Signal Processing 148 (2018) 124–144

Contents lists available at ScienceDirect

Signal Processing

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

An image encryption algorithm based on chaotic system and compressive sensing

Xiuli Chai[a,*], Xiaoyu Zheng[a], Zhihua Gan[b], Daojun Han[a], Yiran Chen[c]

[a] *School of Computer and Information Engineering, Institute of Collaborative Intelligent Transportation System (ITS), Henan University, Kaifeng 475004, China*
[b] *School of Software, Henan University, Kaifeng 475004, China*
[c] *Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, United States*

ARTICLE INFO

Article history:
Received 14 November 2017
Revised 2 February 2018
Accepted 5 February 2018
Available online 6 February 2018

Keywords:
Image encryption
Compressive sensing (CS)
Elementary cellular automata (ECA)
Memristive chaotic system

ABSTRACT

In this paper, we propose an image encryption algorithm based on the memristive chaotic system, elementary cellular automata (ECA) and compressive sensing (CS). Firstly, the original image is transformed by discrete wavelet transform, and the sparse coefficient matrix is obtained. Next, a zigzag scrambling method and the ECA are adopted to scramble the sparse coefficient matrix successively, and this process may effectively improve the scrambling degree. And then, the measurement matrix produced by the memristive chaotic system is used to compress and perceive the scrambled image, and the final cipher image is obtained. In addition, SHA-512 hash function value of the original image is generated to calculate the parameters for zigzag confusion, the initial values of the chaotic system and the initial configurations of the ECA, which enhances the correlation between the algorithm and the plain image and makes the proposed encryption scheme resist the known-plaintext and chosen-plaintext attacks. Moreover, our algorithm can compress and encrypt the image simultaneously by use of CS, which may reduce the amount of data and storage space. Simulation results and performance analyses demonstrate the security and robustness of the proposed scheme.

1. Introduction

With the advent of the Internet era, the vast majority of information in our lives cannot be separated from the support of the Internet. We use it for video conferencing, sending some information and so on. Some image information may be involved in personal privacy, trade secrets, military secrets and even national security, thus it will be very serious that attackers copy, malicious spread and tamper with the images in the transmission process through the network [1–3]. Therefore, in order to protect the image information over the network, many image encryption algorithms have been presented by use of optical transformations [4,5], DNA computing [6–9], Arnold transform [10,11], Latin squares [12], bit-level permutation [13–15] and other methods. These algorithms can encrypt image information effectively and ensure data security.

Recently, with the arrival of big data era, the volume of information is constantly increasing, the amount of data that needs to be transmitted is generally larger and the information redundancy is high. In order to reduce the amount of data transmitted through the network, the image needs to be compressed and then entered into transmission channel. At present, it has become a hotspot of information security research to encrypt images with compressive sensing (CS) and other encryption methods, which has great application potential and high practical value. The theory of CS points out that: by developing the sparse characteristic of the signal, the discrete sample of the signal is obtained by random sampling under the condition of far less than the Nyquist sampling rate, and then the reconstruction signal is perfect by the nonlinear reconstruction algorithm. In 2006, Candes and Donoho formally proposed the concept of CS [16,17], and after that many compression and encryption algorithms have also been presented based on CS [18–21].

The chaotic system has the characteristics of high sensitivity to initial conditions and control parameters, and is widely used in the field of image encryption to further enhance the randomness of the algorithm and keys [22–25]. Currently, the chaotic map can be divided into two categories: one-dimensional (1D) and high-dimensional (HD) chaotic map. In recent years, researchers have combined the 1D chaotic map and CS to design many image encryption algorithms [18–20,26–29]. For example, Zhou et al. [27] used the logistic chaotic map to generate two measurement matrices for CS, next the original image was compressed and en-

[*] Corresponding author.
E-mail address: chaixiuli@vip.henu.edu.cn (X. Chai).

https://doi.org/10.1016/j.sigpro.2018.02.007
0165-1684/© 2018 Elsevier B.V. All rights reserved.

crypted simultaneously from two directions, and then re-encrypted the obtained image by non-linear fractional Merlin transform to obtain the final cipher image. Xiao et al. [28] adopted Arnold transform to scramble the transform domain coefficients of the original image, and then the watermark was adhered to the scrambled data. By CS, a set of watermarked measurements was obtained as the watermarked cipher image. In his algorithm, the measurement matrices were controlled by the chaotic map. In contrast, the HD chaotic map, especially the hyper-chaotic map, has more variables and parameters, with more complex structure and better chaotic performance [30]. Hence, some researchers put forward some image encryption algorithms based on HD chaotic map and CS [30–34]. For instance, Zhou et al. [31] firstly compressed and encrypted the plain image, then the resulting image was re-encrypted by the cyclic shift operation controlled by a hyper-chaotic system, the cyclic shift operation can effectively change the values of the pixel. Tong et al. [32] used the hyper-chaotic system to obtain three keys, and then utilized them to encrypt and compress the color image.

Cellular automata (CA) can produce complex and random patterns out of simple rules, and it has been widely used in encryption field [35–38]. Many image encryption algorithms used CA in diffusion phase, which enhances the security of cryptographic system. Such as, in Ref. [35], two-dimensional reversible memory cellular automata was associated with quadtree decomposition strategy and applied to the diffusion process. In addition, CA may be used in the confusion process. For example, Abdel et al. presented two encryption algorithms based on CA, one using elementary cellular automata (ECA) [36], and the other using two-dimensional CA [37], and these algorithms both achieved a good scrambling effect. However, the above proposed algorithms had small key spaces; what's more, the cipher image has the same size with the plain image, and the image redundancy information was not reduced. In recent years, researchers have combined cellular automata with compressive sensing [39,40]. Among them, Chen et al. [39] adopted ECA to scramble the sparsely transformed image. And in the second stage of encryption, Kronecker compressive sensing (KCS) was adopted to encrypt and compress the scrambled image. The experimental results showed that the proposed scrambling method based on ECA had great performance in terms of scrambling and uniformity of sparsity levels. But the algorithm has low correlation with the plain image and it is easy to attack by known-plaintext and chosen-plaintext attacks. Thus, increasing the correlation between the encryption scheme and the plain image is necessary to upgrade the security level of the encryption algorithm.

Based on the above analyses, in the premise of guaranteeing information security, we introduce an image encryption algorithm based on the memristive chaotic system, elementary cellular automata and compressive sensing. Our contributions are as follows. First of all, the original image can be effectively encrypted and the image information is protected. In the scrambling phase, the zigzag path and ECA are used. Using ECA, we can obtain complex global activities through simple cellular rules to realize fast scrambling process. It has good scrambling effect and high security. Secondly, after the plain image is scrambled, CS is utilized to compress and encrypt the confused image to minimize data and save the transmission time over the network. And the adoption of random zigzag and ECA confusion scheme in scrambling phase can effectively enhance the compression performance. Moreover, the SHA-512 hash function value of the original image is used to compute the parameters for encryption, thus, the proposed algorithm is highly sensitive to the plain image. In addition, a new kind of magnetic-controlled memristive chaotic system is applied to produce the measurement matrix for CS, and the measurement matrix may be obtained by some parameters. When the receivers get them, they can easily recover the plain image. So from this point, our algorithm is easy to manipulate in real condition.

The rest of the paper is organized as follows. In Section 2, some related knowledge is given. In Section 3, the proposed encryption and decryption scheme is described. Numerical simulations are presented in Section 4. Performance analyses of the proposed algorithm are demonstrated in Section 5. And the last section concludes our work.

2. Preliminaries

2.1. The magnetic-controlled memristive chaotic system

In recent years, the general chaotic system has been developed more comprehensively. Besides the classical chaotic systems such as Chen system and Lorenz system, there are some improved systems with more superior non-linear characteristics. Among them, Min et al. [41] proposed a new magnetic-controlled memristive chaotic system. They analyzed the relationship between voltage and current of a new type of magnetic-controlled memristor model based on hyperbolic sine function, found its typical characteristic of memristor, and then proposed the new chaotic system. The memristive chaotic system is defined as

$$\begin{cases} \dot{x} = -ax + by + yz \\ \dot{y} = -0.5xz + cx - dxW(\varphi) \\ \dot{z} = 0.8xy - rz \\ \dot{w} = gx \end{cases} \tag{1}$$

where $W(\varphi) = \cosh(0.02w)$, x, y, z, w represent the state variables of the chaotic system, a, b, c, d, g, r are the control parameters of the chaotic system and are real constant, when $a = 10$, $b = 8$, $c = 15$, $d = 5.2$, $g = 5$ and $r \in (1.21, 3.14)$, $r \in (2.96, 4.05)$ or $r \in (4.25, 5.82)$, the system is in a state of chaos, in particular, when $r = 1.5$, the system has a typical chaotic attractor, as shown in Fig. 1.

The experiment result shows that the memristive chaotic system has strong aperiodicity, strong sensitivity to initial values, large parameter space, and can be easily realized in physics [41]. Applying it to image encryption, some secure encryption algorithms can be designed.

2.2. Cellular automata and elementary cellular automata

Cellular automata (CA) are a kind of discrete dynamical systems [35]. In general, a CA consists of a certain number of identical cells, each of which can take a finite number of states. A state is the value of a cell in a discrete time step, especially the first state is called the initial state (the initial configuration) [39], usually we will write it as {0, 1}. The cells are distributed in space in one or more dimensions. At every time step, all cells update their states synchronously by applying rules (also called transition function), and we call this process an evolution [36]. Evolution is the process of transferring the state of all cells to the next state according to the rules [39]. At the time of evolution, the state of a cell and the state of its neighbor used as input determine its next state, where the rule defines the deterministic way to update the synchronization state of all cells. In theory, the cellular space can be extended indefinitely, but in the actual process, it is difficult to realize this ideal condition on the computer, so we need to define different boundary conditions. The existing boundary conditions are mainly three types: periodic type, reflective type and fixed value type. The periodic boundary means that the left (right) neighborhood of the leftmost (rightmost) cell is the rightmost (leftmost) cell, as shown in Fig. 2.

Since there are a lot of parameters that need to be determined, CA differ in dimension, possible states, neighborhood relationships and rules. CA has many types and these types differ in terms of

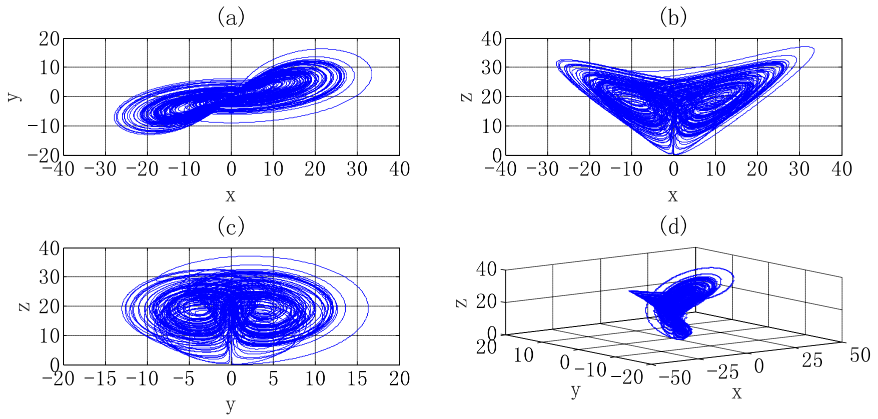


Fig. 1. Chaotic attractor. (a) (*x-y*) plane, (b) (*x-z*) plane, (c) (*y-z*) plane, (d) (*x-y-z*) plane.

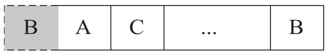


Fig. 2. Periodic boundary conditions.

Table 1
The map of the rule 170.

111	110	101	100	011	010	001	000
1	0	1	0	1	0	1	0

complexity and behavior. Elementary cellular automata (ECA) are the simplest class of one-dimensional CA. Its state set S has only two elements, that is, the number of states $k=2$, and the neighbor radius $r=1$. Therefore, the evolution of ECA can be described entirely by each cell and its neighboring cells [42], i.e., states of a cell and its two neighbor cells (left neighborhood and right neighborhood) determine the next state of the cell and it may be shown as

$$state^{r+1}(i) = f(state^{r}(i-1), state^{r}(i), state^{r}(i+1)) \quad (2)$$

where $f(\bullet)$ represents map rule, and $state^{r}(i)$ denotes the state of the ith cell for the rth round evolution [39]. It has two possible values for each cell (usually 0 or 1) and the rules only depend on the values of the nearest neighbors. Consequently, the evolution of an elementary cellular automaton can completely be described by a table of the possible combination of each cell and its neighbors (8 possible states). The ECA are usually referenced based on this table as there are only 256 elementary cellular automata (2^8), each of which can be indexed with an 8-bit binary number. For example, the rule 170 (binary: 10,101,010) is that $state^{r+1}(i)=0$ for $state^{r}(i-1), state^{r}(i), state^{r}(i+1) \in \{110, 100, 010, 000\}$ and $state^{r+1}(i)=1$ for $state^{r}(i-1), state^{r}(i), state^{r}(i+1) \in \{111, 101, 011, 001\}$ as shown in Table 1.

2.3. Compressive sensing (CS)

Compressive sensing theory asserts that if the signal is naturally sparse or sparse in some transform domains, the high dimensional signal can be projected into a low dimensional space by a measurement matrix unrelated to the sparse base, and these few projections contain enough information about the reconstructed signal, so that the original signal can be reconstructed with high probability by solving the optimization problem with these projections [43].

Suppose that a signal x of length N is K-sparse and it can be expressed as

$$x=\Psi s \quad (3)$$

where s is the transform coefficient vector that contains at most K ($K << N$) important nonzero entries, Ψ is an orthogonal transform matrix (also called sparse basis matrix). A measurement matrix Φ of length $M \times N$ ($M < N$), which is unrelated to the Ψ, is adopted to perform a compression measurement of the signal and shown as

$$y=\Phi x \quad (4)$$

It is possible to obtain M linear observation y, which contains enough information about the reconstructed signal x. Based on Eqs. (3) and (4), the whole process can be represented as

$$y=\Phi x=\Phi\Psi s=\Theta s \quad (5)$$

Then x can be recovered by solving the optimization problem. The optimization problem can be regarded as the solution of p-norm problem, which is based on 1-norm optimization problem, and it can be denoted as

$$\min \|s\|_1 \text{ s.t. } y=\Theta s \quad (6)$$

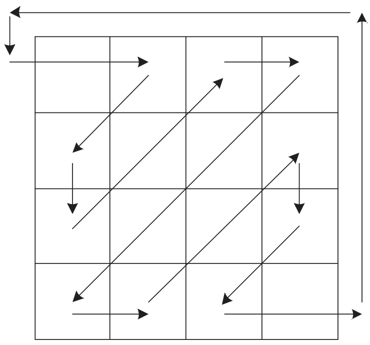
The K-sparse signal s can be accurately recovered, and the original signal can be reconstructed exactly as long as $M \geq K\log_2(\frac{N}{K})$ [18].

The basic model of CS theory mainly includes three major aspects: the sparse representation of signals, compression measurement and signal reconstruction [44]. Common sparse representations include: curvelet transform, discrete cosine transform (DCT) and discrete wavelet transform (DWT) [21], etc. Signal reconstruction is the process of accurately reconstructing the original signal or high-dimensional image by using the low dimensional data of the compression measurement [44], common reconstruction algorithms are: orthogonal matching pursuit (OMP) algorithm, subspace pursuit (SP) algorithm and smooth l_0 norm (SL$_0$) algorithm [29], and here, the SL$_0$ algorithm will be adopted. In the compression measurement, the linear projection of the signal needs a measurement matrix unrelated to the sparse transformation matrix. At present, according to the restricted isometry property (RIP) condition of the measurement matrix, some scholars have proposed a number of measurement matrices, such as Gaussian random matrix, partial orthogonal matrix, Hadamard matrix and circular matrix [44].

In this paper, the measurement matrix Φ is constructed as a circular matrix, and each row of the circular matrix is produced by its previous row moving to the right. Therefore, the circular matrix requires less independent variables, fast computation speed and easy hardware implementation, and has good performance [45]. And, the original row vector of the circular matrices is controlled by the memristive chaotic map.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

(a) The original matrix A



(b) A zigzag path for confusion

1	2	5	9	6	3	4	7	10	13	14	11	8	12	15	16

(c) The matrix after zigzag confusion with starting pixel (1, 1)

6	3	4	7	10	13	14	11	8	12	15	16	1	2	5	9

(d) The matrix after zigzag confusion with starting pixel (2, 3)

Fig. 3. A zigzag path for confusion.

The measurement matrix Φ with the size of $m \times N$ is constructed as follows: Suppose that the original row vector $\Phi(1, N)$= U, and U is a $1 \times N$ vector generated by the memristive chaotic map. To reduce the relevance among the column vectors, the first element of the vector $\Phi(j, 1)$ is set as $\lambda\Phi(j-1, N)$, where $2 \le j \le$ m, $\lambda > 1$, and then Φ may be generated as [29]

$$\begin{cases} \Phi(j,1) = \lambda \cdot \Phi(j-1,N) \\ \Phi(j,2:N) = \Phi(j-1,1:N-1) \end{cases} \tag{7}$$

The following conclusions can be drawn from the generation process of the measurement matrix: first of all, some key parameters can be used to generate the measurement matrix for compressive sensing. In real-time communication networks, we just need to transfer these parameters instead of the entire matrix. The amount of data transferred is largely reduced. Secondly, measurement matrix is produced by the memristive chaotic system, which is highly sensitive to the parameters and initial values. Once the parameters are changed, we can get different measurement matrices, and further obtain different compression results.

2.4. Zigzag confusion

A zigzag path shown in Fig. 3(b) is used to confuse the sparse coefficient matrix of the plain image, and it can disturb the high correlation among image pixels to increase the security level of the encryption algorithm [46]. For zigzag confusion, location of the starting pixel in the matrix is very important, different locations can generate different confusion effects. For instance, for the original matrix as shown in Fig. 3(a), if the location of the starting pixel is (1, 1), that means we start traversing the path from the first pixel, and the matrix after zigzag confusion is shown in Fig. 3(c). If the location is (3, 4), which means that we start traversing the path of the matrix from pixel number $3 \times 4 = 12$, and the matrix after zigzag confusion is illustrated in Fig. 3(d), and in Fig. 3(d), the 12th pixel is 6. In this paper, in order to improve the dependence of the encryption algorithm on the original image, we calculate the starting location of zigzag confusion according to the SHA 512 hash value of the plain image, and then different plain images have different confusion effects, which can upgrade the capability of the proposed encryption algorithm to resist known-plaintext and chosen-plaintext attacks [46].

3. The proposed encryption and decryption scheme

3.1. The generation of initial values of the chaotic system

In the paper, in order to increase the relationship between the encryption scheme and the plain image, the SHA-512 hash function value of the original image is utilized to compute the parameters for encryption. Before encrypting the plain image, its 512-bit hash value is calculated as the secret key K, then it is divided into 8-bit blocks, so it can be converted to 64 decimal digits k_1, k_2, ..., k_{64}. Then the initial values of the memristive chaotic system x_0, y_0, z_0 and w_0 may be calculated. The specific steps are as follows:

Step 1: Regard every 8 bits as a set, convert 512-bit secret key K to 64 decimal numbers k_1, k_2, ..., k_{64}, and then get h_1, h_2, h_3, h_4 via

$$\begin{cases} h_1 = \frac{1}{256}(k_1 \oplus k_2 \oplus k_3 \oplus \ldots\ldots \oplus k_{16}) + t_2 \\ h_2 = \frac{1}{256}(k_{17} \oplus k_{18} \oplus k_{19} \oplus \ldots\ldots \oplus k_{32}) \times t_3 \\ h_3 = \frac{1}{256}(k_{33} \oplus k_{34} \oplus k_{35} \oplus \ldots\ldots \oplus k_{48}) \\ h_4 = \frac{\mathrm{sum}(k_{49}, k_{50}, \ldots, k_{64})}{\max(k_{49}, k_{50}, \ldots, k_{64})} \times t_4 \end{cases} \tag{8}$$

where sum(k_{49}, k_{50}, ..., k_{64}) denotes the sum of k_{49}, k_{50}, ..., k_{64}, max(k_{49}, k_{50}, ..., k_{64}) means the maximum value of k_{49}, k_{50}, ..., k_{64}, $x \oplus y$ is the XOR operation of x and y, t_2, t_3, t_4 are part of secret keys.

Step 2: h_1, h_2, h_3, h_4 are utilized to compute the initial values (x_0, y_0, z_0 and w_0) of the chaotic system by

$$\begin{cases} x_0 = \mathrm{abs}(h_1) - \mathrm{floor}(h_1) \\ y_0 = \mathrm{abs}(h_2) - \mathrm{floor}(h_2) \\ z_0 = \mathrm{mod}(h_3 + h_4, 1) \\ w_0 = \mathrm{mod}\left(\frac{h_4}{7}, 1\right) \end{cases} \tag{9}$$

where abs(x) is the absolute value of x, mod(a, b) returns the remainder of a divided by b. And floor(x) represents the largest integer that is smaller than x.

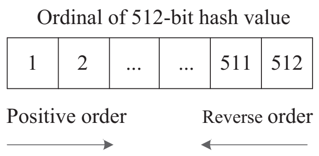


Fig. 4. The order for the initial configurations selection of ECA.

3.2. The computation of the initial position of zigzag confusion

The location (x_0', y_0') of the starting pixel for zigzag confusion is computed by

$$\begin{cases} x'_0 = (h_1 + h_2) \times 10^{15} \bmod N \\ y'_0 = (h_3 \times t_5 + h_4) \times 10^{15} \bmod N \end{cases} \tag{10}$$

where h_1, h_2, h_3 and h_4 are four parameters gotten in Section 3.1, t_5 is part of secret keys, and N is the size of the plain image.

The computed x_0' and y_0' may be zero. In order to remove this problem, we need to modify x_0' and y_0', and the final initial location (x_0', y_0') is obtained by

$$\begin{cases} x_0' = x_0', x_0' \neq 0 \\ x_0' = \lfloor N/3 \rfloor + 2, x_0' = 0 \end{cases} \tag{11}$$

and

$$\begin{cases} y_0' = y_0', y_0' \neq 0 \\ y_0' = \lfloor N/6 \rfloor + 2, y_0' = 0 \end{cases} \tag{12}$$

where $\lfloor x \rfloor$ represents the largest integer that is smaller than x, and N is the size of the plain image.

3.3. The generation of the initial configurations of elementary cellular automata

From the 512-bit hash value of the image, we can get two sequence strings of length N. They are used as the initial configurations C_0^{row} and C_0^{col} of the elementary cellular automata (ECA), and they are also utilized as the row and column coordinates of the scrambling. The methods of producing C_0^{row} and C_0^{col} are as follows:

Step 1: The method of selecting C_0^{row} is as follows: when $1 \leq N \leq 512$, select N values as the initial configuration C_0^{row} from the 512-bit hash value in the direction of reverse order shown in Fig. 4 (i.e., from the back to the front). When $512 \langle N \leq 1024$, firstly choose 512 values in the direction of reverse order, then choose (N-512) values in the direction of positive order. When $N \rangle$ 1024, firstly, pick up values in reverse order, then positive order, and then reverse order, and so on, until you select a sequence with length N as shown in Fig. 5.

Step 2: The method of selecting C_0^{col} is as follows: when $1 \leq N \leq 512$, select N values as the initial configuration C_0^{col} from the 512-bit hash value in the direction of positive order illustrated in Fig. 4 (i.e., from the front to the back). When $512 \langle N \leq 1024$, firstly choose 512 values in the direction of positive order, then choose (N-512) values in the direction of reverse order. When $N \rangle$ 1024, firstly select the values in positive order, then reverse order, and then positive order, and so on, until you get a sequence with length N, as illustrated in Fig. 5.

Finally, we can obtain two initial configurations C_0^{row} and C_0^{col}. C_0^{row} is the initial row configuration, while C_0^{col} is the initial column configuration. In the encryption and decryption process, each evolution result of C_0^{row} and C_0^{col} of ECA represents the row and column coordinate value of the matrix, respectively.

3.4. The generation of measurement matrix for CS

In this paper, the measurement matrix Φ used in compressive sensing (CS) is constructed as a circular matrix, and each row of the circular matrix is produced by its previous row moving to the right. Assuming the size of the measurement matrix is $m \times N$, the construction steps may be described as follows:

Step 1: Use x_0, y_0, z_0 and w_0 produced in Section 3.1 to iterate the memristive chaotic system ($n_0 + N$) times (Here, $n_0 \geq 500$ and it is part of secret keys). In order to eliminate transient effect of the chaotic sequence and enhance their sensitivity to initial conditions, we get rid of the first n_0 group value of the chaotic sequence, and then obtain the sequence X, Y, Z and W, and X = [x_1, x_2, ..., x_N], Y = [y_1, y_2, ..., y_N], Z = [z_1, z_2, ..., z_N], W = [w_1, w_2, ..., w_N].

Step 2: Three matrices X_1, Y_1 and Z_1 are gotten through modifying the elements of the X, Y and Z by

$$\begin{cases} \mathrm{X\_1}(i) = \mathrm{mod}((\mathrm{abs}(x_i) - \mathrm{floor}(x_i)) \times 10^8, px) \\ \mathrm{Y\_1}(i) = \mathrm{mod}((\mathrm{abs}(y_i) - \mathrm{floor}(y_i)) \times 10^8, py) \\ Z\_1(i) = \mathrm{mod}((\mathrm{abs}(z_i) - \mathrm{floor}(z_i)) \times 10^8, pz) \end{cases} \tag{13}$$

where x_i, y_i and z_i are the corresponding elements of X, Y and Z, respectively. X_1(i), Y_1(i) and Z_1(i) are the ith element of X_1, Y_1 and Z_1, respectively, $i = 1, 2, \ldots, N$. px, py and pz are three respective parameters to modifying x_i, y_i and z_i such that the resulting values X_1(i), Y_1(i) and Z_1(i) have stronger randomness and the proposed encryption has higher security level, and $px \in (0, 1]$, $py \in (0, 1]$, $pz \in (0, 1]$. In this paper, we set $px = 0.5$, $py = 0.7$ and $pz = 1$.

Step 3: According to Eq. (14), use sequence X_1, Y_1 and Z_1 to get three sequences U_1, U_2 and U_3 with size of $1 \times N$.

$$\begin{cases} \mathrm{U}_1(i) = [\mathrm{X\_1}(i) + \mathrm{Y\_1}(i) - \mathrm{Z\_1}(i)] \\ \mathrm{U}_2(i) = [\mathrm{X\_1}(i) + \mathrm{Z\_1}(i) - \mathrm{Y\_1}(i)] \\ \mathrm{U}_3(i) = [\mathrm{Y\_1}(i) + \mathrm{Z\_1}(i) - \mathrm{X\_1}(i)] \end{cases} \tag{14}$$

where X_1(i), Y_1(i), Z_1(i), $U_1(i)$, $U_2(i)$ and $U_3(i)$ are the ith element of X_1, Y_1, Z_1, U_1, U_2 and U_3, respectively, and $i = 1, 2, \ldots, N$.

Step 4: Calculate the variance var of the plain image and modify it according to the following equations, and then $var1$ is gotten, $var1 = 1, 2, 3$.

$$Mean = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathrm{I}(i, j) \tag{15}$$

$$var = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\mathrm{I}(i, j) - Mean)^2 \tag{16}$$

and

$$var1 = \mathrm{floor}\left(\mathrm{mod}\left(var \times 10^3, 3\right)\right) + 1 \tag{17}$$

where I(i, j) is the gray value of the pixel located at (i, j) of plain image I, N is the size of the plain image.

Step 5: Choose one sequence from the three sequences U_1, U_2 and U_3 according to $var1$, and then modify the selected sequence. After that we may obtain one sequence U with size of $1 \times N$ by the following rules

if $var1 = 1$, then U(i) = mod($U_1(i)$, 1);
if $var1 = 2$, then U(i) = mod($U_2(i)$, 1);
if $var1 = 3$, then U(i) = mod($U_3(i)$, 1);

where U(i), $U_1(i)$, $U_2(i)$ and $U_3(i)$ are the ith element of U, U_1, U_2 and U_3, respectively, and $i = 1, 2, \ldots, N$.

Step 6: Use sequence U to construct measurement matrix Φ with size of $m \times N$. And here the original row vector is $\Phi(1, N) = \mathrm{U}$, as described in Section 2.3, Φ may be obtained according to Eq. (7).

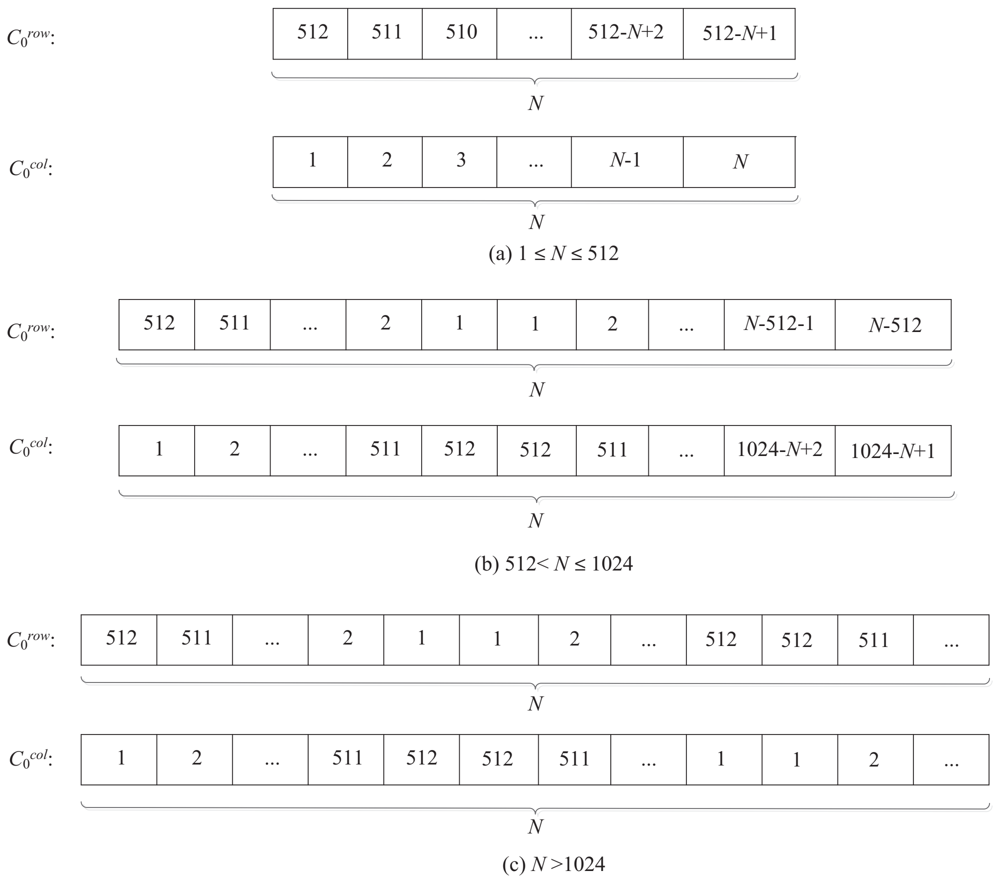


Fig. 5. The method for generating the initial configurations of ECA.

3.5. The proposed encryption algorithm

The proposed image encryption scheme is illustrated in Fig. 6, and the detailed encryption steps are as follows:

Step 1: Assume the size of the plain image I is $N \times N$, then it is sparsified by use of discrete wavelet transform (DWT), and the sparse coefficient matrix I_1 with the same size of $N \times N$ is obtained.

Step 2: Calculate the initial values x_0, y_0, z_0 and w_0 of the memristive chaotic system as described in Section 3.1.

Step 3: Get the initial configurations C_0^{row} and C_0^{col} of the ECA as shown in Section 3.3.

Step 4: Perform zigzag confusion on I_1 with (x_0', y_0') as discussed in Section 2.4, and the 1-D vector p_1 $(1 \times N^2)$ is gotten. Where, the initial location (x_0', y_0') is obtained as illustrated in Section 3.2.

Step 5: Select a certain rule of ECA to evolve the initial row configuration C_0^{row} and initial column configuration C_0^{col} e times, respectively. And here, we adopt periodic boundaries and the radius is set as 1. Then, we can get each evolution result of the row configuration:$C_1^{row}, C_2^{row}, \ldots, C_d^{row}, \ldots, C_e^{row}$ and column configuration: $C_1^{col}, C_2^{col}, \ldots, C_d^{col}, \ldots, C_e^{col}$, where $d = 1, 2, \ldots, e$ and the total number e of evolution of ECA is a part of secret keys.

Step 6: Firstly, construct an $N \times N$ blank matrix I_2, then perform a w round scrambling operation on p_1 $(1 \times N^2)$. In each round of scrambling, the elements in the vector p_1 are filled into the blank matrix I_2 according to the value of $(C_d^{row}(i), C_d^{col}(j))$, the matrix I_3 $(N \times N)$ is obtained after a round of scrambling. After that, convert I_3 to 1-D vector p_1 $(1 \times N^2)$ by column precedence, and continue to perform the next round operation, finally carry on w round scrambling and the final permutated matrix I_4 is gotten. And here, $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, N$, $d = 1, 2, \ldots, e$, C_d^{row} and C_d^{col} denote the dth round results of the evolution of ECA, and e is the total number of evolution of ECA, w is the total number of scrambling rounds. What's more, e and w are part of secret keys. The detailed procedures may be described as follows.

Step 6.1: Set $s = 1$, where, s is the number of scrambling rounds.

Step 6.2: Construct an $N \times N$ blank matrix I_2 and set $d = 1$, d is the number of evolution of ECA.

Step 6.3: Then, the elements in p_1 $(1 \times N^2)$ are sequentially inserted into the matrix I_2 in row-major. The insertion coordinates are (1, 1), that is $(C_d^{row}(i), C_d^{col}(j)) = (1, 1)$, until all the coordinates $(C_d^{row}(i), C_d^{col}(j)) = (1, 1)$ in p_1 are filled with elements. In particular, if there is an element at the position of the coordinate (1, 1) in I_2, do not insert the element and continue to look for the next coordinate (1, 1) until all (1, 1) positions in matrix I_2 are filled with the elements. $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, N$, and C_d^{row} and C_d^{col} denote the dth round results of the evolution of ECA.

Step 6.4: Set $d = d + 1$, then loop executes Step 6.3 and Step 6.4 e times, $d = 1, 2, \ldots, e$, and e is the total number of evolution of ECA and part of secret keys.

Step 6.5: After e time evolution, if there are elements in p_1, then put them into I_2. Particularly, sequentially put the remaining elements of p_1 into the blank space of the matrix I_2 in row-major. After that, one round scrambling process is completed and the permutated matrix I_3 $(N \times N)$ is gotten.

Step 6.6: Set $s = s + 1$, then transform I_3 $(N \times N)$ to 1-D vector p_1 $(1 \times N^2)$. Next, loop executes Step 6.2 to Step 6.6 w times, and the final permutated matrix I_4 $(N \times N)$ is obtained. $s = 1, 2, \ldots, w$, and w is part of secret keys and the total number of scrambling rounds.

Step 7: Construct measurement matrix Φ with a size of $m \times N$ as described in Section 3.4, $m = \mathrm{CR} \times N$, and CR is compression ratio of the plain image.

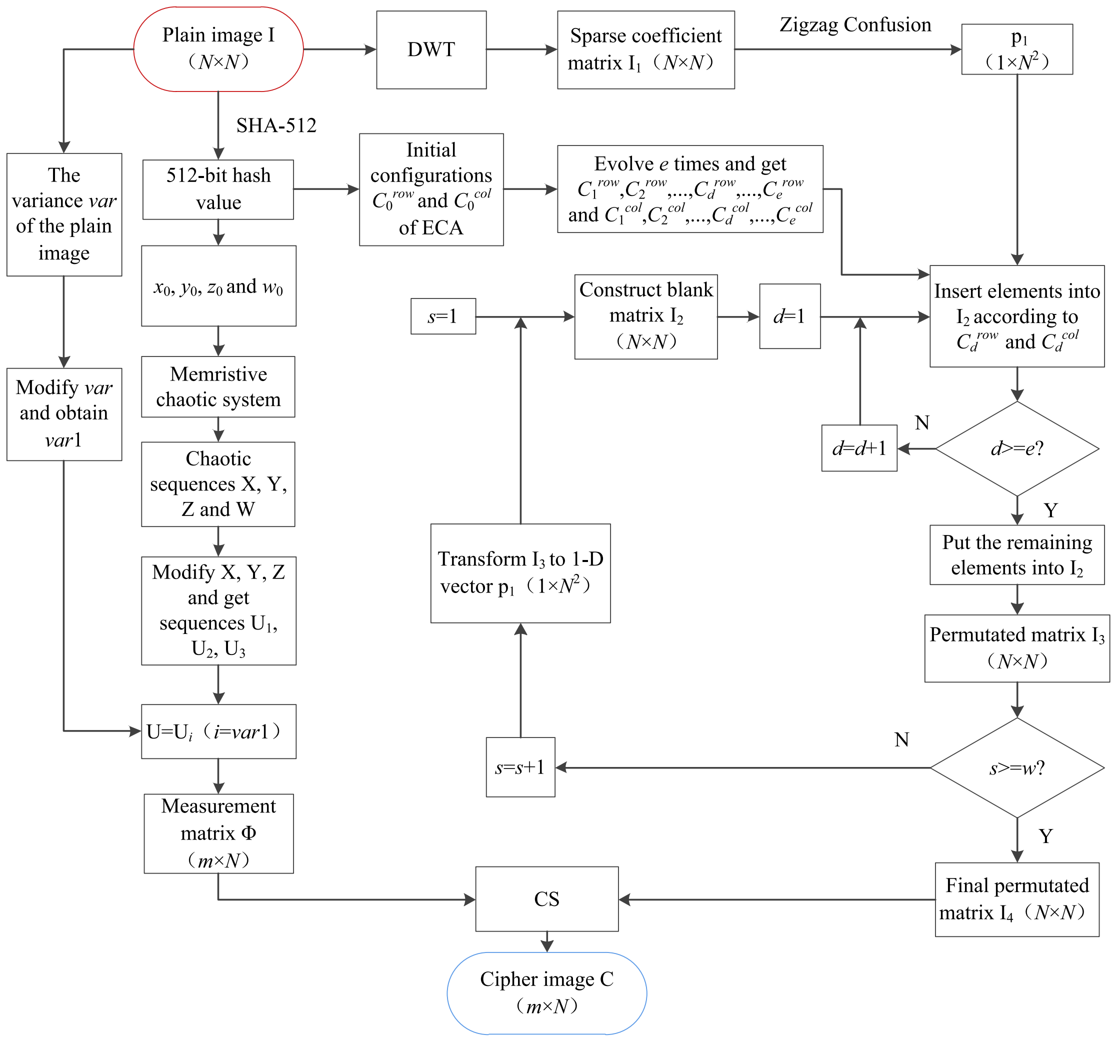

Plain image I (N×N)
DWT
Sparse coefficient matrix I1 (N×N)
Zigzag Confusion
p1 (1×N²)
SHA-512
The variance var of the plain image
512-bit hash value
Initial configurations C0row and C0col of ECA
Evolve e times and get C1row,C2row,...,Cdrow,...,Cerow and C1col,C2col,...,Cdcol,...,Cecol
x0, y0, z0 and w0
s=1
Construct blank matrix I2 (N×N)
d=1
Insert elements into I2 according to Cdrow and Cdcol
Modify var and obtain var1
Memristive chaotic system
Chaotic sequences X, Y, Z and W
N
d=d+1
d>=e?
Y
Modify X, Y, Z and get sequences U1, U2, U3
Transform I3 to 1-D vector p1 (1×N²)
Put the remaining elements into I2
Permutated matrix I3 (N×N)
U=Ui (i=var1)
N
s=s+1
s>=w?
Y
Measurement matrix Φ (m×N)
CS
Final permutated matrix I4 (N×N)
Cipher image C (m×N)

Fig. 6. The flow chart of the proposed encryption algorithm.

Step 8: Compress and encrypt the final permutated matrix I_4 $(N \times N)$ using compressive sensing technology denoted as $C = \Phi I_4$. Finally, the cipher image C with size of $m \times N$ is obtained.

3.6. The decryption algorithm

The decryption process is depicted in Fig. 7, which is the inverse operation of the encryption process. Before the decryption, secret keys including 512-bit hash value K, abandoning number n_0 of chaotic sequences, the total number e of evolutions, the total number w of scrambling rounds, five parameters: λ, t_2, t_3, t_4, t_5, and intermediate key $var1$ are transmitted to the receiver. And the four parameters x_0, y_0, z_0, w_0, initial location (x_0', y_0') and the measurement matrix Φ are firstly computed as described in Section 3. What's more, get the initial configurations C_0^{row} and C_0^{col} of the ECA according to Section 3.3, then evolve them to obtain row configuration $C_1^{row}, C_2^{row}, \ldots, C_d^{row}, \ldots, C_e^{row}$ and column configuration $C_1^{col}, C_2^{col}, \ldots, C_d^{col}, \ldots, C_e^{col}$ as illustrated in Section 3.5.

The cipher image C $(m \times N)$ is reconstructed with the SL_0 algorithm and the reconstructed image F_1 $(N \times N)$ is gotten. Then, perform inverse scrambling on F_1, and the detailed procedures are as follows.

Step 1: Set $s = 1$.

Step 2: Construct an $1 \times N^2$ blank vector p_3 and set $d = 1$.

Step 3: The elements at the position of coordinates (1, 1) in F_1 are sequentially taken out in row-major, and put into vector p_3 one by one. The coordinates are (1, 1), that is $(C_d^{row}(i), C_d^{col}(j))=(1,1)$, until all the elements at coordinates (1, 1) in F_1 are taken out. In particular, if the position of a coordinate (1, 1) in F_1 is empty, then continue to look for the next coordinate (1, 1) until all (1, 1) positions in matrix F_1 are empty. $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, N$, C_d^{row} and C_d^{col} denote the dth round results of the evolution of ECA.

Step 4: Set $d = d + 1$, then loop executes Step 3 and Step 4 e times, $d = 1, 2, \ldots, e$.

Step 5: After e round execution, if there are elements in F_1, take them out sequentially, and then put them into the blank place in vector p_3 one by one. After that, complete one round inverse scrambling process and get inverse permutated vector p_3 $(1 \times N^2)$.

Step 6: Set $s = s + 1$, then transform p_3 $(1 \times N^2)$ to matrix F_1 $(N \times N)$. Next, loop manipulates Step 2 to Step 6 w times, and the final inverse permutated vector p_4 $(1 \times N^2)$ is obtained. And, $s = 1, 2, \ldots, w$.

Step 7: Transform p_4 $(1 \times N^2)$ to matrix F_2 $(N \times N)$, then operate inverse zigzag confusion and inverse DWT to F_2. Finally, the decrypted image Y $(N \times N)$ are gotten.

3.7. Discussion

The proposed image encryption scheme has some advantages.

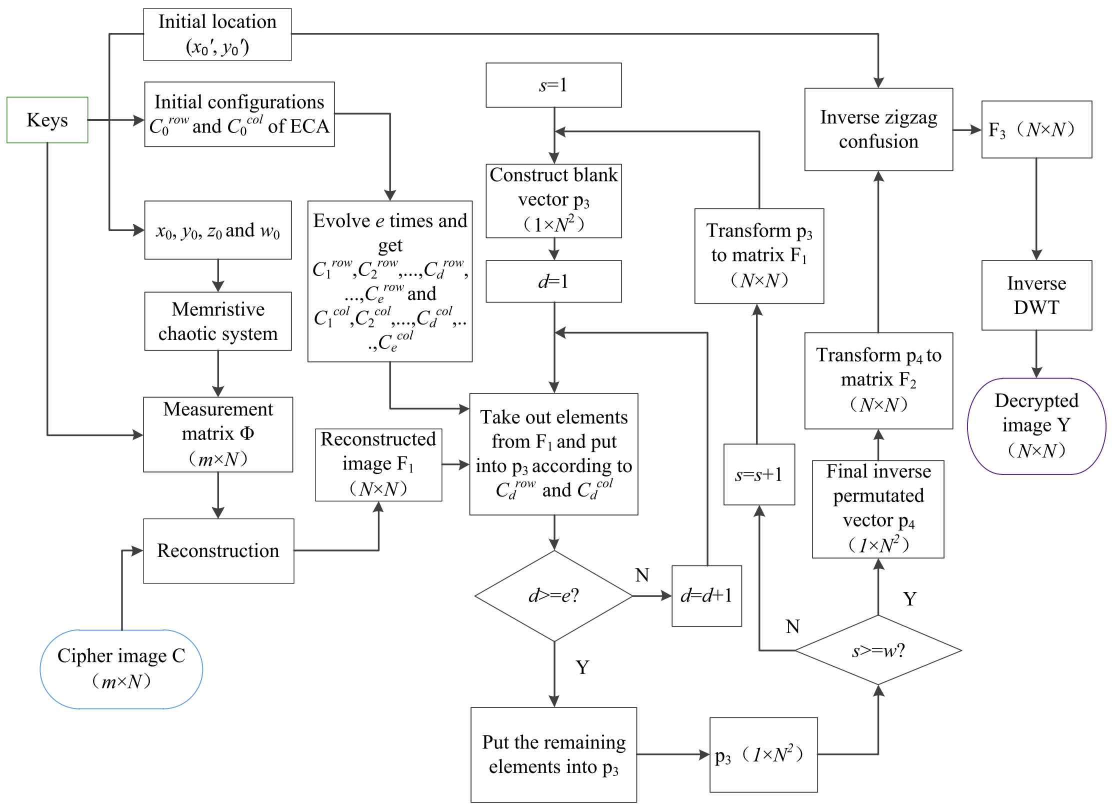

Initial location
(x_0', y_0')
Keys
Initial configurations
C_0^{row} and C_0^{col} of ECA
x_0, y_0, z_0 and w_0
Memristive chaotic system
Measurement matrix Φ (m×N)
Reconstruction
Cipher image C (m×N)
Evolve e times and get $C_1^{row}, C_2^{row}, \ldots, C_d^{row}, \ldots, C_e^{row}$ and $C_1^{col}, C_2^{col}, \ldots, C_d^{col}, \ldots, C_e^{col}$
Reconstructed image F₁ (N×N)
s=1
Construct blank vector p₃ (1×N^2)
d=1
Take out elements from F₁ and put into p₃ according to C_d^{row} and C_d^{col}
d>=e?
N
d=d+1
Y
Put the remaining elements into p₃
p₃ (1×N^2)
Transform p₃ to matrix F₁ (N×N)
s=s+1
N
s>=w?
Y
Final inverse permutated vector p₄ (1×N^2)
Transform p₄ to matrix F₂ (N×N)
Inverse zigzag confusion
F₃ (N×N)
Inverse DWT
Decrypted image Y (N×N)

Fig. 7. The flow chart of the decryption algorithm.

Firstly, two scrambling methods, that is zigzag confusion and the scrambling based on ECA, are utilized. Wavelet coefficients of the plain image are confused by a zigzag path, and then they are re-scrambled by the ECA. The scrambling scheme of the combination of zigzag confusion and ECA confusion may make the pixels of the confused images distributed evenly, and improve the image scrambling degree.

Secondly, compressive sensing (CS) is used to compress and encrypt the confused images for reducing the transmission bandwidth of the dada, so that the cipher images can be sent and stored quickly and efficiently. What's more, the random zigzag confusion and ECA confusion scheme are adopted to confuse the sparse coefficient matrix of the plain image, which can relax the restricted isometry property (RIP) of CS, effectively promote the compression performance and reconstruction effect. In addition, the measurement matrix Φ is constructed as a circular matrix. And it can be obtained by some key parameters. There is no need to deliver the whole measurement matrix to the receiver for decryption, which can save a lot of transmission bandwidth and storage space.

Thirdly, the proposed encryption scheme is highly sensitive to the plain image. The SHA 512 hash function of the original image is used to compute four important parameters h_1, h_2, h_3 and h_4. Then the parameters are utilized for calculating the initial values x_0, y_0, z_0, w_0 of the chaotic system and the initial location (x_0', y_0') for zigzag confusion. Moreover, the initial configurations C_0^{row} and C_0^{col} of ECA are controlled by the 512-bit hash value, that is to say, different scrambling effects are obtained for different plain images. Therefore, the algorithm has a close relationship with the plain image, it can resist against known-plaintext and chosen-plaintext attacks effectively.

Lastly, the measurement matrix is generated by a new kind of magnetic-controlled memristive chaotic system, which has strong sensitivity to initial values and system parameters. By modifying the parameters and initial values of the chaotic system, we can generate different measurement matrices to realize "one time, one key" . And thus our algorithm has high security level.

4. Simulation results

In this section, we have employed Matlab R2016a to verify the encryption and decryption effects of the proposed algorithm in a personal computer with 3.3 GHz CPU and 4 GB memory, the operating system is Microsoft Windows 7. The four different 512 × 512 images "Lena", "Pepper", "brone", "aerial" and five different 256 × 256 images "Lena256", "finger", "Cameraman", "Baboon", "Peppers256" are all used as the plain images.

4.1. Encryption results and decryption results for different images

The parameters we used are as follows: $a = 10$, $b = 8$, $c = 15$, $d = 5.2$, $g = 5$ and $r = 1.5$, scrambling rounds $w = 3$, evolution rounds $e = 9$, $\lambda = 2.3$, $t_2 = 33.2418$, $t_3 = 3.5609$, $t_4 = 2.67$, $t_5 = 1.0314$, $n_0 = 800$, and the ECA rule is 170.

The three different plain images Lena (512 × 512), finger (256 × 256) and brone (512 × 512) are shown in Figs. 8(a), (b) and (c), respectively. Figs. 9-11 are the respective encryption and decryption results with different compression ratios. The compression ratio CR is computed by [32],

$$CR = \frac{C\_height \times C\_width}{I\_height \times I\_width} \tag{18}$$

where *I_height* and *I_width* denote the height and width of the original image, respectively. And *C_height* and *C_width* are the corresponding height and width of the cipher image.

Fig. 9 is the encryption and decrypted results for Lena (shown in Fig. 8(a)) with different CR. As can be seen from the figure that the compression ratio CR varies from 0.25, 0.5 to 0.75. Similarly, the results of finger (256 × 256) and brone (512 × 512) are illustrated in Figs. 10 and 11.

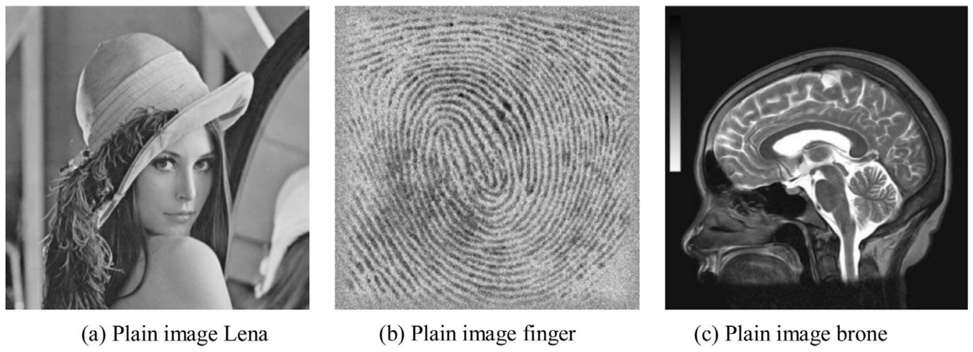


Fig. 8. The plain images: (a) Lena, (b) finger, (c) brone.

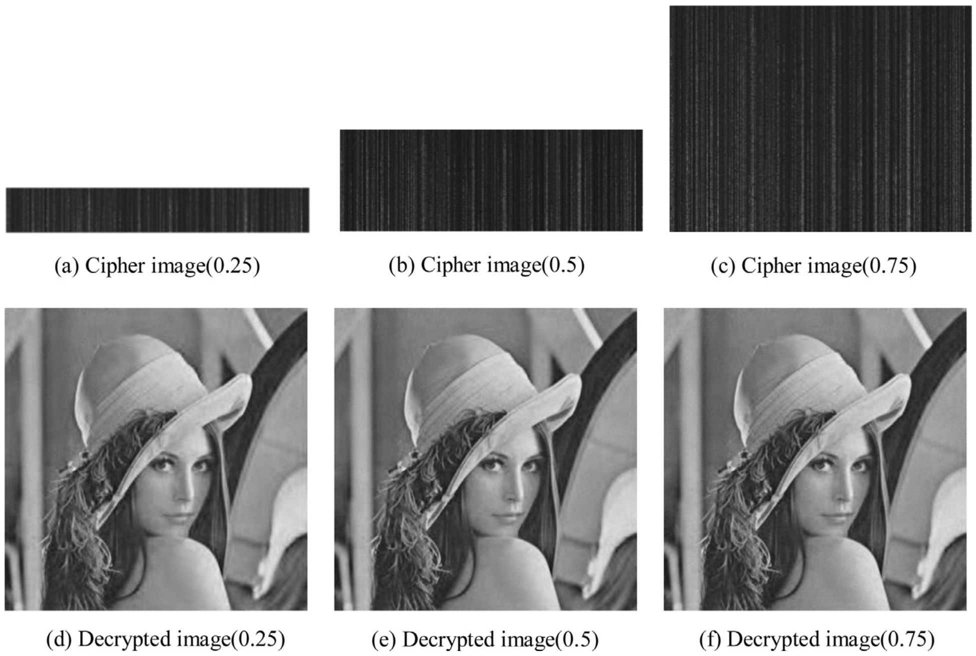


Fig. 9. Encryption and decryption results of Lena (512 × 512): (a)–(c) are the cipher images when CR is 0.25, 0.5, and 0.75, respectively, (d)–(f) are the corresponding decrypted images.

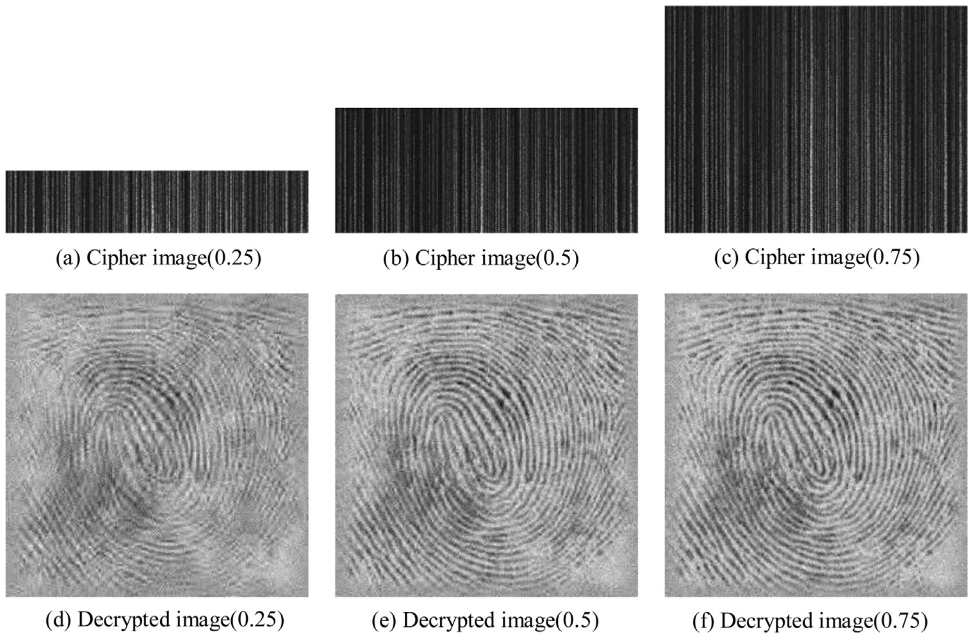


Fig. 10. Encryption and decryption results of finger (256 × 256): (a)–(c) are the cipher images when CR is 0.25, 0.5, and 0.75, respectively, (d)–(f) are the corresponding decrypted images.

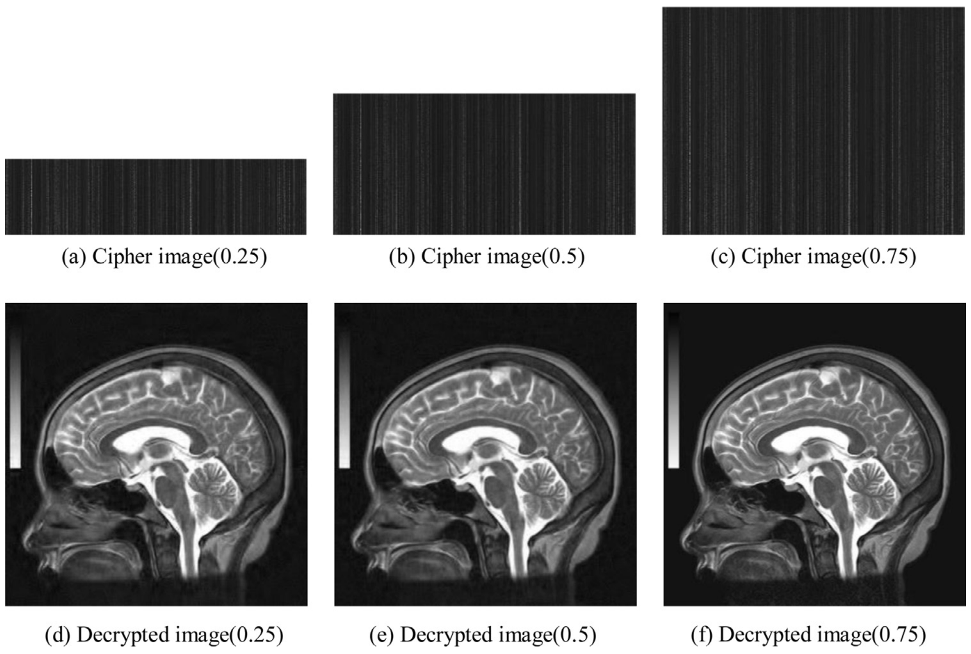


Fig. 11. Encryption and decryption results of brone (512×512): (a)–(c) are the cipher images when CR is 0.25, 0.5, and 0.75, respectively, (d)–(f) are the corresponding decrypted images.

As shown in Figs. 9-11, at different compression ratios, the images can be effectively encrypted and the information of the plain images is not available in the cipher images, which preserve image information available. In addition, from the visual point of view, the decrypted images illustrated in Figs. 9–11 are just the same as the respective plain images shown in Figs. 8(a)–(c). So it can be seen that the algorithm has better encryption and decryption effect. Moreover, the compressed cipher image is smaller than the respective original image, which may save the transmission bandwidth and storage space. And when the compression ratio decreases, the cipher image becomes smaller.

4.2. The effect of the compression ratio on simulation results

In this part, we adopt Mean Structural Similarity (MSSIM) and Peak Signal to Noise Ratio (PSNR) to evaluate the performance of the proposed compression and encryption algorithm at different compression ratios.

(1) **Mean Structural Similarity (MSSIM)**

Structural Similarity (SSIM) is a new index for measuring the similarity between two images. It can improve the traditional metrics such as PSNR by considering human visual system (HVS) [32]. The mean SSIM (denoted as MSSIM) is often used to evaluate the performance of the encryption algorithm, and it is defined as

$$l(\mathrm{X},\mathrm{Y})=\frac{2\mu_{\mathrm{X}}\mu_{\mathrm{Y}}+C_1}{\mu_{\mathrm{X}}^2+\mu_{\mathrm{Y}}^2+C_1} \tag{19}$$

$$c(\mathrm{X},\mathrm{Y})=\frac{2\sigma_{\mathrm{X}}\sigma_{\mathrm{Y}}+C_2}{\sigma_{\mathrm{X}}^2+\sigma_{\mathrm{Y}}^2+C_2} \tag{20}$$

$$s(\mathrm{X},\mathrm{Y})=\frac{\sigma_{\mathrm{XY}}+C_3}{\sigma_{\mathrm{X}}\sigma_{\mathrm{Y}}+C_3} \tag{21}$$

$$\mathrm{SSIM}(\mathrm{X},\mathrm{Y}) = l(\mathrm{X},\mathrm{Y}) \times c(\mathrm{X},\mathrm{Y}) \times s(\mathrm{X},\mathrm{Y}) \tag{22}$$

and

$$\mathrm{MSSIM}(\mathrm{X},\mathrm{Y}) = \frac{1}{M}\sum_{k=1}^{M}\mathrm{SSIM}(x_k, y_k) \tag{23}$$

where μ_{X} and μ_{Y} represent the average values of plain image X and decrypted image Y, σ_{X} and σ_{Y} denote the variance values of X and Y, respectively, σ_{XY} is the covariance of X and Y and C_1, C_2, C_3 are constants, M indicates the total number of image blocks. The parameters we used are as follows: $C_1 = (K_1 \times L)^2$, $C_2 = (K_2 \times L)^2$, $C_3 = \frac{C_2}{2}$, $k_1 = 0.01$, $k_2 = 0.03$, $M = 64$, L is the gray level of plain images and $L = 255$.

Table 2 lists the MSSIM results for different images at different compression ratios. The smaller the MSSIM is, the greater the difference of two images is. From Table 2, we can watch that: (1) for decrypted images, the MSSIM > 0.96, among them, the decrypted image of Lena image and aerial image are more than 0.99. And for finger (256×256), when CR = 0.5 and CR = 0.75, their MSSIM are more than 0.99, too. That is to say, regardless of the size of the original image, the similarity between the original image and the reconstructed image is high, and the image distortion is small, thus, the algorithm can achieve better recovery effect. (2) In the simulation, when the compression ratio is changing, the value of the MSSIM will be changed correspondingly, but the change is not big. The results listed in Table 2 reflect that we can effectively compress and encrypt images according to the different requirements.

(2) **Peak Signal to Noise Ratio (PSNR)**

In order to compare our algorithm with other methods, we also compute the PSNR. PSNR is usually used to judge the quality of reconstructed image after image processing, which is calculated using the following formula [38]

$$\mathrm{MSE} = \frac{1}{N \times N}\sum_{i=1}^{N}\sum_{j=1}^{N}(\mathrm{X}(i,j) - \mathrm{Y}(i,j))^2 \tag{24}$$

$$\mathrm{PSNR} = 10 \times \log_{10}\left(\frac{255 \times 255}{\mathrm{MSE}}\right) \tag{25}$$

where X(i, j) and Y(i, j) are the pixel values of the original image and the decrypted image, respectively. MSE represents the mean variance of the original image and decrypted image, and N is the size of the image.

Table 2
MSSIM values of different images under different compression ratios.

Images Size	Lena 512 × 512	brone 512 × 512	Peppers 512 × 512	aerial 512 × 512	finger 256 × 256	Baboon 256 × 256
CR = 0.25	0.9956	0.9802	0.9740	0.9904	0.9817	0.9693
CR = 0.5	0.9964	0.9777	0.9727	0.9959	0.9901	0.9823
CR = 0.75	0.9957	0.9675	0.9696	0.9962	0.9936	0.9892

Table 3
The compression performance of different algorithms.

Images (256 × 256)	CR	Ours	Ref. [47]
Lena 256	CR = 0.25	26.06dB	25.93dB
	CR = 0.5	29.82dB	29.82dB
	CR = 0.75	29.56dB	34.19dB
Cameraman	CR = 0.25	25.23dB	22.64dB
	CR = 0.5	29.43dB	26.71dB
	CR = 0.75	28.93dB	30.85dB

The larger the value of the PSNR, the higher the similarity between the original image and the reconstructed image. Table 3 lists the compression performance of our method and other algorithm. From Table 3, one can see that the proposed algorithm has almost the same PSNR values as Ref. [47]. And the PSNR of our algorithm is close to 30 dB for CR > 0.5. Moreover, it can achieve a better reconstruction result when the image is sampled with a small amount of compression.

4.3. The effect of different parameters on gray difference degree (GDD)

Before compression, using the lightweight encryption method to encrypt the images can achieve a good encryption effect [18]. In the proposed encryption, lightweight encryption may have a high scrambling effect. The scrambling method based on zigzag and ECA is one phase of our proposed compression and encryption scheme. Therefore, it is necessary to evaluate scrambling degree. To evaluate the effect of image scrambling, we introduce gray difference degree (GDD) [37], which is defined as

$$\mathrm{GDD} = \frac{E'(GD(i,j)) - E(GD(i,j))}{E'(GD(i,j)) + E(GD(i,j))} \tag{26}$$

where

$$GD(i,j) = \frac{1}{4}\sum_{i',j'}\left[\mathrm{I}(i,j) - \mathrm{I}(i',j')\right]^2 \tag{27}$$

$$E(GD(i,j)) = \frac{\sum_{i=2}^{M-1}\sum_{j=2}^{N-1} GD(i,j)}{(M-2)\times(N-2)} \tag{28}$$

where $(i', j') = \{(i-1, j), (i+1, j), (i, j-1), (i, j+1)\}$, and $\mathrm{I}(i, j)$ denotes the gray value at position (i, j) of the original image I. M and N represent the size of the image. $E(GD(i, j))$ and $E'(GD(i, j))$ denote the corresponding average neighborhood gray difference of the plain image and permutated image.

The GDD value will be a number between −1 and 1. Better scrambling effect corresponds to an absolute value near 1 [37,39].

4.3.1. The influence of the number of evolutions and the number of scrambling rounds on GDD

In this paper, e is the total number of evolutions of ECA, and w is the total number of scrambling rounds. The number of evolutions is important as it reflects the characteristics of the ECA evolution, and different evolution rounds generate different confusing effect. Thus, in simulation, we compute GDDs of Lena image (512 × 512) at different e when $w = 1, 2, 3, 4$, and GDDs of brone image (512 × 512) at different e when $w = 1, 2, 3, 4, 5$, respectively. And the ECA rule is 170. The results are shown in Figs. 12 and 13.

From Figs. 12 and 13, it is clear that: (1) the encryption method used in this algorithm has better scrambling effect, and the GDDs of some images are more than 0.95. So the security of data information can be effectively protected. (2) When the number w of scrambling rounds is constant, with the number e of ECA evolutions increases, GDD also increases correspondingly, but when e increases to a certain value, the GDD changes little. The relation is close, although the influence gradually decreases when the number of evolutions is greater than 7. Therefore, the more the number of evolutions, the better the scrambling effect. (3) Similarly, when e is constant, along with the increasing of w, the GDD also increases correspondingly, while when w increases to a certain value, the GDD changes rarely. It can be concluded that the number of evolutions and the number of scrambling rounds have a certain effect on GDD, but both tend to be a stable value. (4) For the two figures, when $w = 1$, the GDD of Lena image is between 0.9422 and

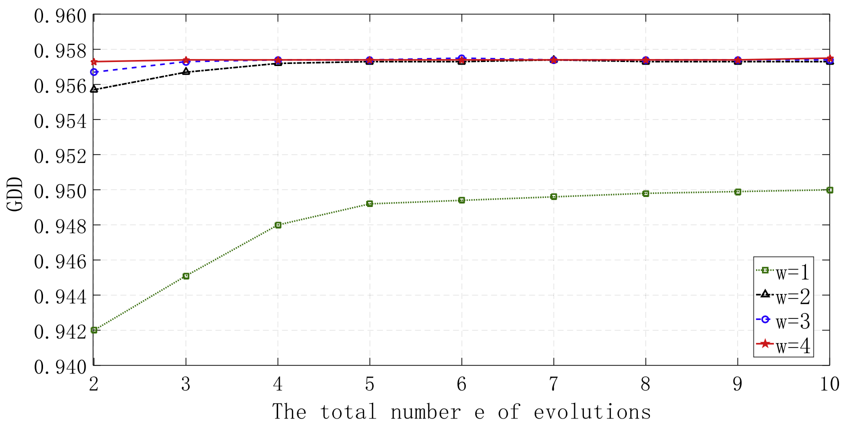


Fig. 12. GDDs of the Lena image (512 × 512) with different numbers of evolutions when scrambling rounds $w = 1, 2, 3, 4$.

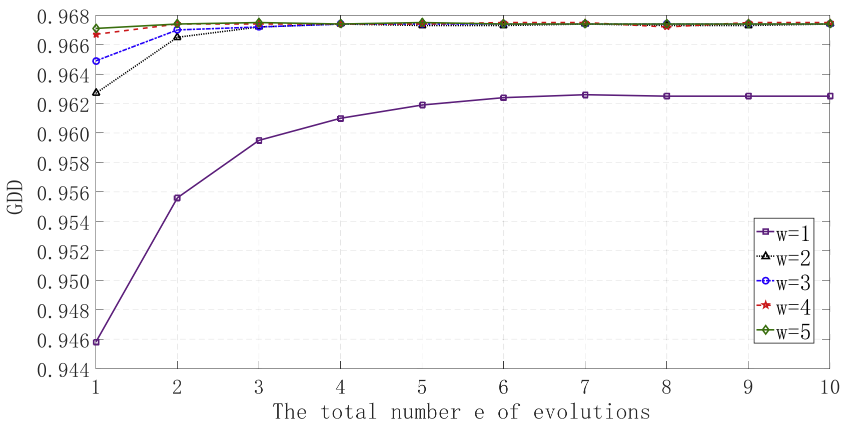


Fig. 13. GDDs of the brone image (512 × 512) with different numbers of evolutions when scrambling rounds $w = 1, 2, 3, 4, 5$.

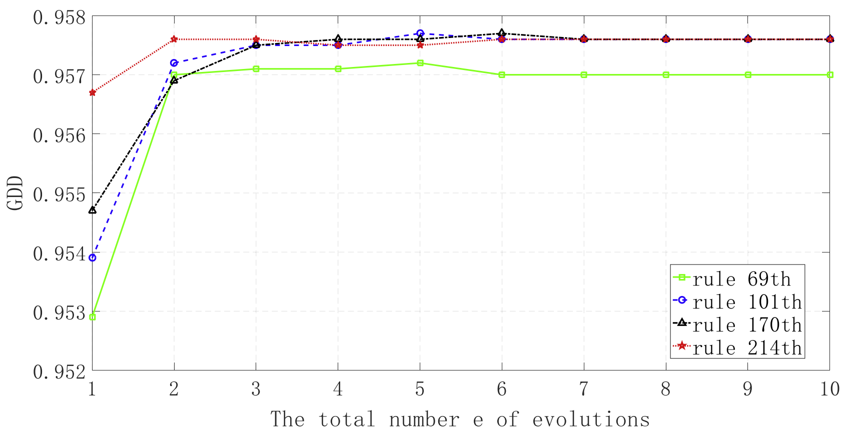


Fig. 14. GDD of the Lena image (512 × 512) with different numbers of evolutions and ECA rules.

0.9502, the GDD of brone image is between 0.9458 and 0.9626, the scrambling effect is poor and the change is greatest. Hence, it is best to set $w \in (1, 5)$ and $e \in (5, 10)$. This can not only get better scrambling effect and improve security, but also can save time and upgrade encryption efficiency. And we will use $w = 3$ and $e = 9$ in the following experiments.

4.3.2. The influence of different ECA rules on GDD

The evolution rule of ECA defines the deterministic way to update the synchronization state of all cells, and simply, the rule influences the scrambling process in our algorithm. Thus, it is important to analyze its effect on the confusion results. In this simulation, we select ECA rule 69th, 101th, 170th and 214th to test the GDDs of Lena image and brone image with different numbers of evolutions, respectively. The results are shown in Figs. 14 and 15.

It can be seen from Figs. 14 and 15 that the GDDs are greater than 0.95 when different ECA rules are used. When the total number e of evolutions is constant, the GDD is different and the corresponding scrambling encryption effect is different, too. That is to say, different evolution rules have a different impact on scrambling effect, and in practical application, it can upgrade security level by changing rules. Similarly, we can know that when $e > 3$, the GDDs are high and will gradually stabilized. Therefore, the best number e of ECA evolution is greater than 3 and no more than 10, which can improve encryption efficiency.

Table 4
Comparison of key space of the proposed method and other methods.

Algorithm	Proposed	Ref. [40]	Ref. [51]	Ref. [52]	Ref. [53]
Key Space	$> 2^{232}$	2^{96}	10^{48}	$10^{44} \times 5$	2^{78}

5. Performance analyses

5.1. Key space analysis

In cryptography, the key space of an algorithm is one of the most important attributes that determines the strength of a cryptosystem [29,48]. The key space of an ideal cryptographic system should be large enough to make brute-force attack infeasible, and in general, the key space should more than 2^{100} [49,50].

The secret key of the proposed encryption algorithm mainly includes: (1) 512-bit hash value K from the SHA-512 hash function of the original image; (2) the given parameters: λ, t_2, t_3, t_4, t_5. Besides, the abandoning number n_0 of chaotic sequences, the total number e of evolutions, the total number w of scrambling rounds, and intermediate key $var1$ can also be taken as secret keys. If the computational precision of the computer is 10^{-14}, the key space is about $(10^{14})^5 = 10^{70} > 2^{232}$. If the 512-bit hash value K is considered, the overall key space is much larger than 2^{100}. Hence, the key space of our method is large enough to resist all kinds of brute-force attacks. In addition, as shown in Table 4, the proposed

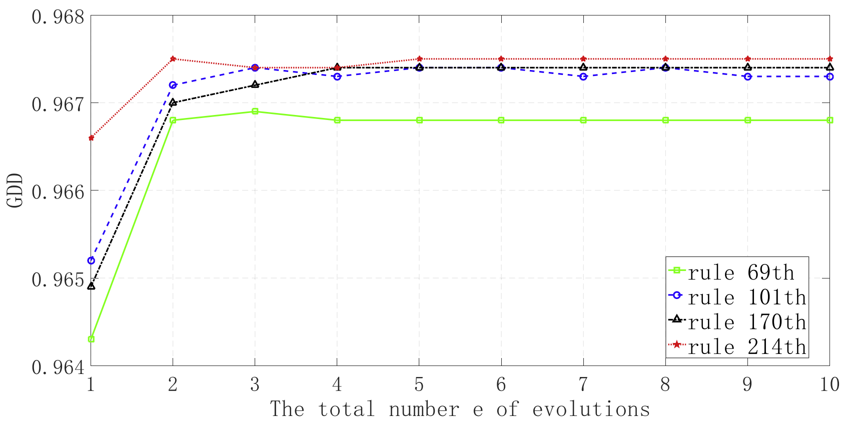


Fig. 15. GDD of the brone image (512 × 512) with different numbers of evolutions and ECA rules.

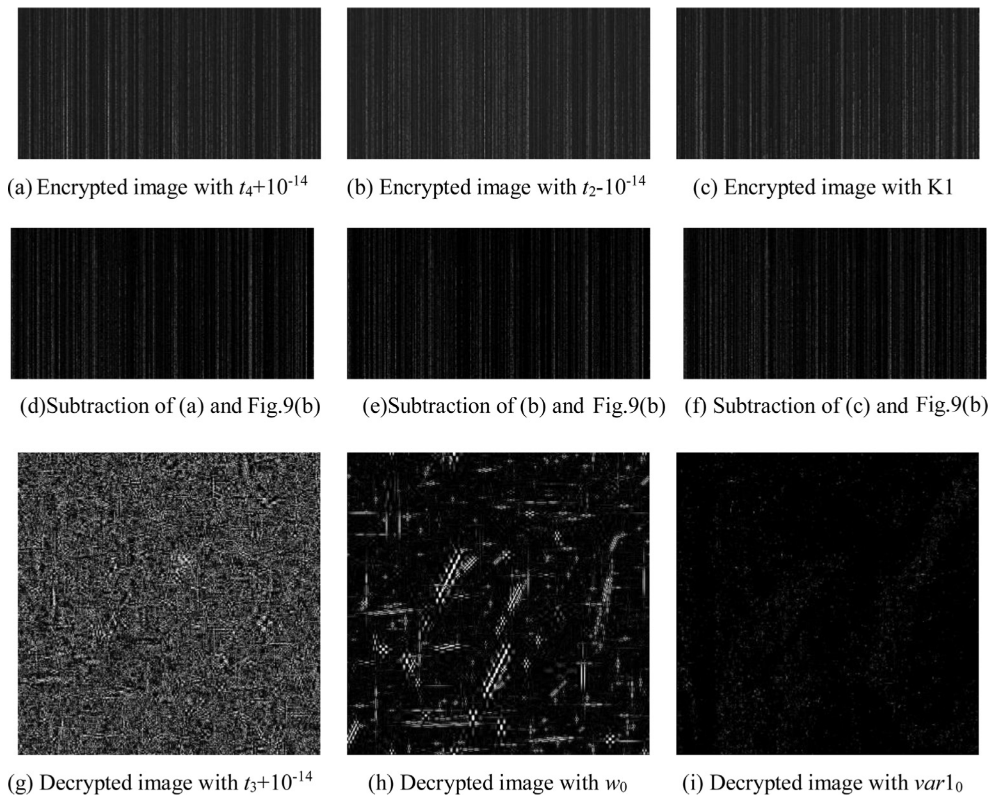

(a) Encrypted image with t_4+10^{-14}
(b) Encrypted image with t_2-10^{-14}
(c) Encrypted image with K1
(d)Subtraction of (a) and Fig.9(b)
(e)Subtraction of (b) and Fig.9(b)
(f) Subtraction of (c) and Fig.9(b)
(g) Decrypted image with t_3+10^{-14}
(h) Decrypted image with w_0
(i) Decrypted image with $var1_0$

Fig. 16. Key sensitivity test results in encryption and decryption process.

scheme has the largest key space than the encryption schemes in Ref. [40,51–53].

5.2. Key sensitivity analysis

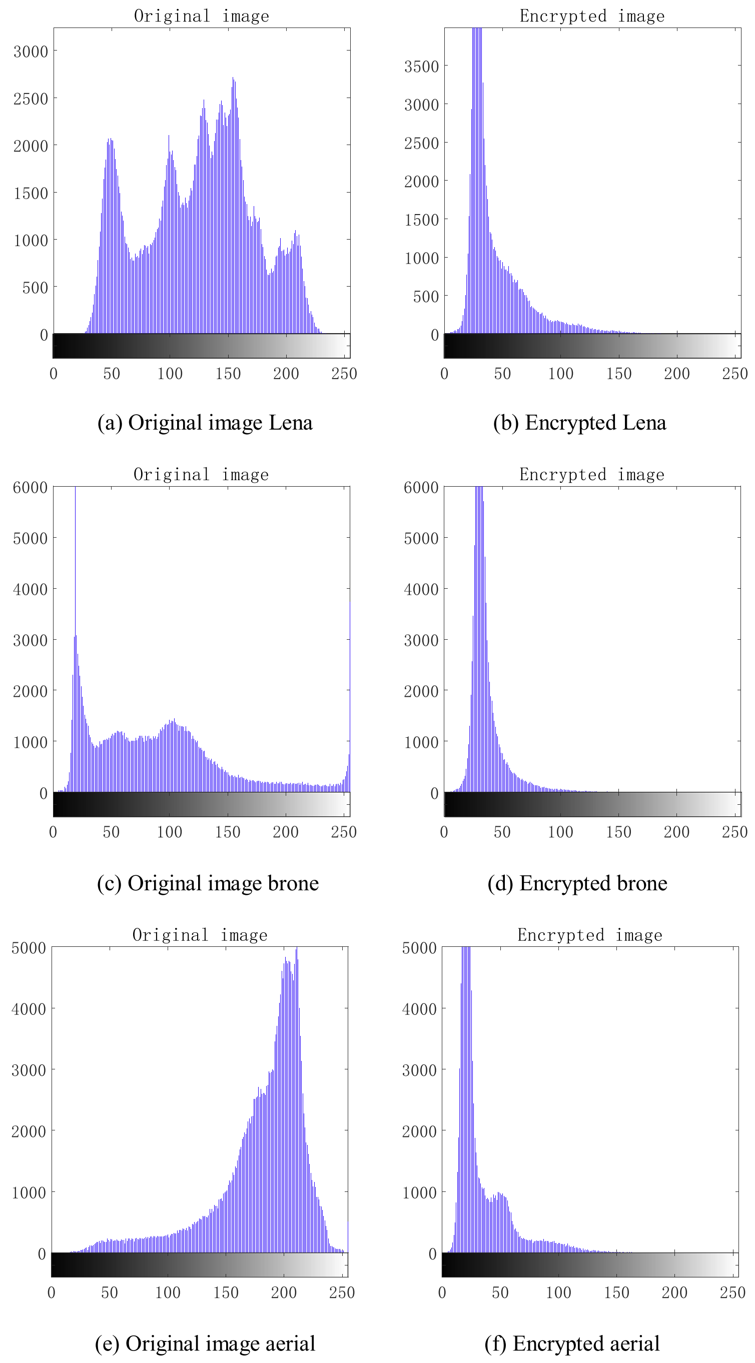
A good image encryption algorithm should have a high sensitivity to secret keys in encryption and decryption process. It means that a tiny change in the keys would cause a great distortion in the encrypted and decrypted image [7].

In encryption process, Lena image (shown as Fig. 8(a)) is used as the plain image and the correct encrypted image and decrypted images are shown in Fig. 9(b) and (e), respectively. We change t_4 by adding 10^{-14} and change t_2 by subtracting 10^{-14}. We also modify a bit of 512-bit hash value K, and a new K1 is obtained. K and K1 are illustrated as follows:

K =[7 C 2 3 4 4 A C 5 6 8 A E F B D B 8 7 D A 3 3 D C E 5 E D 4 C C 8 6 6 4 D F D 0 3 7 3 0 B 5 3 F 8 5 4 5 B D 3 1 B F 9 9 0 F 1 C E A 7 3 2 B 5 D 7 4 0 F 3 C E 2 3 0 1 3 3 1 6 0 F 7 C 8 4 5 2 4 F F 5 4 2 1 5 9 4 9 5 7 4 9 4 5 1 2 D C 2 F C D 6 6 A D 4 0 9 1]

K1 =[6 C 2 3 4 4 A C 5 6 8 A E F B D B 8 7 D A 3 3 D C E 5 E D 4 C C 8 6 6 4 D F D 0 3 7 3 0B 5 3 F 8 5 4 5 B D 3 1 B F 9 9 0 F 1 C E A 7 3 2 B 5 D 7 4 0 F 3 C E 2 3 0 1 3 3 1 6 0 F 7 C 8 4 5 2 4 F F 5 4 2 1 5 9 4 9 5 7 4 9 4 5 1 2 D C 2 F C D 6 6AD409 1]

These changed secret keys are used to encrypt the original image and every time one parameter is changed and others are constant, the corresponding encrypted images are shown in Figs. 16(a)–(c). In order to visually observe the change of cipher images, we get the subtractions of cipher images (Figs. 16(a)–(c)) and correct cipher image (Fig. 9(b)), the results are shown in Figs. 16(d)–(f).



(a) Original image Lena (b) Encrypted Lena

(c) Original image brone (d) Encrypted brone

(e) Original image aerial (f) Encrypted aerial

Fig. 17. Histogram analysis results.

Table 5
The NPCR between cipher images generated with slightly different keys and correct cipher image (Fig. 9(b)).

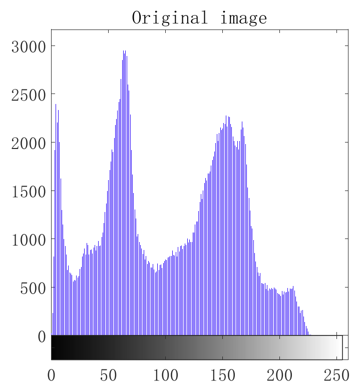
Secret keys	NPCR
Encrypted image with $t_4 + 10^{-14}$	0.9676
Encrypted image with t_2 - 10^{-14}	0.9634
Encrypted image with K1	0.9724

The number of pixel change rate (NPCR) is used to measure the dissimilitude between two encrypted images, and it stands for the percentage of different pixel numbers between two encrypted images [4,7]. The NPCR is defined by Eq. (29) [54]. And Table 5 lists the obtained NPCR between cipher images generated with slightly different keys and the correct cipher image (Fig. 9(b)).

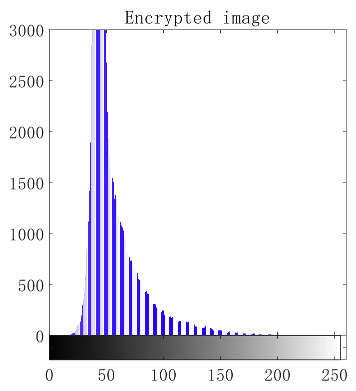
$$\mathrm{NPCR} = \frac{\sum_{i,j} \mathrm{D}(i,j)}{N \times N} \times 100\% \tag{29}$$

where D(i, j) is the difference value of the cipher images $C_1(i, j)$ and $C_2(i, j)$ produced with slightly different keys. When $C_1(i, j) = C_2(i, j)$, then D(i, j) = 0, when $C_1(i, j) \neq C_2(i, j)$, then D(i, j) = 1.

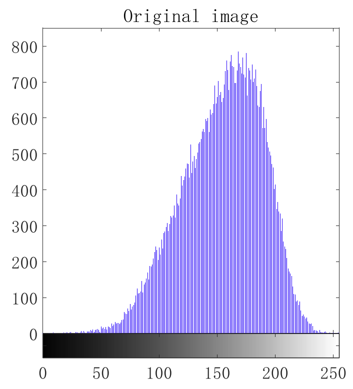
Next, we also change t_3 by adding 10^{-14} and modify each of w and $var1$ to get new w_0 and $var1_0$. That is to say, the right ones are $w = 3$ and $var1 = 1$, while the changed ones are $w_0 = 2$ and $var1_0 = 2$. Similarly, these changed secret keys are used to decrypt the cipher image (Fig. 9(b)) to test the key sensitivity in the decryption process, the corresponding decrypted images are illus-



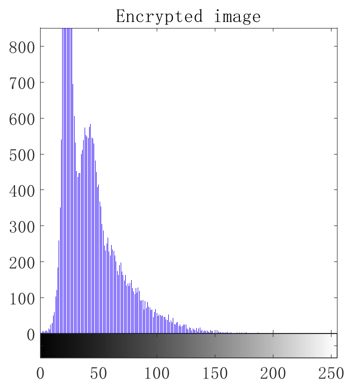
(g) Original image Peppers



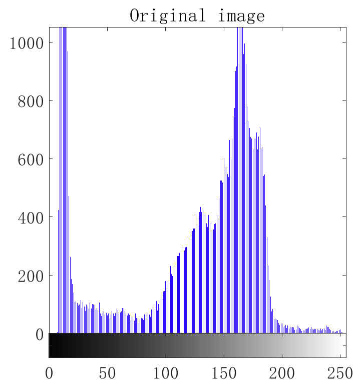
(h) Encrypted Peppers



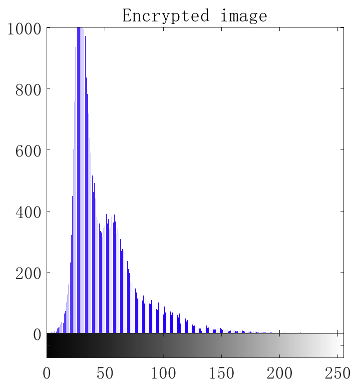
(i) Original image finger



(j) Encrypted finger



(k) Original image Cameraman



(l) Encrypted Cameraman

Fig. 17. Continued

trated in Figs. 16(g)–(i). It can be seen that the decrypted images cannot show any information of the original image even if a very small error occurs to secret keys.

From Fig. 16 and Table 5, it is clear that (1) a tiny change in the keys would cause a great change in the encrypted image and the NPCR is more than 0.96. (2) From Fig. 16(c) and Fig. 16(f), the tiny change of SHA-512 hash value of the original image has a great effect on the image encryption result and the NPCR is 0.9724, that is to say, when the original image changes slightly, the cipher image is different. The presented image compression–encryption algorithm is sensitive to the plain images. (3) The decryption images with incorrect keys in Fig. 16(g)–Fig. 16(i) are distorted greatly and show no visual information about the original image. In short, the algorithm is sensitive to keys. In the encryption phase, the encryption result changes greatly after the key is altered. In the decryption phase, it is difficult to recover the original image after the key is changed.

5.3. Histogram analysis

Histogram is an important statistical feature of the images, which is often used to evaluate the performance of image encryption schemes. It would be the best that the histogram of the encrypted image is fairly uniformed in distribution and that the histogram of the cipher image is flat [55] or the second best when histograms of different encrypted images are similar to each other [27,31,47].

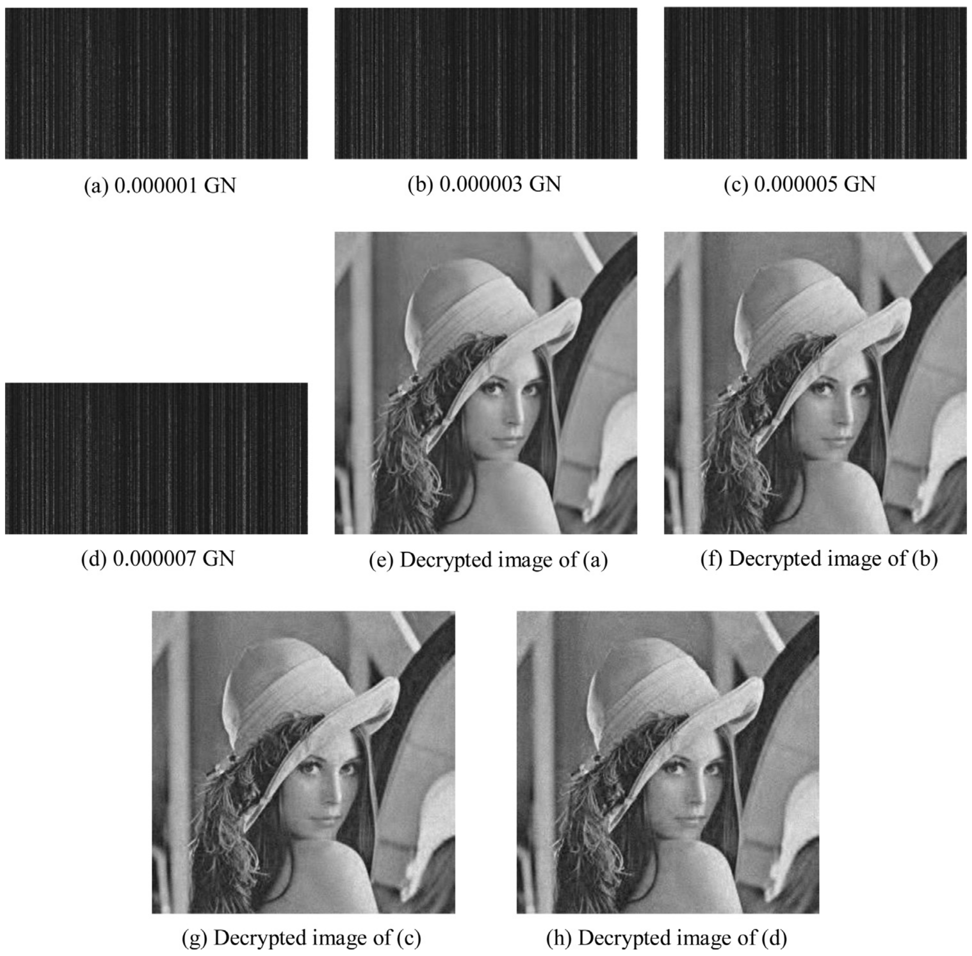

(a) 0.000001 GN
(b) 0.000003 GN
(c) 0.000005 GN
(d) 0.000007 GN
(e) Decrypted image of (a)
(f) Decrypted image of (b)
(g) Decrypted image of (c)
(h) Decrypted image of (d)

Fig. 18. Cipher and decrypted images under Gaussian noise with different levels of noise.

Figs. 17(a), (c), (e) and (g) are the histograms of four original images with size of 512 × 512: “Lena”, “brone”, “aerial” and “Peppers”, respectively. Figs. 17(b), (d), (f) and (h) are the histograms of their corresponding encrypted images. It is clear that the histograms of the four original images are obviously different from each other, while the histograms of their corresponding encrypted images are similar. Besides, Fig. 17(i) and (k) are the histograms of two original images with size of 256 × 256: “finger” and “Cameraman”, respectively. Fig. 17(j) and (l) are the histograms of their corresponding encrypted images. For images with size of 256 × 256, the histograms of cipher images are similar, too. Conclusively, when the hackers obtain the cipher image, he may not recover the plain image through analyzing the histogram, and thus the proposed scheme can make the statistical analysis attack infeasible to some extent.

5.4. Robustness analysis

5.4.1. Noise attack

During the transmission, the cipher images are easily affected by all kinds of noises, such as Gaussian noise (GN), Salt & Pepper noise (SPN) and Speckle noise (SN). It is necessary to discuss the performance of the proposed scheme against the noise attack [56].

In the simulation, different levels of noise are added to the cipher image of Lena (shown in Fig. 9(b)), and other parameters are set as Section 4.1. We will evaluate the ability of the proposed algorithm to resist the noise attack. Figs. 18–20 are the corresponding test results. Among them, Fig. 18 illustrates the cipher image (Fig. 9(b)) that is affected by Gaussian noise with different variances and the corresponding decrypted images, and the variances of the noise are 0.000001, 0.000003, 0.000005 and 0.000007. Similarly, the results affected by Salt & Pepper noise and Speckle noise are shown in Figs. 19 and 20, respectively. And the PSNRs of the noisy decrypted images and plain image Lena are illustrated in Fig. 21.

From the above Figs. 18–21, we can watch that: (1) GN noise has the largest effect on the decryption results, when the variances of the noise changes from 0.000001 to 0.000007, the PSNR values vary from 30.10 dB to 25.39 dB, and the corresponding recovered images are all clear. (2) As the PSNR values are almost the same during the whole process, our encryption scheme has the strongest resisting capability to SN and the PSNR values are changing from 31.05 dB to 30.99 dB; (3) The proposed encryption algorithm has a certain withstanding ability to SPN, the PSNR values vary from 31.05 dB to 27.93 dB, it is shown that the major information of the image can be recognized, although the quality of decrypted images decreases with the increasing of noise intensity. Thus, in the presence of noise, the recovered images are shown clearly, and the proposed encryption scheme has good robustness to noise attacks.

5.4.2. Cropping attack

The robustness of a cryptosystem against data loss is also an important requirement in image communication. It is necessary to discuss the performance of the proposed scheme against the cropping attack. The encrypted image (shown in Fig. 9(b)) with five different data losses are shown in Figs. 22(a)–(e) and the correspond-

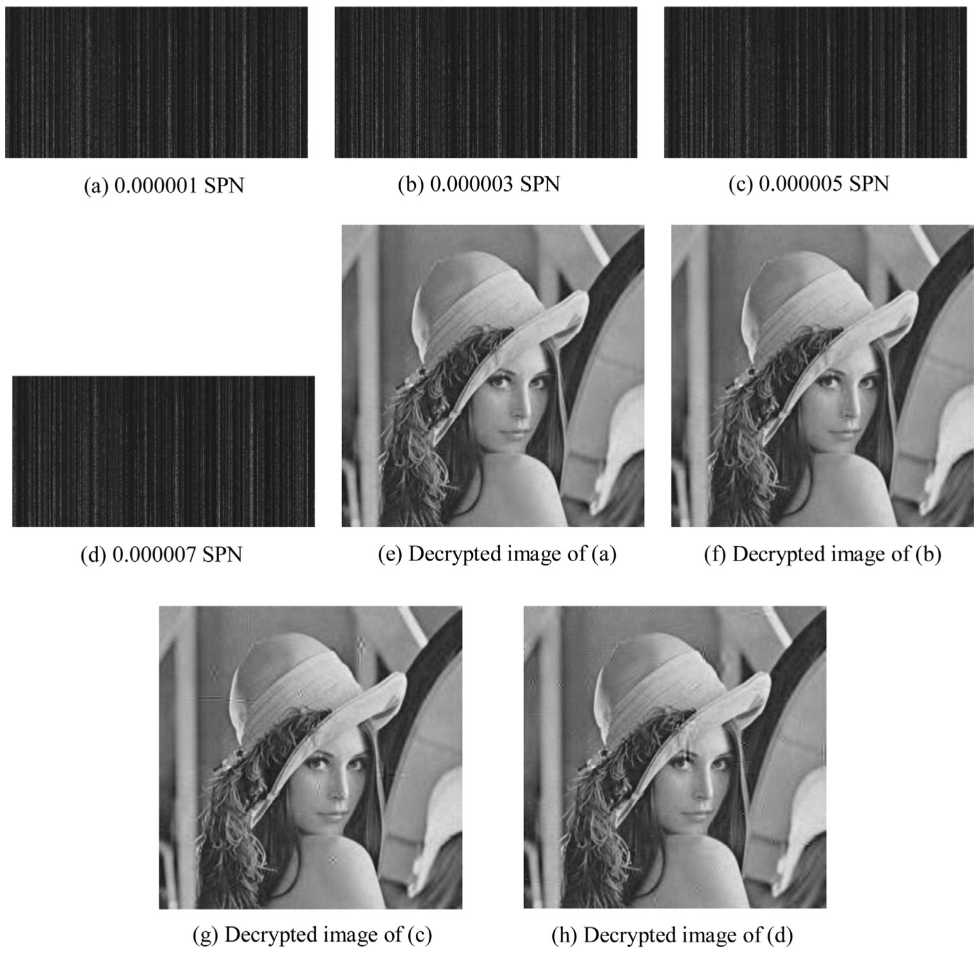

(a) 0.000001 SPN
(b) 0.000003 SPN
(c) 0.000005 SPN
(d) 0.000007 SPN
(e) Decrypted image of (a)
(f) Decrypted image of (b)
(g) Decrypted image of (c)
(h) Decrypted image of (d)

Fig. 19. Cipher and decrypted images under Salt & Pepper noise with different levels of noise.

Table 6
MSSIM between the decrypted image and the plain image with different data losses.

Data loss intensity	MSSIM
1/4 data loss	0.6410
1/8 data loss	0.7605
1/16 data loss	0.8161
1/32 data loss in the upper left corner	0.8919
1/32 data loss in the middle part	0.8840

ing recovered images are illustrated in Figs. 22(f)–(j), respectively. The MSSIM of the decrypted image and plain image Lena are listed in Table 6.

As can be seen from Fig. 22 and Table 6 that even if the cipher image is cropped off, the recovered image retains important information contained in the plain image. When the data loss size is varying from 1/32 to 1/4, the quality of recovered image decreases, and MSSIM values are changing from 0.8919 to 0.6410. Besides, different cutting positions get different results, cropping the 1/32 in the upper left corner of the cipher image, and the MSSIM is 0.8919, while the MSSIM of cropping the 1/32 in the middle part is 0.8840. In a word, our scheme could resist data loss attack to a certain degree.

In conclusion, the algorithm proposed in this paper has good robustness, and it can withstand noise attack and cropping attack to a certain degree. And it is more suitable for real applications.

5.5. Analysis of resisting known-plaintext and chosen-plaintext attacks

Plaintext attack is one of the most common attacks for a multimedia cryptosystem [57–59]. In the paper, we take some methods to improve the resistance of the proposed encryption algorithm to known-plaintext and chosen-plaintext attacks. Firstly, the initial values of chaotic maps, the parameters used in the zigzag confusion process, the initial configurations C_0^{row} and C_0^{col} of the ECA are all generated by using SHA-512 hash function value of the plain image. In addition, the variance value of the original image is used to select the chaotic sequence group, and then the measurement matrix is generated by the sequence group, which further enhances the correlation between the algorithm and the plain image. Thus, when different plain images are encrypted, the corresponding key stream changes, too, and the different cipher images are obtained. Hence, the proposed system is robust or invulnerable to the known-plaintext and chosen-plaintext attacks.

5.6. Time complexity analysis

(1) **Effects of different compression ratios on the encryption and decryption process**

Regardless of the security considerations, encryption speed is also important, especially in real-time internet applications. In this paper, we analyze the encryption and decryption time of different size images at different compression ratios (CR), and the results are listed in Tables 7 and 8 ($w = 3$ and $e = 9$).

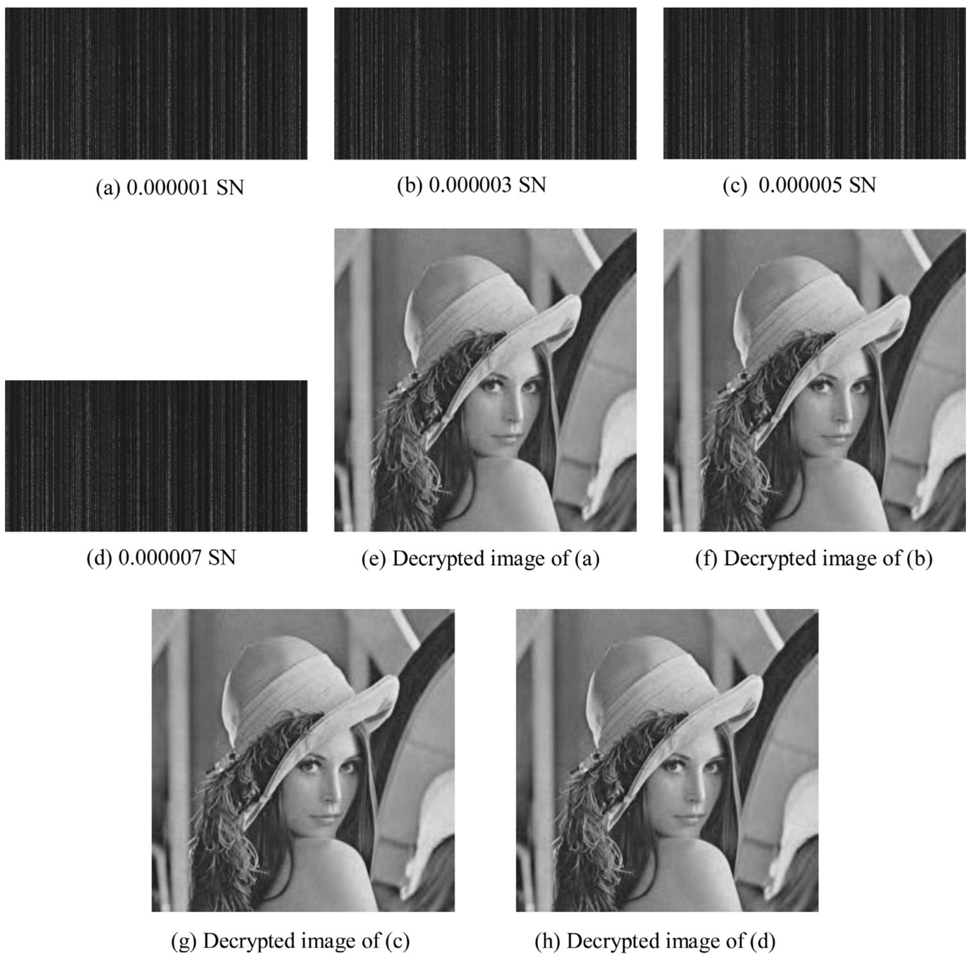

(a) 0.000001 SN
(b) 0.000003 SN
(c) 0.000005 SN
(d) 0.000007 SN
(e) Decrypted image of (a)
(f) Decrypted image of (b)
(g) Decrypted image of (c)
(h) Decrypted image of (d)

Fig. 20. Cipher and decrypted images under Speckle noise with different levels of noise.

From Tables 7 and 8, we can watch that (1) for the same plain image, the change of CR has a slight impact on the encryption time. When CR varies from 0.25 to 0.75, the encryption time of 512 × 512 images is changing from 0.75 s to 1.0 s, the encryption time for 256 × 256 images is about 0.46 s. (2) For the same original image, the decryption time under different compression ratios is different. And with the increasing of CR, the decryption time also increases. As shown in Table 8, when CR changes from 0.25 to 0.75, the time of decrypting Lena image is varying from 5.0 s to 22 s. The reason lays in solving the optimal solution in the reconstruction process, and the larger the measurement matrix, the more time needs. (3) As you can see, when CR = 0.25 and the image sizes vary from 256 × 256 to 512 × 512, the encryption time is varying from 0.45 s to 1 s, but the decryption time is changing from 1.1 s to 5.1 s. Similarly, when CR = 0.5 or CR = 0.75, the 512 × 512 images take more time than 256 × 256. In short, the times of encryption and decryption process are both affected by the size of the image. The bigger the image, the more time it takes. (4) In

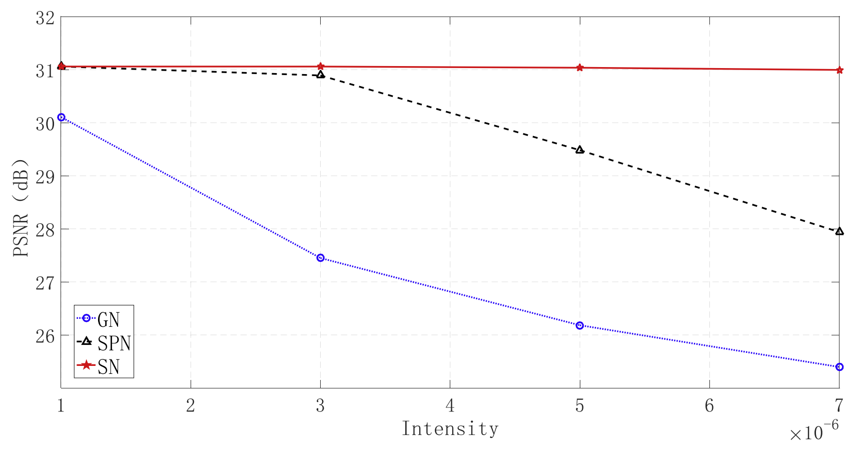


Fig. 21. PSNR between the decrypted image and the plain image with different noise.

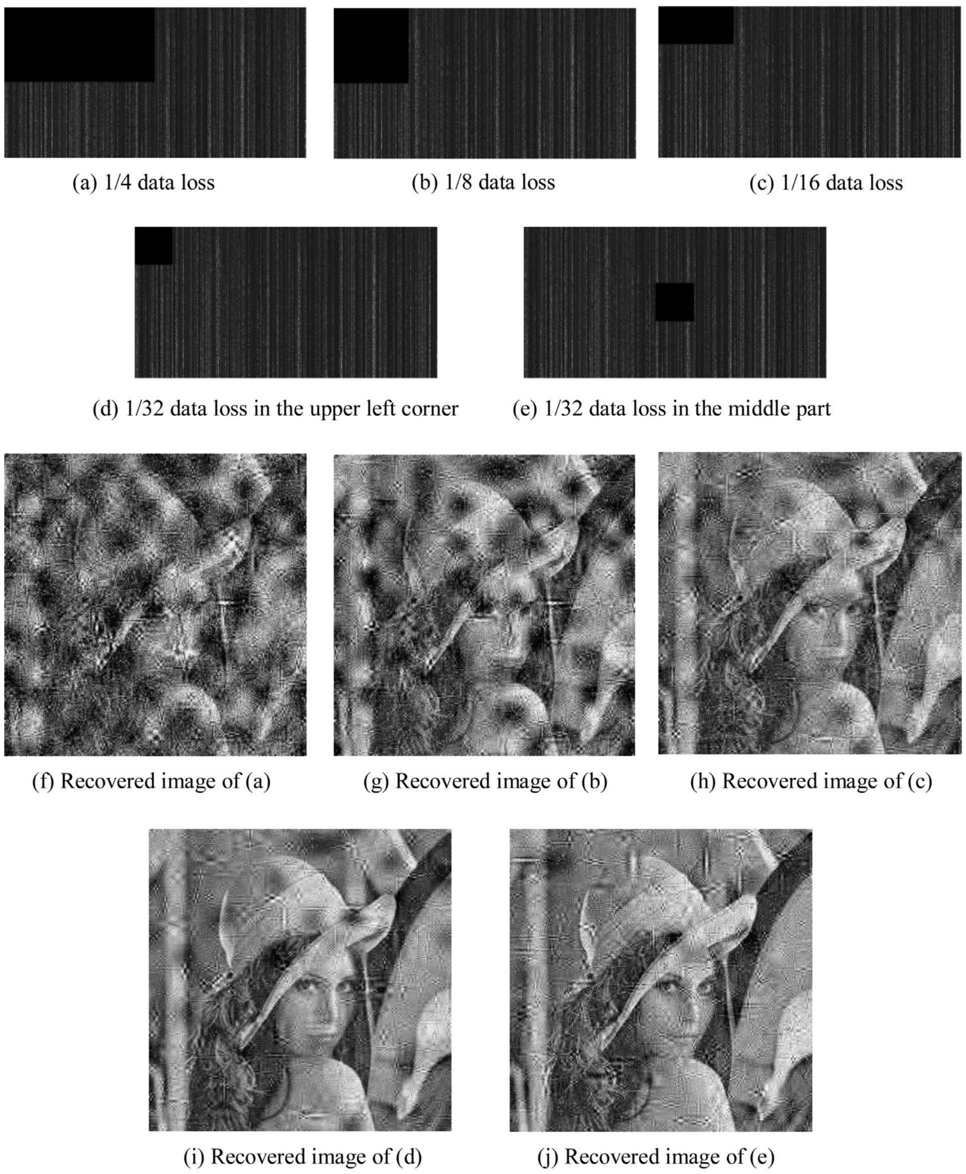

(a) 1/4 data loss
(b) 1/8 data loss
(c) 1/16 data loss
(d) 1/32 data loss in the upper left corner
(e) 1/32 data loss in the middle part
(f) Recovered image of (a)
(g) Recovered image of (b)
(h) Recovered image of (c)
(i) Recovered image of (d)
(j) Recovered image of (e)

Fig. 22. Resisting cropping attack results of the encryption scheme.

Table 7
Encryption time (second).

Images Size	finger 256 × 256	Baboon 256 × 256	Lena 512 × 512	brone 512 × 512	Peppers 512 × 512
CR = 0.25	0.4536	0.4607	0.7472	0.7921	0.9934
CR = 0.5	0.4605	0.4682	0.7742	0.8166	0.9925
CR = 0.75	0.4545	0.4852	0.7830	0.8225	1.0085

Table 8
Decryption time (second).

Images Size	finger 256 × 256	Baboon 256 × 256	Lena 512 × 512	brone 512 × 512	Peppers 512 × 512
CR = 0.25	1.1374	1.1413	5.0167	4.8581	5.0698
CR = 0.5	1.9577	2.0958	14.0266	13.6796	13.4131
CR = 0.75	2.8476	2.8635	21.8731	21.4699	22.0483

the simulation, in order to ensure the encryption quality of the proposed algorithm, we set the parameters $w = 3$ and $e = 9$. When CR = 0.25, the total time for encryption and decryption one image is about 1.58 s (for finger of 256 × 256), and 6.06 s (for Peppers of 512 × 512). When CR = 0.5, the total time is about 2.51 s (for finger of 256 × 256), and 14.41 s (for Peppers of 512 × 512), and when CR = 0.75, the total time is about 3.29 s (for finger of 256 × 256), and 23.04 s (for Peppers of 512 × 512). Thus, in practical application, we should carry out the comprehensive selection of encryption time and CR according to the actual situation.

Consequently, the encryption time of the proposed algorithm is short, while the greater the compression ratio, the longer the decryption time requires, therefore, the algorithm performs better in small compression ratio. The decryption speed of the proposed algorithm is slow, in the following work, block compression sensing or other reconstruction algorithms can be considered to improve the encryption and decryption speed.

(2) **Comparisons with other typical image encryption schemes**

Table 9 shows the comparison results of encryption time between our algorithm and other algorithms, and the size of the plain image is 256 × 256 and CR is changing from 0.25 to 0.75. As shown in Table 9, when the same plain image is encrypted, our scheme has the shortest running time than other algorithms

Table 9
The encryption time comparison results with other algorithms (second).

Images Size	Lena256 256 × 256	finger 256 × 256	Baboon 256 × 256	Cameraman 256 × 256	Peppers256 256 × 256
Proposed	0.58	0.46	0.47	0.71	0.66
Ref. [60]	2.25	2.18	2.55	2.66	2.76
Ref. [61]	3.23	3.44	3.53	3.51	3.68
Ref. [62]	11.12	11.30	11.45	11.19	12.13

in Refs. [60–62], and it can be applicable in real-time image encryption conditions.

6. Conclusions

In the paper, an image encryption algorithm based on the memristive chaotic system, elementary cellular automata and compressive sensing is introduced. First of all, the wavelet coefficients of the plain image are scrambled by the zigzag path and ECA. The novel scrambling method leads to a higher scrambling degree. Next, we take advantage of the CS theory to compress and encrypt the scrambled image, and obtain the final cipher image. For CS, a circular measurement matrix produced by a new kind of magnetic-controlled memristive chaotic system is adopted, which further decrease the energy consumption of data transmission. And what's more, the adoption of the memristive chaotic system generates the large key space, and makes the proposed encryption algorithm resist against the brute force attack. Besides, the SHA-512 hash value of the original image is utilized to get some parameters used in the encryption process, thus the algorithm has a high relationship with the plain image. And it can withstand the known-plaintext and chosen-plaintext attacks effectively.

Simulation results and performance analyses demonstrate that our proposed scrambling method based on zigzag path and ECA has significant scrambling performance, and our scheme has good encryption and decryption effect for different plain images. Furthermore, the presented algorithm has large key space, high key sensitivity, and good robustness to noise and occlusion attacks, and it can be effectively applied in the grayscale and color image secure communication. However, the decryption time of our encryption is a bit longer, and it may not work efficiently in real-time encryption fields, and thus in the future work, we intend to improve the processing speed and make it more proper for reliable and practical cryptographic applications.

Acknowledgments

All the authors are deeply grateful to the editors for smooth and fast handling of the manuscript. The authors would also like to thank the anonymous referees for their valuable suggestions to improve the quality of this paper. This work is supported by the National Natural Science Foundation of China (Grant nos. 41571417, U1604145 and U1404618), National Science Foundation of the United States (Grant nos. CNS-1253424 and ECCS-1202225), Science and Technology Foundation of Henan Province of China (Grant nos. 182102210027 and 172102210186), China Postdoctoral Science Foundation (Grant No. 2016M602235) and the Research Foundation of Henan University (Grant No. xxjc20140006).

References

[1] J.X. Chen, Z.L. Zhu, C. Fu, H. Yu, L.B. Zhang, An efficient image encryption scheme using gray code based permutation approach, Opt. Lasers Eng. 67 (2015) 191–204.
[2] Z.Y. Hua, Y.C. Zhou, Design of image cipher using block-based scrambling and image filtering, Inf. Sci. 396 (2017) 97–113.
[3] Y.C. Zhou, Z.Y. Hua, C.M. Pun, Cascade chaotic system with application, IEEE Trans. Cybern. 45 (9) (2015) 2001–2012.
[4] S.M. Pan, R.H. Wen, Z.H. Zhou, N.R. Zhou, Optical multi-image encryption scheme based on discrete cosine transform and nonlinear fractional Mellin transform, Multimedia Tools Appl. 76 (2) (2017) 2933–2953.
[5] W. Zamrani, E. Ahouzi, A. Lizana, J. Campos, M.J. Yzuel, Optical image encryption technique based on deterministic phase masks, Opt. Eng. 55 (10) (2016) 103108.
[6] X.Y. Wang, Y.Q. Zhang, X.M. Bao, A novel chaotic image encryption scheme using DNA sequence operations, Opt. Lasers Eng. 73 (2015) 53–61.
[7] T. Hu, Y. Liu, L.H. Gong, S.F. Guo, H.M. Yuan, Chaotic image cryptosystem using DNA deletion and DNA insertion, Signal Process. 134 (2017) 234–243.
[8] R. Guesmi, M.A.B. Farah, A. Kachouri, M. Samet, A novel chaos-based image encryption using DNA sequence operation and Secure Hash Algorithm SHA-2, Nonlinear Dyn. 83 (3) (2015) 1123–1136.
[9] Y.Q. Zhang, X.Y. Wang, J. Liu, Z.L. Chi, An image encryption scheme based on the MLNCML system using DNA sequences, Opt. Lasers Eng. 82 (2016) 95–13.
[10] S. Kumar, R.K. Sharma, Securing color images using Two-square cipher associated with Arnold map, Multimedia Tools Appl. 76 (6) (2017) 8757–8779.
[11] H. Zhu, C. Zhao, X. Zhang, L. Yang, An image encryption scheme using generalized Arnold map and affine cipher, Optik 125 (55) (2014) 6672–6677.
[12] G.Q. Hu, D. Xiao, Y.S. Zhang, An efficient chaotic image cipher with dynamic lookup table driven bit-level permutation strategy, Nonlinear Dyn. 87 (2) (2017) 1359–1375.
[13] Y.P. Li, C.H. Wang, H. Chen, A hyper-chaos-based image encryption algorithm using pixel-level permutation and bit-level permutation, Opt. Lasers Eng. 90 (2017) 238–246.
[14] X.L. Chai, An image encryption algorithm based on bit level Brownian motion and new chaotic systems, Multimedia Tools Appl. 76 (1) (2017) 1159–1175.
[15] C.Q. Li, D.D. Lin, J.H. Lu, Cryptanalyzing an image-scrambling encryption algorithm of pixel bits, IEEE. Multimedia 24 (2017) 64–71.
[16] D.L. Donoho, Compressive sensing, IEEE Trans. Inf. Theory 52 (2006) 1289–1360.
[17] E.J. Candes, J. Romberg, T. Tao, Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information, IEEE Trans. Inf. Theory 52 (2) (2006) 489–509.
[18] Y.S. Zhang, Study on the Coordination of Data Compression and Encryption, Chongqing University, 2014.
[19] D. Xiao, M.M. Deng, X.Y. Zhu, A reversible image authentication scheme based on compressive sensing, Multimedia Tools Appl. 74 (18) (2015) 7729–7752.
[20] N. Rawat, B. Kim, R. Kumar, Fast digital image encryption based on compressive sensing using structurally random matrices and Arnold transform technique, Optik 127 (4) (2016) 2282–2286.
[21] D. Xiao, L. Wang, T. Xiang, Y. Wang, Multi-focus image fusion and robust encryption algorithm based on compressive sensing, Opt. Laser Technol. 91 (2017) 212–225.
[22] X.Y. Wang, C.M. Liu, D.H. Xu, C.X. Liu, Image encryption scheme using chaos and simulated annealing algorithm, Nonlinear Dyn. 84 (3) (2016) 1417–1429.
[23] Z.Y. Hua, S. Yi, Y.C. Zhou, Medical image encryption using high-speed scrambling and pixel adaptive diffusion, Signal Process. 144 (2018) 134–144.
[24] R. Guesmi, B. Farah, A. Kachouri, M. Samet, Hash key-based image encryption using crossover operator and chaos, Multimedia Tools Appl. 75 (8) (2016) 4753–4769.
[25] X.W. Li, C.Q. Li, I.K. Lee, Chaotic image encryption using pseudo-random masks and pixel mapping, Signal Process. 125 (2016) 48–63.
[26] Y.S. Zhang, J.T. Zhou, F. Chen, L.Y. Zhang, K.W. Wong, X. He, D. Xiao, Embedding cryptographic features in compressive sensing, Neurocomputing 205 (2016) 472–480.
[27] N.R. Zhou, H.L. Li, D. Wang, S.M. Pan, Z.H. Zhou, Image compression and encryption scheme based on 2D compressive sensing and fractional Mellin transform, Opt. Commun. 343 (2015) 10–21.
[28] D. Xiao, H.K. Cai, H.Y. Zheng, A joint image encryption and watermarking algorithm based on compressive sensing and chaotic map, Chin. Phys. B 24 (6) (2015) 198–206.
[29] N.R. Zhou, J.P. Yang, C.F. Tan, S.M. Pan, Z.H. Zhou, Double-image encryption scheme combining DWT-based compressive sensing with discrete fractional random transform, Opt. Commun. 354 (2015) 112–121.
[30] W.H. Liu, K.H. Sun, C.X. Zhu, A fast image encryption algorithm based on chaotic map, Opt. Lasers Eng. 84 (2016) 26–36.
[31] N.R. Zhou, S.M. Pan, S. Chen, Z.H. Zhou, Image compression–encryption scheme based on hyper-chaotic system and 2D compressive sensing, Opt. Laser Technol. 82 (2016) 121–133.
[32] X.J. Tong, M. Zhang, Z. Wang, J. Ma, A joint color image encryption and compression scheme based on hyper-chaotic system, Nonlinear Dyn. 84 (4) (2016) 2333–2356.
[33] H. Liu, D. Xiao, R. Zhang, S. Bai, Robust and hierarchical watermarking of encrypted images based on compressive sensing, Signal Process. Image Commun. 45 (C) (2016) 41–51.
[34] Y. Zhang, B. Xu, N.R. Zhou, A novel image compression-encryption hybrid algorithm based on the analysis sparse representation, Opt. Commun. 392 (2017) 223–233.
[35] A. Souyah, K.M. Faraoun, An image encryption scheme combining chaos-memory cellular automata and weighted histogram, Nonlinear Dyn. 86 (1) (2016) 639–653.
[36] A.L.A. Dalhoum, A. Madain, H. Hiary, Digital image scrambling based on elementary cellular automata, Multimedia Tools Appl. 75 (24) (2016) 17019–17034.

[37] A.A. Dalhoum, B.A. Mahafzah, A.A. Awwad, I. Aldamari, A. Ortega, M. Alfonseca, Digital image scrambling using 2D cellular automata, IEEE. Multimedia 19 (4) (2012) 28–36.
[38] 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.
[39] T.H. Chen, M. Zhang, J.H. Wu, C. Yuen, Y. Tong, Image encryption and compression based on kronecker compressed sensing and elementary cellular automata scrambling, Opt. Laser Technol. 84 (2016) 118–133.
[40] S.N. George, N. Augustine, D.P. Pattathil, Audio security through compressive sampling and cellular automata, Multimedia Tools Appl. 74 (23) (2015) 10393–10417.
[41] F.H. Min, Z.L. Wang, E.R. Wang, Y. Cao, New memristor chaotic circuit and its application to image encryption, J. Electron. Inf. Technol. 38 (10) (2016) 2681–2688.
[42] C. Xu, C.Q. Li, J.H. Lv, S. Shu, On the network analysis of the state space of discrete dynamical systems, Int. J. Bifurc. Chaos 27 (4) (2017).
[43] Y.M. Ren, Y.N. Zhang, Y. Li, Advances and perspective on compressed sensing and application on image processing, Acta Autom. Sin. 40 (8) (2014) 1563–1575.
[44] H.P. Yin, Z.D. Liu, Y. Chai, X.G. Jiao, Survey of compressed sensing, Control Decis. 28 (10) (2013) 1441–1445.
[45] J.B. Guo, R. Wang, Construction of a circulant compressive measurement matrix based on chaotic sequence and RIPless theory, Acta Phys. Sin. 63 (19) (2014) 373–382.
[46] X.L. Chai, Z.H. Gan, Y.R. Chen, Y.S. Zhang, A visually secure image encryption scheme based on compressive sensing, Signal Process. 134 (2017) 35–51.
[47] N.R. Zhou, A.D. Zhang, F. Zheng, L.H. Gong, Novel image compression–encryption hybrid algorithm based on key-controlled measurement matrix in compressive sensing, Opt. Laser Technol. 62 (10) (2014) 152–160.
[48] H. Liu, A. Kadir, Asymmetric color image encryption scheme using 2D discrete-time map, Signal Process. 113 (2015) 104–112.
[49] B. Norouzi, S.M. Seyedzadeh, S. Mirzakuchaki, M.R. Mosavi, A novel image encryption based on row-column, masking and main diffusion processes with hyper chaos, Multimedia Tools Appl. 74 (3) (2015) 781–811.
[50] G. Alvarez, S. Li, Some basic cryptographic requirements for chaos-based cryptosystems, Int. J. Bifurc. Chaos 16 (8) (2006) 2129–2151.
[51] X.Y. Wang, K. Guo, A new image alternate encryption algorithm based on chaotic map, Nonlinear Dyn. 76 (4) (2014) 1943–1950.
[52] A. Bakhshandeh, Z. Eslami, An authenticated image encryption scheme based on chaotic maps and memory cellular automata, Opt. Lasers Eng. 51 (6) (2013) 665–673.
[53] S.N. George, D.P. Pattathil, A secure LFSR based random measurement matrix for compressive sensing, Sensing Imaging 15 (1) (2015) 1–29.
[54] R. Enayatifar, A.H. Abdullah, I.F. Isnin, A. Altameem, M. Lee, Image encryption using a synchronous permutation-diffusion technique, Opt. Lasers Eng. 90 (2017) 146–154.
[55] Chanil Pak, L.L. Huang, A new color image encryption using combination of the 1D chaotic map, Signal Process. 138 (2017) 129–137.
[56] S. Yi, Y.C. Zhou, Binary-block embedding for reversible data hiding in encrypted images, Signal Process. 133 (2017) 40–51.
[57] L.L. Yao, C.J. Yuan, J.J. Qiang, S.T. Feng, S.P. Nie, Asymmetric image encryption method based on gyrator transform and vector operation, Acta Phys. Sin. 65 (21) (2016) 214203.
[58] Y. Xie Eric, C.Q. Li, S.M. Yu, J.H. Lu, On the cryptanalysis of Fridrich's chaotic image encryption scheme, Signal Process. 132 (2017) 150–154.
[59] C.Q. Li, T. Xie, Q. Liu, G. Cheng, Cryptanalyzing image encryption using chaotic logistic map, Nonlinear Dyn. 78 (2014) 1545–1551.
[60] J. Ahmad, S.O. Hwang, A secure image encryption scheme based on chaotic maps and affine transformation, Multimedia Tools Appl. 75 (21) (2015) 13951–13976.
[61] F. Ahmed, A. Anees, V.U. Abbas, M.Y. Siyal, A noisy channel tolerant image encryption scheme, Wireless Pers. Commun. 77 (4) (2014) 2771–2791.
[62] A. Anees, A.M. Siddiqui, F. Ahmed, Chaotic substitution for highly autocorrelated data in encryption algorithm, Commun. Nonlinear Sci. Numer. Simul. 19 (9) (2014) 3106–3118.