisfying that their corresponding plain-images have only one different pixel. M and N are width and height of the image respectively.

Wu et al. in [21] give strict *NPCR* and *UACI* critical values. These critical values are listed in Tables 5 and 6. In Table 5, N_α^* denotes critical *NPCR* score for significance level α. If the obtained *NPCR* value is not smaller than N_α^*, the encryption algorithm is

considered to pass the *NPCR* text successfully. In Table 6, u_α^{*-} and u_α^{*+} denote critical *UACI* scores for significance level α. If the obtained *UACI* value falls into the interval $(u_\alpha^{*-}, u_\alpha^{*-})$, the encryption algorithm is considered to pass the *UACI* test successfully.

The *NPCR* and *UACI* values of five different images in the proposed algorithm are listed in Table 7. The *NPCR* and *UACI* values of Lena in the five different algorithms are listed in Table 8.

From the above tables, we can see that for all the test images, the proposed algorithm can pass both *NPCR* and *UACI* tests with different significance levels.

4.6. Efficiency analysis

Apart from the security issues discussed above, the efficiency of an encryption algorithm is another important issue to consider. In the proposed algorithm, if the image size is $N \times N$, the maximum number of iterations in each stage(SOLS generation, the permutation at top-level, the substitution at middle-level, and the permutation at bottom-level) is N^2. Thus, the computation complexity of the proposed algorithm is $O(N^2)$, which is same to the complexity of algorithms in [9,16,18,26].

In simulations, we compare the actual speed performance of the four different algorithms. For fairness, we encrypt the same image Lena ten times by each algorithm under the same running environment. The average running time of each algorithm is listed in Table 9.The complexity analysis and simulation results show that the proposed algorithm is suitable for real time applications.

5. Conclusion

In this paper, we propose a novel image encryption algorithm based on SOLS. The algorithm adopts “permutation-substitution-permutation” mode in one round encryption, and it uses one SOLS to perform the whole encryption process. The permutation operations at top level and bottom level can protect the substitution operation from differential attack in one round encryption. Besides, the diffusion effect is contributed by both substitution operation and permutation operation at bottom level. Simulation results demonstrate the security and efficiency of the proposed algorithm.

Acknowledgements

The authors wish to thank the editor and the anonymous referees for the constructive comments and suggestions. This work was supported by the National Natural Science Foundation of China [grant number 11471096], Graduate Innovation Project of Hebei Province [grant number CXZZBS2018102], Natural Science Foundation of Hebei Province [grant number A2018210120].

References

[1] A.H. Abdullah, R. Enayatifar, M. Lee, A hybrid genetic algorithm and chaotic function model for image encryption, AEU-Int. J. Electron. Commun. 66 (10) (2012) 806–816.
[2] M. Ahmad, F. Ahmad, Cryptanalysis of Image Encryption Based on Permutation-Substitution Using Chaotic Map and Latin Square Image Cipher 16 Springer International Publishing, 2015, pp. 481–488 (4).
[3] R.K. Brayton, D. Coppersmith, A.J. Hoffman, Self-orthogonal latin squares of all orders n ≠ 2,3,6, Bull. Am. Math. Soc. 80 (1974) 116–118.
[4] X.L. Chai, Z.H. Gan, K. Yang, Y.R. Chen, X.X. Liu, An image encryption algorithm based on the memristive hyperchaotic system, cellular automata and DNA sequence operations, Signal Process.: Image Commun. 52 (2017) 6–19.
[5] G.Q. Hu, D. Xiao, Y.S. Zhang, T. Xiang, An efficient chaotic image cipher with dynamic lookup table driven bit-level permutation strategy, Nonlinear Dyn. 87 (2) (2017) 1359–1375.
[6] A.V. Diaconu, Circular inter-intra pixels bit-level permutation and chaos-based image encryption, Inf. Sci. 355–356 (2016) 314–327.
[7] B. Wang, F.C. Zou, J. Cheng, A memristor-based chaotic system and its application in image encryption, Optik 154 (2018) 538–544.
[9] A. Kadir, A. Hamdulla, W. Guo, Color image encryption using skew tent map and hyper chaotic system of 6th-order CNN, Optik 125 (5) (2014) 1671–1675.
[10] X. Li, L.Y. Wang, Y.F. Yan, P. Liu, An improvement color image encryption algorithm based on DNA operations and real and complex chaotic systems, Optik 127 (5) (2016) 2558–2565.
[11] M.A. Murillo-Escobar, C. Cruz-Hernandez, F. Abundiz-Perez, et al., A RGB image encryption algorithm based on total plain image characteristics and chaos, Signal Process. 109 (2015) 119–131.
[12] H.T. Panduranga, N. Kumar, S.K. Kiran, Image encryption based on permutation-substitution using chaotic map and Latin square image cipher, Eur. Phys. J.-Spec. Top. 223 (8) (2014) 1663–1677.
[13] C. Shannon, Communication theory of secrecy systems, Bell Syst. Tech. J. 28 (4) (1949) 656–715.
[14] H. Zhu, C. Zhao, X. Zhang, L. Yang, An image encryption scheme using generalized Arnold map and affine cipher, Int. J. Light Electron. Opt. 125 (22) (2014) 6672–6677.
[15] X.J. Wu, D.W. Wang, J. Kurths, H.B. Kan, A novel lossless color image encryption scheme using 2D DWT and 6D hyperchaotic system, Inf. Sci. 41 (2) (2016) 137–153.
[16] X. Wei, L. Guo, Q. Zhang, J. Zhang, S. Lian, A novel color image encryption algorithm based on DNA sequence operation and hyper-chaotic system, J. Syst. Softw. 85 (2) (2012) 290–299.
[17] R. Enayatifar, A.H. Abdullah, I.F. Isnin, Chaos-based image encryption using a hybrid genetic algorithm and a DNA sequence, Opt. Lasers Eng. 56 (2014) 83–93.
[18] Y. Wu, Y.C. Zhou, J.P. Noonan, S. Agaian, Design of image cipher using Latin squares, Inf. Sci. 264 (2014) 317–339.
[19] Q. Zhang, L. Guo, X. Wei, A novel image fusion encryption algorithm based on DNA sequence operation and hyper-chaotic system, Optik 124 (2013) 3596–3600.
[20] X.P. Zhang, Z.M. Zhao, J.Y. Wang, Chaotic image encryption based on circular substitution box and key stream buffer, Signal Process.: Image Commun. 29 (8) (2014) 902–913.
[21] Y. Wu, J.P. Noonan, S. Agaian, NPCR and UACI randomness tests for image encryption, Cyber J.: Multidiscip. J. Sci. Technol. JSAT 4 (2011) 31–38.
[22] Y.S. Zhang, D. Xiao, Y.L. Shu, J. Li, A novel image encryption scheme based on a linear hyperbolic chaotic system of partial differential equations, Signal Process.: Image Commun. 28 (2013) 292–300.
[23] G.R. Chen, Y.B. Mao, C.K. Chui, A symmetric image encryption scheme based on 3D chaotic cat maps, Chaos Solitons Fractals 21 (2004) 749–761.
[24] J.X. Chen, Z.L. Zhu, C. Fu, H. Yu, Y.S. Zhang, Reusing the permutation matrix dynamically for efficient image cryptographic algorithm, Signal Process. 111 (2015) 294–307.

[25] Y.S. Zhang, D. Xiao, Self-adaptive permutation and combined global diffusion for chaotic color image encryption, Int. J. Electron. Commun. 68 (2014) 361–368.
[26] L.Y. Zhang, X.B. Hu, Y.S. Liu, K.W. Wong, J. Gan, A chaotic image encryption scheme owning temp-value feedback, Commun. Nonlinear Sci. Number Simulat. 19 (2014) 3653–3659.
[27] T. Xie, Y.S. Liu, J. Tang, Breaking a novel image fusion encryption algorithm based on DNA sequence operation and hyper-chaotic system, Optik 125 (2014) 7166–7169.
[28] Y.S. Liu, L.Y. Zhang, J. Wang, Y.S. Zhang, K.W. Wong, Chosen-plaintext attack of an image encryption scheme based on modified permutation-diffusion structure, Nonlinear Dyn. 84 (2016) 2241–2250.
[29] L.Y. Zhang, Y.S. Liu, C. Wang, J.T. Zhou, Y.S. Zhang, G.R. Chen, Improved known-plaintext attack to permutation-only multimedia ciphers, Inf. Sci. 430–431 (2018) 228–239.
[30] J. Fridrich, Symmetric ciphers based on two-dimensional chaotic maps, Int. J. Bifurc. Chaos 8 (6) (1998) 1259–1284.