Optics & Laser Technology 163 (2023) 109345

Contents lists available at ScienceDirect

Optics and Laser Technology

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

Full length article

A visually secure image encryption scheme based on adaptive block compressed sensing and non-negative matrix factorization

Yuandi Shi [a], Rongrong Chen [a], Donglin Liu [b], Bin Wang [a,*]

[a] The Key Laboratory of Advanced Design and Intelligent Computing, Ministry of Education, School of Software Engineering, Dalian University, Dalian, Liaoning 116622, China

[b] Jining Central Branch, The People's Bank of China, Jining, Shandong 272008, China

ARTICLE INFO

Keywords:
ABCS
Image encryption
NMF
Tetrolet transform

ABSTRACT

The traditional block compressive sensing theory cannot make full use of the difference of image block information characteristics for sampling, and the measurement matrix cannot fully measure the image according to the image block information characteristics. To solve these problems, we design an adaptive block compressed sensing (ABCS) algorithm, and we propose a visually secure image encryption scheme based on ABCS and non-negative matrix factorization (NMF). First, the plain image is decomposed by Tetrolet transform. Then, we perform matrix shuffling to optimize the sparsity of the sparse matrix. Next, adaptive sampling and measurement are performed according to the information distribution characteristics between the image blocks to compress the image to a quarter of its original size. Last, we design a scrambling–diffusion framework based on tetrominoes to obtain the secret image, which is then embedded into the carrier image by means of NMF to obtain a visually secure cipher image. The experimental results show that the proposed scheme can consider the security of image information transmission and the integrity of reception, and has good resistance to various attacks.

1. Introduction

With the increasing amount of data transmission, information security has received increasing research attention. As one of the main carriers of information, images are widely used in medicine, military, finance, art, and other fields. However, not all image information can be disclosed. In many cases, for confidentiality reasons, the sender wants only the receiver to obtain the complete information. Therefore, the image needs to be encrypted to achieve the purpose of information protection.

The technologies involved in the field of image encryption are becoming increasingly extensive, including aspects of mathematics, biology, physics, and many other fields, which have a direct or indirect impact on the development of image encryption. A number of encryption techniques, including magic square [1,2], S-box[3,4], DNA coding [5–8], cellular automata[9–11], elliptic curve [12],and quantum theory [13], have been proposed, and these have substantially reduced the risk of encrypted images being cracked. However, some of these algorithms have a single encryption framework and a simple encryption process, which makes them vulnerable to being cracked [14,15]. Due to its high sensitivity to the initial values, the chaotic system is highly suitable as the random sequence generator of an encryption system. Plain information is directly or indirectly used for the iteration of the chaotic system, which can effectively reduce the risk of the cipher image being cracked. Therefore, how to use the information of the plain image throughout the entire encryption process and expand the function of the plain information as much as possible, instead of it being limited to a single process such as key generation, will be key factors for improving the security of encryption schemes. They are also two of the main research focuses of this study.

In recent years, image encryption technology has become increasingly mature. Especially when combined with compression technology, the result greatly reduces the transmission and storage space requirements and relieves the pressure of signal transmission while ensuring information security [16–20]. It was in 2004 that compressed sensing was first proposed as a new sampling theory. The emergence of this theory has enabled sampling and compression of the signal to be performed at a rate much lower than the Nyquist sampling rate, and accurate reconstruction of the signal can be achieved by optimizing the calculation method, which solves the problem of having a lot of redundant information in the signal[21]. The two main processes of obtaining the measurements of a signal, namely through the

* Corresponding author.
E-mail address: wangbin@dlu.edu.cn (B. Wang).

https://doi.org/10.1016/j.optlastec.2023.109345
Received 7 September 2022; Received in revised form 27 January 2023; Accepted 5 March 2023
Available online 20 March 2023

measurement matrix and reconstruction of the measurements to restore the original signal, are highly consistent with the process of encryption and decryption in a cryptosystem. Therefore, by embedding cryptographic features in the process of compressed sensing theory, protection of information can be achieved[22]. Image encryption schemes based on block compressed sensing have been one of the most popular research directions in recent years. Since this type of scheme can simplify the operation object from the entire image matrix to a single image block matrix, it substantially lowers the required storage space and computational complexity and thus substantially improves its practicability. However, the inability to determine the appropriate sparsity of the image, the use of a fixed sampling rate, and the inability to construct a suitable measurement matrix cause encryption schemes based on the traditional block compressive sensing theory to be unable to achieve good results. Therefore, these three problems need to be solved urgently in this field, and many researchers have proposed corresponding solutions.

First, the difference in sparsity caused by different spatial features between the blocks of the image affects the reconstruction of the image. Zhu et al. [19] scrambled the block coefficient matrix after discrete cosine transform (DCT) with the help of the coefficient random permutation (CRP) strategy and achieved good results, but this method has strong randomness and cannot guarantee that the optimal result can be obtained after each scrambling. In order to distribute the non-zero values between the image blocks as evenly as possible to improve the sampling efficiency, Zhang et al. [23] proposed an optimized matrix scrambling algorithm, which makes the scrambled image blocks have a more balanced sparsity distribution, so that the reconstruction of the image is significantly improved.

Second, ignoring the characteristics of each block and uniformly sampling image blocks with a fixed number of samples is another area where the traditional block-based compressed sensing theory needs to be improved. The scheme proposed in [18] only compressed the detail components. The horizontal decomposition coefficient LH and the diagonal decomposition coefficient HH use a smaller sampling rate, whereas the vertical decomposition coefficient HL adopts a larger sampling rate, and this improves the reconstruction quality of the decrypted image. However, since the sampling rate is artificially set, the information of the image itself is not considered, which leads to certain limitations. Monika et al. [24] proposed the theory of adaptive block compressive sensing using energy content, which provides a new idea for optimizing the sampling process by calculating the energy values in the blocks to assign different measurement values to each block.

Finally, the measurement matrix is a crucial part of compressed sensing theory, and constructing a suitable measurement matrix can greatly improve the quality of image reconstruction. With the goal of improving the incoherence of the matrix, Chai et al. [17] used singular value decomposition(SVD) to optimize the measurement matrix, which greatly improved the reconstruction quality of the decrypted image. Similarly, in [25], a measurement matrix designed based on the MGS algorithm was proposed, which reduced the reconstruction error. Gan et al. [26] proposed a measurement matrix optimization algorithm based on adaptive step size and used the generated measurement matrix to measure the high-frequency components, which achieved a good balance between the compression performance of the cipher image and the reconstruction quality of the decrypted image, but the optimization process had high complexity. In this study, with the help of the statistical features of plain images, we follow the principle of making the small coefficients in the sparse signal closer to zero and the large coefficients larger to simply and efficiently optimize the construction process of the measurement matrix.

In addition, in the face of various attack methods of attackers, pure image encryption schemes cannot prevent the risk of being cracked. Therefore, some encryption schemes with visual security as the protection method have been proposed [17,19,20,22,27–30]. In these, the cipher image containing the secret information is visually only a natural image, which will not attract the attention of the attacker and thus reduce the risk of being cracked. Zhu et al. [19] used the SVD technique to improve the embedding capacity by changing all elements of the diagonal matrix and embedded the segmented secret image information into the carrier image. Chai et al. [16] enhanced the potential key space by replacing the two least significant bits of the carrier image coefficients with the secret image information, thus making the information of the secret image independent of the content of the carrier image. In [20], Wang et al. obtained three components of red, green, and blue by performing three-dimensional (3D) DCT on the color carrier image and then embedded three secret images into the three components, which greatly improved the encryption efficiency. In order to not only satisfy the invisibility and robustness of the image but also to improve the security of image transmission and storage, Chai et al. [28] proposed a block HD embedding method, which embeds the secret image corresponding to the RGB components into the Hessenberg matrix decomposed by the high frequency coefficients of the RGB components of the carrier image, and this approach achieved good results. Considering the information-hiding process of the above schemes, as long as a signal transformation can be formed, there is the possibility of hiding part of the information in the transformation space. Non-negative matrix factorization (NMF) is a widely used data processing tool in the field of signal processing. It leads to sparseness of the corresponding description to an extent because of its non-negativity limitation, which makes the description of the data convenient and reasonable. At the same time, it can also restrain the influence of external interference factors on data feature extraction to an extent. Therefore, it has the advantages of simplicity in implementation, interpretability in decomposition forms and results, and small storage space requirement, thus providing a new idea for information hiding.

In order to solve the shortcomings of many existing experimental schemes in the security of encrypted images and the recovery quality of decrypted images, we propose an image encryption scheme based on adaptive block compressed sensing and NMF. First, Tetrolet transform is used as the transform domain [31], and it is combined with optimized matrix scrambling to make the sparsity of the image more balanced [23]. Second, the image is sampled and measured according to the theory of adaptive compressed sensing. After that, scrambling and diffusion algorithms are designed, and the image is encrypted with the help of the tetrominoes generated in the Tetrolet transformation process. Finally, the encrypted image is embedded into the carrier image using NMF. The main contributions of this study are as follows:

(1) Instead of setting a fixed number of samples for each block of the image, the sampling process is adapted to the image information, and an adaptive block sampling strategy based on region energy is proposed, so that the number of samples in the block can be allocated more reasonably.
(2) The construction process of the measurement matrix is optimized. We increase the proportion of large coefficients and reduce the proportion of coefficients according to the contribution of the image coefficients to signal recovery in order to improve the recovery of the image information.
(3) Based on the tetrominoes in the construction process of the Tetrolet transform, combined with related mathematical problems, a pixel-level scrambling method based on the domino tiling problem, a bit-level scrambling method based on the domino knocking problem, and a diffusion method based on tetrominoes are designed, which closely combines the plain information with the encryption process and improves the anti-attack ability.
(4) NMF is introduced to embed the secret image into the carrier image, which can not only retain the image information to a large extent but also suppress the influence of external changes to an extent.

The rest of the paper is organized as follows. Related works and

contributions are introduced in Section 2. The description of our scheme is provided in Section 3. Simulation results and discussion are presented in Section 4. Finally, conclusions are drawn in Section 5.

2. Related works and contributions

2.1. Related works and methods

2.1.1. Tetrolet transform

Jens Krommweh[31] proposed a new adaptive Haar wavelet transform—Tetrolet transform. Its construction process is similar to Wedgelet transform, and the Haar function is applied to the edge part. The process of sparsification first divides the low-frequency coefficient of the image into several 4×4 blocks and then determines a tetromino in each block that is suitable for the image's geometric structure and then divides each block into four regions by Tetrolet transformation. The process is iterated to achieve the purpose of sparse approximation. Compared with the traditional Haar wavelet transform, Tetrolet transform can express the image considerably sparsely, which is appropriate to the process of image compression. Compared with discrete wavelet transform (DWT), which preferentially selects horizontal and vertical directions, Tetrolet transform strives to achieve the best effect on images containing geometric structures in other directions. Therefore, this study constructs a novel compression-encryption framework based on Tetrolet transform.

2.1.2. NMF

The concept of NMF was proposed by Lee and Seung [32]. The basic idea is to define an appropriate cost function and find the factorization of the non-negative matrix as accurately as possible by minimizing the cost function. Its mathematical realization form is as follows. Given an $m \times n$ non-negative matrix V, it can be decomposed into the product of two matrices, denoted as W and H.

$$V_{m\times n} \approx W_{m\times r} \times H_{r\times n} \tag{1}$$

where $W_{m\times r}$ is the basis matrix; $H_{r\times n}$ is the coefficient matrix; and the rank $r \leqslant \min(m,n)$. If $V = [V_1, V_2, \ldots, V_n]$, $H = [H_1, H_2, \ldots, H_n]$, then

$$V_i = W \times H_i \tag{2}$$

where V_i is the ith column of V, and H_i is the ith column of H.

2.1.3. Optimal permutation matrix generating algorithm

In order to reduce the maximum block sparsity of the image matrix and obtain better image reconstruction, Zhang et al. [23] proposed an algorithm to generate an optimized permutation matrix. After scrambling the image matrix, the non-zero values in the matrix are distributed evenly to reduce the maximum block sparsity. After thinning the image, we apply this algorithm to the sparse matrix to reduce the sparsity differences between the image blocks.

2.1.4. Six-dimensional hyperchaotic system

Yang et al. [33] proposed a new six-dimensional (6D) hyperchaotic system through a sequential design of linearly controlling a 3D chaotic system. Its expression is as follows:

$$\begin{cases} \dot{x}_1 = h(x_2 - x_1) + x_4 \\ \dot{x}_2 = -fx_2 - x_1x_3 + x_6 \\ \dot{x}_3 = -l + x_1x_2 \\ \dot{x}_4 = -rx_2 - x_5 \\ \dot{x}_5 = kx_2 + x_4 \\ \dot{x}_6 = gx_1 + mx_2 \end{cases} \tag{3}$$

where h, l, f, r, k, g and m are parameters. When the parameters are taken as $(h, l, f, k, g, m, r) = (10, 100, 2.7, 2, -3, 1, 1)$, the system has no equilibrium point, but there are hidden hyperchaotic attractors and four positive Lyapunov exponents.

2.1.5. Hybrid chaotic map

By combining the logistic map, tent map, and sin map, M. A. Ben Farah et al. [4] proposed a new hybrid chaotic map:

$$H = \begin{cases} \left(\left(r^{10}\right)\frac{r}{4}\sin\left(\pi\frac{r^2}{2}x_i\right)(1-x_i)\right) \ mod\ 1, rx_i(1-x_i) \leqslant \frac{1}{2} \\ \left(\left(r^{10}\right)\frac{r}{4}\sin\left(\pi\frac{r}{2}(1-rx_i(1-x_i))\right)\right) \ mod\ 1, rx_i(1-x_i) > \frac{1}{2} \end{cases} \tag{4}$$

where r is the control parameter of the hybrid chaotic map whose value range is $r \in [1.4, 4]$.

2.1.6. $2 \times n$ Domino tiling problem

The $2 \times n$ domino tiling problem is to calculate how many ways there are to completely cover a $2 \times n$ rectangle with 2×1 or 1×2 dominoes. Fig. 1 is the situation when $n = 1, 2, 3, 4, 5$, and T_2 is the corresponding tile number. It has been proved that without distinguishing whether the dominoes are horizontal or vertical, the problem can be generalized to the following generating function:

$$T_2 = \frac{1}{1 - z - z^2} \tag{5}$$

where z represents the horizontal or vertical domino. Considering that the dominoes are horizontal or vertical, the mathematical form, Eq. (6), can be obtained:

$$\begin{aligned} T_2 &= \frac{1}{1-a-b^2} = 1 + \left(a+b^2\right) + \left(a+b^2\right)^2 + \left(a+b^2\right)^3 + \cdots \\ &= \sum_{k\geqslant 0}\left(a+b^2\right)^k = \sum_{j,k\geqslant 0}\binom{k}{j}a^j b^{2k-2j} = \sum_{j,m\geqslant 0}\binom{j+m}{j}a^j b^{2m} s \end{aligned} \tag{6}$$

where a and b represent vertical and horizontal dominoes, respectively. This formula shows that $\binom{j+m}{j}$ is the number of methods for laying a $2 \times (j + 2m)$ rectangle with j vertical dominoes and $2m$ horizontal dominoes.

2.1.7. $3 \times n$ Domino tiling problem

The $3 \times n$ domino tiling problem is to calculate how many ways there are to completely cover a $3 \times n$ rectangle with 2×1 or 1×2 dominoes. In particular, when n is odd, the problem has no solution. Fig. 2 is the situation when $n = 2, 4$, and T_3 is the corresponding tile number. It has been proved that without distinguishing whether the dominoes are horizontal or vertical, the problem can be generalized to the following generating function:

$$T_3 = \frac{1 - z^3}{1 - 4z^3 + z^6} \tag{7}$$

where z represents the horizontal or vertical domino. Considering that

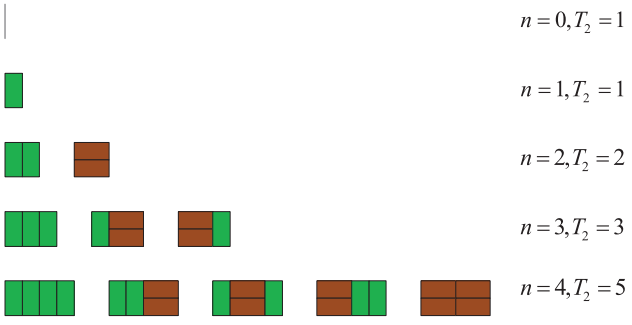

Fig. 1. $2 \times n$ domino tiling problem.

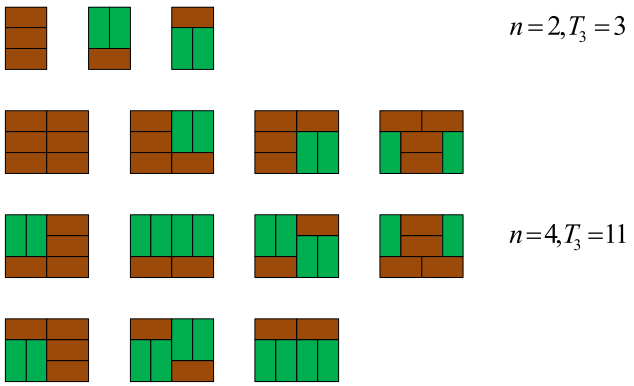

Fig. 2. $3 \times n$ domino tiling problem.

the dominoes are horizontal or vertical, the mathematical form, Eq. (6), can be obtained:

$$T_3 = \frac{1-b^3}{(1-b^3)^2 - 2a^2b} = \frac{1}{1-b^3} + \frac{2a^2b}{(1-b^3)^3} + \frac{4a^4b^2}{(1-b^3)^5} + \frac{8a^6b^3}{(1-b^3)^7} + \cdots = \sum_{k\geqslant 0} \frac{2^k a^{2k} b^k}{(1-b^3)^{2k+1}} = \sum_{k,m\geqslant 0} \binom{m+2k}{m} 2^k a^{2k} b^{k+3m} \tag{8}$$

where a and b represent vertical and horizontal dominoes, respectively. This formula shows that $\binom{m+2k}{m}2^k$ is the number of methods for laying a $3 \times (3k+3m)$ rectangle with $2k$ vertical dominoes and $k+3m$ horizontal dominoes.

2.2. Our contributions for the new scheme

2.2.1. Adaptive block sampling strategy based on region energy

Considering the difference in the information distribution characteristics between image blocks, to sample the image more effectively, we propose an adaptive block sampling strategy based on image region energy. The specific steps of the algorithm are as follows.

Step 1: Set the block area size B, the image sampling rate SR, and the weight matrix W. Considering the computational complexity and the correlation of the adjacent pixels of the image, we set the size of B to be 4, 8, and 16, and then, Eq. (9) can be obtained.

$$\begin{cases} W = [1,2,2,1], \ if \ B = 4 \\ W = [1,1,2,2,2,2,1,1], \ if \ B = 8 \\ W = [1,1,1,1,2,2,2,2,2,2,2,2,1,1,1,1], \ if \ B = 16 \end{cases} \tag{9}$$

Step 2: Let the length and width of the image be m and n, respectively. Calculate the number of image blocks N and the number of samples SF according to Eqs. (10) and (11).

$$N = m \times n / B^2 \tag{10}$$

$$SF = m \times n \times SR \tag{11}$$

Step 3: Block the image and calculate the energy E_i of the pixels within the block area.

$$\begin{cases} E_i = sum(W^{'} \times W_{.} \times X_i./36), \ if \ B = 4 \\ E_i = sum(W^{'} \times W_{.} \times X_i./144), \ if \ B = 8 \\ E_i = sum(W^{'} \times W_{.} \times X_i./576), \ if \ B = 16 \end{cases} \tag{12}$$

where X_i is the image block.

Step 4: Calculate the energy percentage PE_i for each image block.

$$PE_i = abs(E_i) / \sum_{i=1}^{n} abs(E_i) \tag{13}$$

Step 5: Calculate the sampling number AM_i of each image block.

$$AM_i = PE_i \times SF, i = 1,2,\ldots,N \tag{14}$$

The number of block samples at the corresponding position forms the sampling matrix AM of the entire image:

$$AM = \begin{pmatrix} AM_1 & \cdots & AM_{m/B} \\ \vdots & \ddots & \vdots \\ AM_{n/B} & \cdots & AM_N \end{pmatrix} \tag{15}$$

Considering that the number of samples may not be an integer, the actual image block sampling number $AM_i^{'}$ and the image sampling matrix $AM^{'}$ are obtained by the following rounding operations:

$$AM_i^{'} = round(PE_i \times SF), i = 1,2,\ldots,N \tag{16}$$

$$AM^{'} = \begin{pmatrix} AM_1^{'} & \cdots & AM_{m/B}^{'} \\ \vdots & \ddots & \vdots \\ AM_{n/B}^{'} & \cdots & AM_N^{'} \end{pmatrix} \tag{17}$$

Sort $AM^{'}$ in ascending order to obtain the index sequence ind_AM:

$$[\sim, ind\_AM] = sort(AM^{'}) \tag{18}$$

Adjust the largest element in $AM^{'}$ so that the total number of samples remains the same:

$$AM^{'}(ind\_AM(end)) = AM^{'}(ind\_AM(end)) + round(sum(AM) - sum(AM^{'})) \tag{19}$$

Note that the largest element in $AM^{'}$ is $maxAM$, and this parameter is used for the iterative process of the chaotic system.

2.2.2. Optimized measurement matrix

Considering the difference in the information features between image blocks, it is necessary to optimize the measurement matrix generated by the chaotic sequence.

Step 1: Set the size of the block area B; divide the image into blocks; calculate the sum of the absolute values of the pixel values in the blocks to obtain the matrix Sum; and arrange the elements in Sum in descending order to obtain the sorting index value, which is recorded as Pos. Adjust the initial measurement matrix $chaosMat$ to an $M \times N$ matrix, where $N = m \times n/B^2$; m and n are the rows and columns of the image matrix; $M = Num/N$; and Num is the number of elements in the matrix $chaosMat$.

Step 2: Calculate the sum of the mean and median of the absolute value in the column vector of the measurement matrix and assign the corresponding weight to obtain the sequence $Sum1$:

$$Sum1(i) = w_1 \times sum(abs(chaosMat(:,i))) + w_2 \times median(abs(chaosMat(:,i))) \tag{20}$$

In this experiment, we take $w_1 = 0.5, w_2 = 0.5$.

Step 3: Sort $Sum1$ in descending order to obtain the index sequence $Pos1$, and rearrange the column vector of the measurement matrix $chaosMat$ of the corresponding size according to the size of the pixel sum in the image blocks to obtain Φ:

$$\Phi(:, Pos(i)) = chaosMat(:, Pos1(i)) \tag{21}$$

2.2.3. Mask division

According to the theory of Tetrolet transform proposed by Jens Krommweh [30], 16 tilings with different directions are obtained, as shown in Fig. 3. In the case of the 16 tilings as masks, we further divide them into 1×2 or 2×1 puzzles, as shown in Fig. 4. The first and sixteenth tilings have two division forms. The symbol _ is used to represent the 1×2 horizontal puzzle; the symbol | is used to represent the 2×1 vertical puzzle; and then, the coding of puzzle can be obtained, as shown in Table 1.

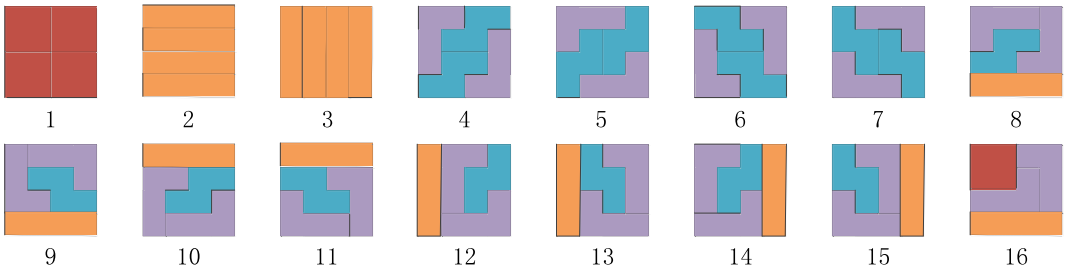

Fig. 3. Sixteen tilings with different directions.

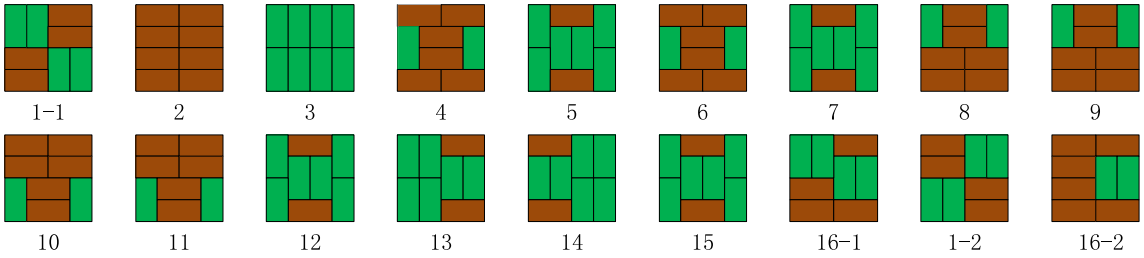

Fig. 4. Mask division.

Table 1
Puzzle coding.

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

2.2.4. Pixel-level scrambling: A scrambling method based on the domino tiling problem

In this round of scrambling, four random sequences $x1$, $y1$, $z1$, $u1$ are used. First, perform the modulo operation for $x1$, $y1$ to obtain $x1'$, $y1'$:

$$x1' = mod\ (x1, 16) + 1, y1' = mod\ (y1, 16) + 1 \tag{22}$$

Take each element in sequence $x1'$ as the number of the tiling in Fig. 3; perform mask division according to Fig. 4; and use the coding rule of Table 1 to encode the result. Then, a character sequence *dominoesSeq* containing _ and | can be obtained. Use $y1'$ as the index sequence to index each character sequence to obtain a flag bit 0 or 1, and finally a flag bit matrix *Flags* can be obtained:

$$\begin{cases} Flags(i) = 1,\ if\ dominoesSeq(y1'(i)) = '|' \\ Flags(i) = 0,\ if\ dominoesSeq(y1'(i)) = '\_' \end{cases} \tag{23}$$

According to the relevant theory of the domino tiling problem described in Sections 2.1.6 and 2.1.7, two sequences of *Num*2 and *Num*3 are obtained, where *Num*2 and *Num*3 are the sequences composed of the method numbers of the corresponding $2 \times n$ and $3 \times n$ domino tiling problems when n takes 0–255. Next, initialize the following matrices: *Initial*, *Full*, *Finish*, *Flags*1, *S*, where *Initial* represents the initial value of the image matrix; *Full* is the target value to be achieved by each element in the image matrix; *Finish* is the result of scrambling the image matrix; and *Flags*1 is the flag bit matrix, which is used to record the scrambled element positions in the image matrix. *S* is the index matrix used to determine the next element position to be found by the current matrix element.

First, rearrange the sequence $z1$, $u1$ to obtain the matrices $z1'$, $u1'$, which have the same dimensions as the image matrix. Then, determine the length N of the rectangle in the domino tiling problem according to Eq. (24).

$$N = u1'(i,j) \tag{24}$$

The construction processes of the matrices *Full* and *S* are shown in Eqs. (25) and (26).

$$\begin{cases} n = 2, Full(i,j) = Num2(N+1);\ if\ mod\ (z1'(i,j), 2) = 0 \\ n = 3, Full(i,j) = Num3(N+1);\ else \end{cases} \tag{25}$$

$$\begin{cases} S(i,j) = z1'(i,j),\ if\ z1'(i,j)\ meets\ one\ \text{of the special cases} \\ S(i,j) = \text{dominoesTile}(n, N, z1'(i,j), Flags(i,j)),\ else \end{cases} \tag{26}$$

where dominoesTile is the function used to obtain the required horizontal or vertical dominoes for tiling a $n \times N$ rectangle when $z1'(i,j)$ horizontal or vertical dominoes are given according to Eq. (6) or (8). However, not all inputs can be solved, and nine special cases exist as follows:

$$\begin{cases} z1'(i,j) \times 2 > n \times N \\ n = 3 \& Flags(i,j) = 0 \& \left(N \times \frac{3}{2} - z1'(i,j) \right) mod\ 2 \neq 0 \\ n = 3 \& Flags(i,j) = 0 \& (z1'(i,j) \times 2 - N) \langle 0 \\ N < z1'(i,j) \\ n = 3 \& N\ mod\ 2 \neq 0 \\ n = 3 \& Flags(i,j) = 1 \& z1'(i,j)\ mod\ 2 \neq 0 \\ n = 3 \& Flags(i,j) = 0 \& (N - z1'(i,j))\ mod\ 2 \neq 0 \\ n = 2 \& Flags(i,j) = 0 \& z1'(i,j)\ mod\ 2 \neq 0 \\ n = 2 \& Flags(i,j) = 1 \& (N - z1'(i,j))\ mod\ 2 \neq 0 \end{cases} \tag{27}$$

The process entails first scrambling the elements of the image matrix and then performing a similar filtering operation on the scrambled matrix. First, use the index matrix S to scramble the image matrix elements. Then, expand the image matrix I and the index matrix S to sequences I' and S', respectively. Each time, half of the sequence length *len* is used as the starting point *start* and, the element s' at the corresponding position in S' is taken as the index value to take out the element i' at the position $((start + s')\ mod\ len)$ in I' and add it to the end of the new

sequence E. The position of element $i^{'}$ is used as the new starting point *start*, and the above process is repeated until all elements in $I^{'}$ are scrambled. Fig. 5 is an example of a single scrambling process. In order to make the scrambling process more complicated, the scrambling algorithm in the scheme will repeat the above scrambling process at most twice. Let the sequence composed of the remaining elements after screening be *restE*. If there are still elements that fail to complete the scrambling after two scrambling processes, the remaining elements will be placed in the remaining positions in turn, and the scrambling process will end. Algorithm 1 is the scrambling algorithm. Fig. 6 is an example of the multiple scrambling process.

Algorithm 1 Scrambling algorithm based on the domino tiling problem

Input: image matrix *Img*, initial value matrix *Initial*, target value matrix *Full*, flag bit matrix *Flags*1, index matrix *S*, random number matrix $z1^{'}$, $u1^{'}$
Output: Scrambled image matrix *Finish*

```
count = 0, roundCount = 1;
for i = 1: row of Img do
  for j = 1: column of Img do
    next = E(i,j);
    if (Initial(i,j) + roundCount × next) is greater or equal to Full(i,j)
      Finish(i,j) = next, count = count + 1, Flags1(i,j) = 1;
    end if
  end for
end for
while count is less than number of Finish
  roundCount = rountCount + 1;
  [NZRows, NZCols] = find(Flags1 = = 0);
  for i = 1: number of NZRows do
    if z1'(NZRows(i), NZCols(i)) mod 2 is equal to 0
      n = 2;
    else
      n = 3;
    end if
    pos = ((NZCols(i) − 1) × size(img, 1) + NZRows(i)) × roundCount mod numel (z1') + 1;
    N = u1'(pos);
    if z1'(pos) meets one of the special cases
      S(i) = z1'(pos);
    else
      S(i) = dominoesTile(n, N, z1'(pos), Flags(pos));
    end if
  end for
  Repeat the scrambling process
end while
```

2.2.5. Bit-level scrambling: A scrambling method based on the domino knocking problem

Use the sequence $v1$ as the mask sequence *coveringSeq*1, and use the threshold sequence *thresholdSeq* to generate the index sequence *resultLR*, *resultUD* for binary shift according to Eqs. (28)–(32).

$$dominoes = \text{divideDominoes}(coveringSeq1) \tag{28}$$

where the divideDominoes function is used to determine the result after the tilings represented by the mask sequence are divided according to Fig. 4.

$$\begin{cases} horizontalSeq(i) = \text{'\_'}, verticalSeq(i)=^{'}\text{'.'}^{'}; \text{ if } dominoes(i) = \text{'\_'} \\ horizontalSeq(i)=^{'}\text{'.'}^{'}, verticalSeq(i)=^{'}\text{'|'}^{'}; \text{ else} \end{cases} \tag{29}$$

$$\begin{cases} pushLRSeq=^{'}L^{'}, \text{ if } verticalSeq(i)=^{'}\text{'|'}^{'} \ \& \ thresholdSeq(i) < 0.5 \\ pushLRSeq=^{'}R^{'}, \text{ if } verticalSeq(i)=^{'}\text{'|'}^{'} \ \& \ thresholdSeq(i) > 0.5 \\ pushLRSeq=^{'}\text{'.'}^{'}, \text{ if } verticalSeq(i)=^{'}\text{'|'}^{'} \ \& \ thresholdSeq(i) = 0.5 \end{cases} \tag{30}$$

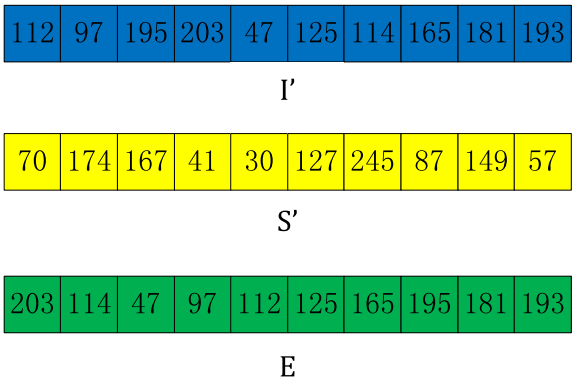

Fig. 5. Single scrambling process.

$$\begin{cases} pushUDSeq=^{'}L^{'}, \text{ if } horizontalSeq\ (i)=^{'}\text{\_}^{'} \ \& \ thresholdSeq(i) < 0.5 \\ pushUDSeq=^{'}R^{'}, \text{ if } horizontalSeq\ (i)=^{'}\text{\_}^{'} \ \& \ thresholdSeq(i) > 0.5 \\ pushUDSeq=^{'}\text{'.'}^{'}, \text{ if } horizontalSeq\ (i)=^{'}\text{\_}^{'} \ \& \ thresholdSeq(i) = 0.5 \end{cases} \tag{31}$$

$$\begin{cases} resultLR = \text{getFinalState}(pushLRSeq) \\ resultUD = \text{getFinalState}(pushUDSeq) \end{cases} \tag{32}$$

where the getFinalState function is used to solve the state of dominoes represented by the simulation sequence after being pushed down. Fig. 7 is an example of the domino knocking problem.

Convert the decimal $r \times c$ image matrix into a binary $r \times (c \times 8)$ matrix *bita*; shift the rows and columns of matrix *bita* according to Algorithm 2 to obtain the binary matrix *bitb*; and then, convert *bitb* into a decimal matrix.

Algorithm 2 Binary matrix shift algorithm

Input: binary matrix *bita*, index sequences *resultLR*, *resultUD*
Output: binary matrix *bitb*

```
for i = 1 : 2 : r do
  cntL = 0, cntR = 0;
  for j = (i − 1) × c × 8 +1 : i × c × 8 do
    if resultLR(j) is equal to 'L'
      cntL = cntL + 1;
    elseif resultLR(j) is equal to 'R'
      cntR = cntR + 1;
    end if
  end for
  bitb(i,:) = circshift(bita(i,:), −cntL, 2), bitb(i + 1,:) = circshift(bitb(i + 1:), cntR, 2);
end for
for i = 1 : 2 : c × 8 do
  cntU = 0, cntD = 0;
  for j = (i − 1) × r +1 : i × r do
    if resultUD(j) is equal to 'L'
      cntU = cntU + 1;
    elseif resultUD(j) is equal to 'R'
      cntD = cntD + 1;
    end if
  end for
  bitb(:,i) = circshift(bitb(:,i), −cntU, 1), bitb(:,i + 1) = circshift(bitb(:,i + 1), cntD, 1);
end for
```

2.2.6. Diffusion method based on tetrominoes

Before the diffusion process, the image matrix needs to be preprocessed, as shown in Fig. 8. The random number sequences $seq11 - seq41$ are added to the outside of the image matrix to prevent overflow.

$$I = [seq31, seq32, seq33, [seq11; seq12; seq13; I; seq23; seq22; seq21], seq43, seq42, seq41] \tag{33}$$

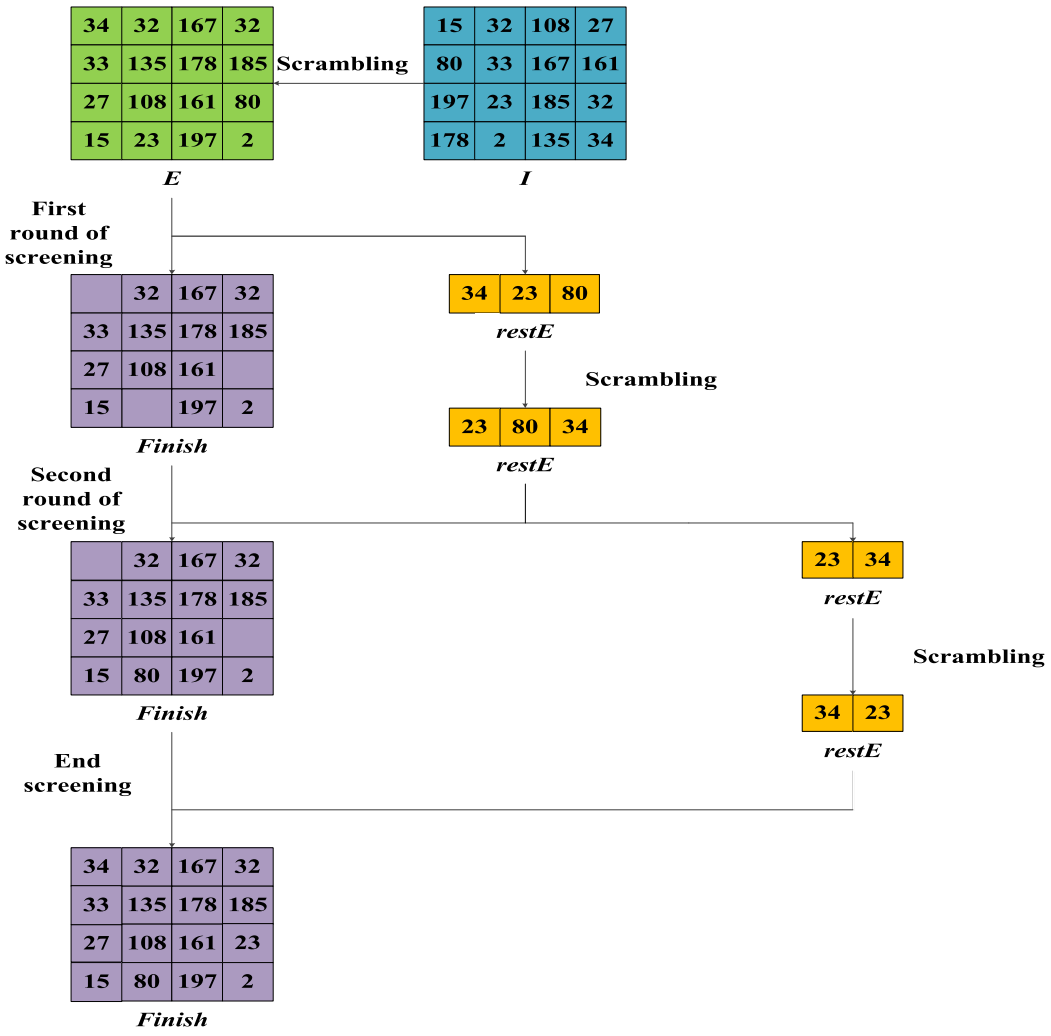

Fig. 6. Multiple scrambling process.

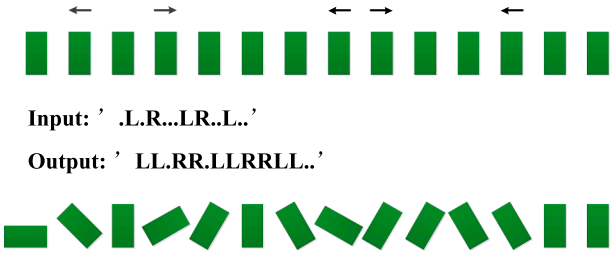

Fig. 7. Example of a domino knocking problem.

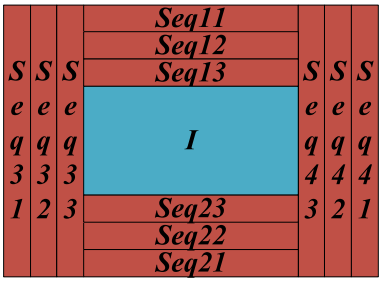

Fig. 8. Preprocessing of diffusion operations.

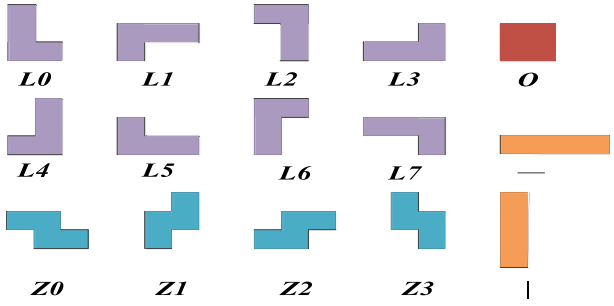

Fig. 9. Fifteen mask shapes.

Then, use the sequences $v2$ and $w3$ as the mask sequences *coveringSeq*2 and *coveringSeq*3 to segment the tilings in Fig. 3 to form the15 mask shapes shown in Fig. 9. According to the mask shapes, perform a diffusion operation on the matrix, and define this operation as shape_bitxor. That is, select the three adjacent pixels of the pixel point $I(i,j)$ according to the parameter *shape*, and then, obtain $I'(i',j')$.

$$\begin{cases} I'(i',j') = I(i,j) \oplus I(i,j+1) \oplus I(i+1,j) \oplus I(i+1,j+1), \ if\ shape='O' \\ I'(i',j') = I(i,j) \oplus I(i,j+1) \oplus I(i,j+2) \oplus I(i,j+3), \ if\ shape='-' \\ I'(i',j') = I(i,j) \oplus I(i+1,j) \oplus I(i+2,j) \oplus I(i+3,j), \ if\ shape='|' \\ I'(i',j') = I(i,j) \oplus I(i+1,j) \oplus I(i+2,j) \oplus I(i+2,j+1), \ if\ shape='L0' \\ I'(i',j') = I(i,j) \oplus I(i,j-1) \oplus I(i,j-2) \oplus I(i-1,j-2), \ if\ shape='L1' \\ I'(i',j') = I(i,j) \oplus I(i-1,j) \oplus I(i-2,j) \oplus I(i-2,j-1), \ if\ shape='L2' \\ I'(i',j') = I(i,j) \oplus I(i,j+1) \oplus I(i,j+2) \oplus I(i-1,j+2), \ if\ shape='L3' \\ I'(i',j') = I(i,j) \oplus I(i+1,j) \oplus I(i+2,j) \oplus I(i+2,j-1), \ if\ shape='L4' \\ I'(i',j') = I(i,j) \oplus I(i,j-1) \oplus I(i,j-2) \oplus I(i-1,j-2), \ if\ shape='L5' \\ I'(i',j') = I(i,j) \oplus I(i-1,j) \oplus I(i-2,j) \oplus I(i-2,j+1), \ if\ shape='L6' \\ I'(i',j') = I(i,j) \oplus I(i,j+1) \oplus I(i,j+2) \oplus I(i+1,j+2), \ if\ shape='L7' \\ I'(i',j') = I(i,j) \oplus I(i,j+1) \oplus I(i+1,j+1) \oplus I(i+1,j+2), \ if\ shape='Z0' \\ I'(i',j') = I(i,j) \oplus I(i+1,j) \oplus I(i+1,j-1) \oplus I(i+2,j-1), \ if\ shape='Z1' \\ I'(i',j') = I(i,j) \oplus I(i,j-1) \oplus I(i+1,j-1) \oplus I(i+1,j-2), \ if\ shape='Z2' \\ I'(i',j') = I(i,j) \oplus I(i-1,j) \oplus I(i-1,j-1) \oplus I(i-2,j-1), \ if\ shape='Z3' \end{cases} \tag{34}$$

Algorithm 3 is the diffusion algorithm, where the chooseCoveringAndDirection function is used to determine the shape of the tetrominoes; the first parameter is used to obtain the number of tilings according to Fig. 3; and the second parameter is used to determine the number of masks according to Fig. 9.

Algorithm 3 Diffusion method based on tetrominoes

Input: matrix $Y5$, sequences $coveringSeq1$, $coveringSeq2$, $rotSeq1$, $rotSeq2$, $chSeq$, $S1$
Output: post-diffusion matrix $Y7$

```
k = 1, rotMat1 = Y5, rotMat2 = Y5, tempCol1 = Y5(:,3), tempCol2 = (:,n5−2);
for i = 4 : m5 −3 do
  tempRow1 = Y5(i−1,:), tempRow2 = Y5(i+1,:);
  if the row of rotMat1 is less than the column of rotMat1
    Y5(i−1,:) = rotMat1(chSeq(i−3),:), Y5(:,3) = rotMat1(:,chSeq(i−1));
  else
    Y5(i−1,:) = rotMat1(:,chSeq(i−3))', Y5(:,3) = rotMat1(chSeq(i−3),:)';
  end if
  if the row of rotMat2 is less than the column of rotMat2
    Y5(i+1,:) = rotMat2(m5 − chSeq(i−3)+1,:), Y5(:,n5−2) = rotMat2(:,n5 − chSeq(i−3)+1);
  else
    Y5(i+1,:) = rotMat2(:,m5 − chSeq(i−3)+1)', Y5(:,n5−2) = rotMat2(n5 − chSeq(i−3)+1,:)';
  end if
  for j = 4 : n5 −3 do
    shape = chooseCoveringAndDirection(coveringSeq2(k), coveringSeq3(k));
    Y6(i−3, j−3) = shape_bitxor(Y5(i, j), shape);
    if j is equal to 4
      if i is equal to 4
        Y7(i−3, j−3) = mod(S1(i−3, j−3) + Y6(i−3, j−3), 256);
      else
        Y7(i−3, j−3) = mod(Y7(i−4, n5−6) + S1(i−3, j−3) + Y6(i−3, j−3), 256);
      end if
    else
      Y7(i−3, j−3) = mod(Y7(i−3, j−4) + S1(i−3, j−3) + Y6(i−3, j−3), 256);
    end if
    Y5(i, j) = Y7(i−3, j−3), k = k + 1;
  end for
  Y5(i−1,:) = tempRow1, Y5(i+1,:) = tempRow2, Y5(:,3) = tempCol1, Y5(:,n5−2) = tempCol2;
  if i is not equal to m5 −3
    rotMat1 = rot90(Y5, rotSeq1(i−3)), rotMat2 = rot90(rotMat1, rotSeq2(i−3));
  end if
end for
```

3. A visually secure image encryption scheme based on adaptive block compressed sensing (ABCS) and NMF

3.1. Key and random sequence generation

3.1.1. Key generation for the 6D hyperchaotic system

First, use the SHA512 function to generate the 512-bit hash value K of the plain image.

$$K = \mathrm{SHA512}(img) \tag{35}$$

Convert K into 64 decimal numbers in groups of 8 bits, denoted as array k. Then, obtain matrices $k11$ –$k62$ and $k7$ through k. The calculation process is shown in Eq. (36).

$$\begin{cases} k11 = \mathrm{reshape}(k(1:6),3,2), k12 = \mathrm{reshape}(k(7:10),2,2) \\ k21 = \mathrm{reshape}(k(11:16),3,2), k22 = \mathrm{reshape}(k(17:20),2,2) \\ k31 = \mathrm{reshape}(k(21:26),3,2), k32 = \mathrm{reshape}(k(27:30),2,2) \\ k41 = \mathrm{reshape}(k(31:36),3,2), k42 = \mathrm{reshape}(k(37:40),2,2) \\ k51 = \mathrm{reshape}(k(41:46),3,2), k52 = \mathrm{reshape}(k(47:50),2,2) \\ k61 = \mathrm{reshape}(k(51:56),3,2), k62 = \mathrm{reshape}(k(57:60),2,2) \\ k7 = k(61:64) \end{cases} \tag{36}$$

For $k11$ –$k62$, calculate as per Eq. (37).

$$\begin{cases} kr1 = \mathrm{KR}(k11,k12), kr2 = \mathrm{KR}(k21,k22), kr3 = \mathrm{KR}(k31,k32) \\ kr4 = \mathrm{KR}(k41,k42), kr5 = \mathrm{KR}(k51,k52), kr6 = \mathrm{KR}(k61,k62) \end{cases} \tag{37}$$

where KR is used to compute the Khatri–Rao product of two matrices. $k7$ is used to operate with $kr1$ –$kr6$. For example, calculate the Kronecker product of $k7$ and $kr1$ to obtain a 6×8 matrix $kro71$, and record the average value of $kro71$ as $Mkro71$. Then, convert $kro71$ to a one-dimensional (1D) matrix $kro71'$, and add $Mkro71$ to the end of the matrix $kro71'$ to form a 1×49 matrix $kro71''$ and rearrange it into a 7×7 matrix $kro71'''$.

$$\begin{cases} kro71 = kron(k1, kr1) \\ kro71' = reshape(kro71, 1, 48) \\ kro71'' = [kro71', Mkro71] \\ kro71''' = reshape(kro71'', 7, 7) \end{cases} \tag{38}$$

Perform SVD on $kro71'''$ to obtain the diagonal matrix $s1$ containing the singular values, and use the mean value $me1$ of $s1$ to obtain.

$$\begin{cases} [u1, s1, v1] = svd(kro71''') \\ key1 = mod(me1/10^7, 1) \end{cases} \tag{39}$$

Similarly, we can obtain $key2$ –$key6$, $key1$, $key2$, $key3$, $key4$, $key5$, $key6$ to form a set of keys as the initial values x_0, y_0, z_0, u_0, v_0, w_0 of the 6D hyperchaotic system.

3.1.2. Key generation for the hybrid chaotic map

Use the array k generated by the hash value of the plain image to obtain the matrix $k1$ –$k4$ according to Eq. (40).

$$\begin{cases} k1 = \mathrm{reshape}(k(1:16),4,4), k2 = \mathrm{reshape}(k(17:32),4,4) \\ k3 = \mathrm{reshape}(k(33:48),4,4), k4 = \mathrm{reshape}(k(49,64),4,4) \end{cases} \tag{40}$$

Calculate the Hadamard product of $k1$ and $k4$ to obtain $k14$ and the Hadamard product of $k2$ and $k3$ to obtain $k23$.

$$\begin{cases} k14 = k1. \times k4 \\ k23 = k2. \times k3 \end{cases} \tag{41}$$

Perform SVD on $k14$ and $k23$ to obtain the two diagonal matrices $s14$ and $s23$ containing the singular values.

$$\begin{cases} [u14, s14, v14] = \mathrm{svd}(k14) \\ [u23, s23, v23] = \mathrm{svd}(k23) \end{cases} \tag{42}$$

$key14$ and $key23$ are obtained as the parameters and initial values r_0 and xx_0 of the hybrid chaotic map through the average values $me14$ and $me23$ in $s14$ and $s23$.

$$\begin{cases} key14 = 1.4 + mod\left(me14/10^4, 2.6\right) \\ key23 = mod\left(me23/10^4, 1\right) \end{cases} \tag{43}$$

3.1.3. Random sequence generation

The random sequences used in scrambling, diffusion and embedding in the proposed scheme are all generated by the 6D hyperchaotic system.

Let the size of image matrix be $m \times n$; the total number of pixels be *num*; and the sampling interval of the chaotic sequences be d. First, substitute the initial values of the chaotic system x_0, y_0, z_0, u_0, v_0, w_0, and iterate $num \times d$ times to generate six corresponding chaotic sequences xn, yn, zn, un, vn, wn. Use d to sample the six sequences to obtain the corresponding subsequences according to Eqs. (44)–(49).

$$\begin{cases} x1 = \text{floor}(\text{abs}(xn(1:d:end-d+1)\times 100)+1) \\ x2 = mod\,(\text{floor}(\text{abs}(xn(2:d:end-d+2)\times 100)+1), 256) \\ x3 = mod\,(\text{floor}(\text{abs}(xn(3:d:end-d+3)\times 100)+1), 256) \\ x4 = mod\,(\text{floor}(\text{abs}(xn(4:d:end-d+4)\times 100)+1), 256) \end{cases} \tag{44}$$

$$\begin{cases} y1 = \text{floor}(\text{abs}(yn(1:d:end-d+1)\times 100)+1) \\ y2 = mod\,(\text{floor}(\text{abs}(yn(2:d:end-d+2)\times 100)+1), 256) \\ y3 = mod\,(\text{floor}(\text{abs}(yn(3:d:end-d+3)\times 100)+1), 256) \\ y4 = mod\,(\text{floor}(\text{abs}(yn(4:d:end-d+4)\times 100)+1), 256) \end{cases} \tag{45}$$

$$\begin{cases} z1 = \text{floor}(mod\,(\text{abs}(zn(1:d:end-d+1)\times 100), 100)+1) \\ z2 = mod\,(\text{floor}(\text{abs}(zn(2:d:end-d+2)\times 100)+1), 256) \\ z3 = mod\,(\text{floor}(\text{abs}(zn(3:d:end-d+3)\times 100)+1), 256) \\ z4 = mod\,(\text{floor}(\text{abs}(zn(4:d:end-d+4)\times 100)+1), 256) \end{cases} \tag{46}$$

$$\begin{cases} u1 = \text{floor}(mod\,(\text{abs}(un(1:d:end-d+1)\times 5), 9)+1) \\ u2 = mod\,(\text{floor}(\text{abs}(un(2:d:end-d+2)\times 100)+1), 256) \\ u3 = mod\,(\text{floor}(\text{abs}(un(3:d:end-d+3)\times 100)+1), 256) \\ u4 = mod\,(\text{floor}(\text{abs}(un(4:d:end-d+4)\times 100)+1), 256) \end{cases} \tag{47}$$

$$\begin{cases} v1 = \text{floor}(mod\,(\text{abs}(vn(1:d:end-d+1)\times 100), 16)+1) \\ v2 = \text{floor}(mod\,(\text{abs}(vn(2:d:end-d+2)\times 100), 16)+1) \\ v3 = \text{floor}(\text{abs}(vn(3:d:end-d+3)\times 100)+1) \\ v4 = \text{floor}(\text{abs}(vn(4:d:end-d+1)\times 100)+1) \\ v5 = mod\,(\text{abs}(vn(5:d\times 2:end)), 1), v6 = mod\,(\text{abs}(vn(6:d\times 2:end)), 1) \\ v56 = [v5, v6] \end{cases} \tag{48}$$

$$\begin{cases} w1 = \text{floor}(\text{abs}(wn(1:d:end-d+1)\times 100)+1) \\ w2 = \text{floor}(\text{abs}(wn(2:d:end-d+2)\times 100)+1) \\ w3 = \text{floor}(mod\,(\text{abs}(wn(3:d:end-d+3)\times 100), 4)+1) \\ w4 = \text{floor}(\text{abs}(wn(4:d:end-d+4)\times 100)+1) \\ w5 = mod\,(\text{abs}(wn(5:d\times 2:end)), 1), w6 = mod\,(\text{abs}(wn(6:d\times 2:end)), 1) \\ w56 = [w5, w6] \end{cases} \tag{49}$$

Quantize the sequence $w1, w2, w4, v3, v4$ to the [0,1] interval according to Eq. (50).

$$\begin{cases} w1(i) = 1/(1+\exp(w1(i))), w2(i) = 1/(1+\exp(w2(i))), w4(i) = 1/(1+\exp(w4(i))) \\ v3(i) = 1/(1+\exp(v3(i))), v4(i) = 1/(1+\exp(v4(i))), i = 1, 2, \ldots, n \end{cases} \tag{50}$$

Perform further operations on the above sequence to obtain the threshold sequence *thresholdSeq* for the bit-level scrambling process via.

$$\begin{cases} w11 = mod\,(w1 + w4, 1), w12 = mod\,(\text{ones}(1,n) + w1 - w4, 1) \\ w21 = mod\,(w2 + w4, 1), w22 = mod\,(\text{ones}(1,n) + w2 - w4, 1) \\ v31 = mod\,(v3 + w4, 1), v32 = mod\,(\text{ones}(1,n) + v3 - w4, 1) \\ v41 = mod\,(v4 + w4, 1), v42 = mod\,(\text{ones}(1,n) + v4 - w4, 1) \\ thresholdSeq = [w11, w21, v31, v41, w12, w22, v32, v42] \end{cases} \tag{51}$$

where the ones(·) function is used to construct an all-one matrix. After that, we construct 12 random number sequences $seq11 - seq43$ for the diffusion process according to Eq. (52)and insert them into the outside of the image matrix.

$$\begin{cases} seq11 = x2(1:M:end), seq12 = y3(1:M:end), seq13 = z4(1:M:end) \\ seq21 = y2(1:M:end), seq22 = z3(1:M:end), seq23 = u4(1:M:end) \\ seq31 = [z2(1:N:end), z2(2:7)]', seq32 = [u3(1:N:end), u3(3:8)]', seq33 = [x4(1:N:end), x4(4:9)]' \\ seq41 = [u2(1:N:end), u2(2:7)]', seq42 = [x3(1:N:end), x3(3:8)]', seq43 = [y4(1:N:end), y4(4:9)]' \end{cases} \tag{52}$$

Construct *rotSeq*1, *rotSeq*2 and *chSeq* for the diffusion process.

$$\begin{cases} rotSeq1 = mod\,(x2(2:N:end-N), 3)+1 \\ rotSeq2 = mod\,(y2(2:N:end-N), 3)+1 \\ chSeq = mod\,(z2(2:N:end), 3)+1 \end{cases} \tag{53}$$

Then, construct sequences $S1$ and $S2$ for diffusing the image matrix according to Eqs.(54)–(57).

$$\begin{cases} S11 = x3(3:8:end), S12 = y3(3:8:end), S13 = z3(3:8:end), S14 = u3(3:8:end) \\ S15 = x4(3:8:end), S16 = y4(3:8:end), S17 = z4(3:8:end), S18 = u4(3:8:end) \end{cases} \tag{54}$$

Fig. 10. Encryption process flow chart.

$$\begin{cases} S1(i) = S11\left(\frac{i-1}{8}+1\right), S1(i+1) = S12\left(\frac{i-1}{8}+1\right), S1(i+2) = S13\left(\frac{i-1}{8}+1\right) \\ S1(i+3) = S14\left(\frac{i-1}{8}+1\right), S1(i+4) = S15\left(\frac{i-1}{8}+1\right), S1(i+5) = S16\left(\frac{i-1}{8}+1\right) \\ S1(i+6) = S17\left(\frac{i-1}{8}+1\right), S1(i+7) = S18\left(\frac{i-1}{8}+1\right), \ \mathrm{i} = 1, 9, 17, \ldots, M \times N \end{cases} \tag{55}$$

$$\begin{cases} S21 = x3(4:8:end), S22 = y3(4:8:end), S23 = z3(4:8:end), S24 = u3(4:8:end) \\ S25 = x4(4:8:end), S26 = y4(4:8:end), S27 = z4(4:8:end), S28 = u4(4:8:end) \end{cases} \tag{56}$$

$$\begin{cases} S2(i) = S21\left(\frac{i-1}{8}+1\right),\ S2(i+1) = S22\left(\frac{i-1}{8}+1\right), S2(i+2) = S23\left(\frac{i-1}{8}+1\right) \\ S2(i+3) = S24\left(\frac{i-1}{8}+1\right), S2(i+4) = S25\left(\frac{i-1}{8}+1\right), S2(i+5) = S26\left(\frac{i-1}{8}+1\right) \\ S2(i+6) = S27\left(\frac{i-1}{8}+1\right), S2(i+7) = S28\left(\frac{i-1}{8}+1\right),\ \mathrm{i} = 1, 9, 17, \ldots, M \times N \end{cases} \tag{57}$$

3.2. Encryption process

The encryption process in this study is divided into two stages. In the first stage, after the plain image is parsed through Tetrolet transform, the image is compressed through the process of ABCS, and then, the compressed image is encrypted into a noise-like secret image by the proposed scrambling and diffusion methods. In the second stage, a meaningful cipher image is obtained by embedding the secret image into the high-frequency coefficients of the carrier image through NMF. The encryption process is shown in Fig. 10.

3.2.1. Encrypt plain image to secret image

Step 1: Perform Tetrolet transform on the plain image to generate a coefficient matrix X.

Step 2: Set the block size B_1; rearrange X into a matrix $X1$ with row dimension B_1.In order to reduce the maximum block sparsity and improve the sampling efficiency, matrix scrambling is performed on $X1$ to balance the block sparsity, and $X2$ is obtained.

Step 3: Given a block size B_2, calculate the sampling rate of each block according to the adaptive block sampling strategy based on region energy described in Section 2.2.1.

Step 4: Let the image size be $M \times N$. According to the maximum block sampling number *maxAM*, the hybrid chaotic system is iterated $maxAM \times M \times N + t$ times. The first t values are discarded, and the remaining values generate the initial measurement matrix *chaosMat* of $maxAM \times M \times N$. According to the method described in Section 2.2.2, the measurement matrix *chaosMat* is optimized to obtain the optimized measurement matrix Φ and finally normalized to obtain the measurement matrix Φ'. The normalization process is as per Eq. (58).

$$\Phi' = \sqrt{\frac{2}{M}}\Phi \tag{58}$$

Step 5: Determine the final block measurement matrix Φ'' according to the block sampling rates; divide $X2$ into blocks according to B_2; and convert each block into a column vector to obtain matrix $X3$.

$$\Phi'' = \begin{pmatrix} \Phi_1' \\ \Phi_2' \\ \Phi_3' \\ \Phi_4' \end{pmatrix} \tag{59}$$

where Φ_i' is the matrix of $AM'(i) \times B_2^2$, which is obtained by rearranging the column vector $\Phi'(1 : AM'(i) \times B_2^2, i), i = 1, 2, \ldots, M \times N / B_2^2$. Then, use block compressed sensing to compress the coefficient matrix and obtain the measurement value matrix $Y1$.

$$Y_i = \Phi_i X3(:, i) \tag{60}$$

$$Y1 = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_{M \times N / B_2^2} \end{pmatrix} \tag{61}$$

After that, rearrange $Y1$ into an $(SR \times M) \times N$ matrix.

Step 6: Quantize matrix $Y1$ to the [0,255] interval to obtain the matrix $Y2$.

$$Y2 = round\left(255 \times \frac{Y1 - Min}{Max - Min}\right) \tag{62}$$

where *Min* and *Max* are the maximum and minimum values of $Y1$, respectively.

Step 7: $Y3$ is obtained by performing pixel-level scrambling on the matrix $Y2$ with the scrambling method based on the domino tiling problem described in Section 2.2.4.

Step 8: Using the scrambling method based on the domino knocking problem described in Section 2.2.5, bit-level scrambling is performed on the matrix $Y3$ to obtain $Y4$.

Step 9: The matrix $Y4$ is diffused to obtain $Y7$ using the diffusion method based on tetrominoes described in Section 2.2.6, and then, another diffusion operation is performed to obtain $Y8$ according to Eq. (63), which is the final secret image *secImg*.

$$Y8(i) = mod\left(Y8(i+1) + S2(i) + Y7(i), 256\right) \tag{63}$$

3.2.2. Embed the secret image into the carrier image

Step 1: Let the rank of the NMF $r = 256$, and use the sequence $v56$, $w56$ to obtain the initial iterative matrix *Winit*, *Hinit*.

$$\begin{cases} Winit = \mathrm{reshape}(v56, 256, 256) \\ Hinit = \mathrm{reshape}(w56, 256, 256) \end{cases} \tag{64}$$

Step 2: Divide the carrier image matrix into $B_3 \times B_3$ blocks. Divide the secret image matrix *secImg*, the initial iteration matrix *Winit*, *Hinit* into $\frac{B_3}{2} \times \frac{B_3}{2}$ blocks.

Step 3: Perform integer wavelet decomposition (IWT) on the block matrix of the carrier image to obtain the high-frequency coefficient matrix HH_i, and perform NMF on the matrix HH_i to obtain the matrices W_i, H_i.

$$[W_i, H_i] = \mathrm{NMF}(HH_i, Winit, Hinit) \tag{65}$$

Step 4: For matrix H_i, find its orthogonal matrix.

$$OH_i = \mathrm{orth}(H_i) \tag{66}$$

If H_i is not a full rank matrix, then change the rank of the matrix. First, perform SVD on the matrix H_i.

$$[u, s, v] = \mathrm{svd}(H_i) \tag{67}$$

For the singular values in the singular value matrix s, if the following condition is met

$$s(i,j) < 1^{-10} \tag{68}$$

then, let

$$s^{'}(i,j) = s(i-1,j-1) \tag{69}$$

The changed matrix is denoted as $s^{'}$, and then, a new sub-matrix $H_i^{'}$ is obtained by multiplying the matrices.

$$H_i^{'} = u \times s^{'} \times v^T \tag{70}$$

where v^T is the transposed matrix of v.

Step 5: The block matrix of the secret image $secImg_i$ is embedded in the sub-matrix W_i using the addition criterion, and $W_i^{'}$ is obtained.

$$W_i^{'} = W_i + \alpha \times secImg_i \tag{71}$$

Step 6: After that, the block high-frequency coefficient matrix $HH_i^{'}$ is obtained according to Eq. (72).

$$HH_i^{'} = W_i^{'} \times OH_i \tag{72}$$

The block matrices are combined to obtain the matrix $HH^{'}$, and finally, the cipher image is obtained through inverse IWT.

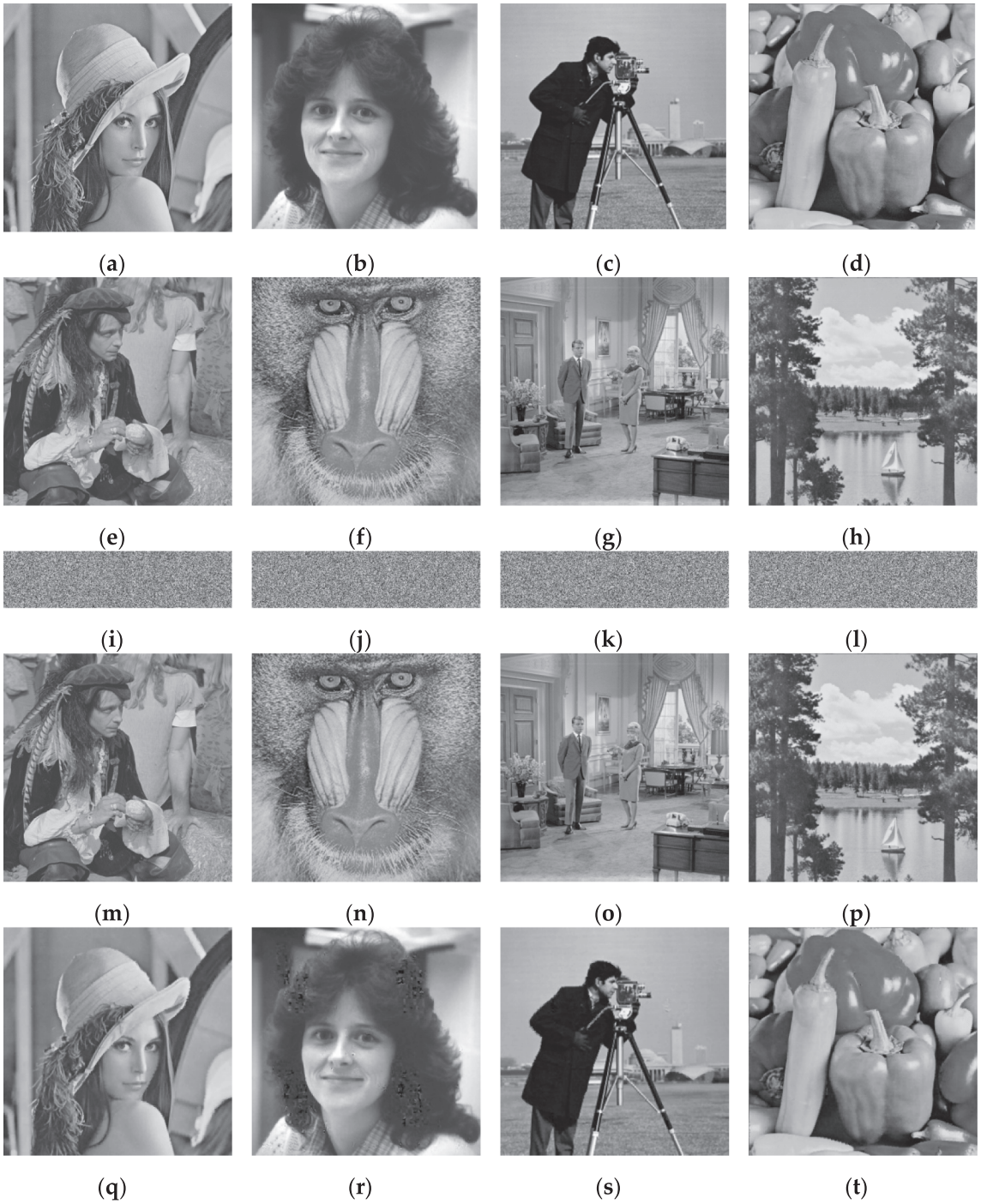

Fig. 11. Simulation results of the scheme. (a)–(d) are the plain images Lena, Woman, Cameraman, and Peppers. (e)–(h) are the carrier images Private, Mandril, Livingroom, and Lake. (i)–(l) are the corresponding secret images. (m)–(p) are the corresponding cipher images. (q)–(t) are the corresponding decrypted images.

Table 2
PSNR and MSSIM values of simulation result.

Plain image	Carrier image	$PSNR_{cip}$(dB)	$MSSIM_{cip}$	$PSNR_{dec}$(dB)	$MSSIM_{dec}$
Lena	Private	32.0495	0.9997	32.4778	0.9606
Woman	Mandril	32.0290	0.9998	32.5132	0.9081
Cameraman	Livingroom	32.2501	0.9997	32.1999	0.9637
Peppers	Lake	31.7729	0.9996	30.8927	0.9601

Table 3
PSNR and MSSIM values for the different carrier images.

Plain image	Carrier image	$PSNR_{cip}$(dB)	$MSSIM_{cip}$
Lena	Private	32.0495	0.9997
	Mandril	32.0350	0.9998
	Livingroom	32.2277	0.9997
	Lake	31.7770	0.9996

Table 4
Comparison of the PSNR values of the decrypted images.

Plain image	Ref. [9]	Ref. [19]	Ref. [30]	Ref. [35]	$PSNR_{dec}$(dB)
Lena	31.4240	31.15	<30		32.4778
Cameraman	30.4164		<30		32.1999
Peppers	30.6809	30.22		29.9759	30.8927
Barbara		26.70			26.1338
Couple	30.1862			29.2881	29.4315
Zelda				32.0312	36.5801
Einstein				31.2057	34.9061

Table 5
Comparison of the PSNR values of the cipher images.

Plain image	Carrier image	Ref. [34]	Ours
Cameraman	Lena	36.6939	38.9829
	Baboon	36.7386	37.2015
	Couple	36.7315	37.8220
	Boat	36.7405	37.0999
	Pepper	36.7453	37.6326

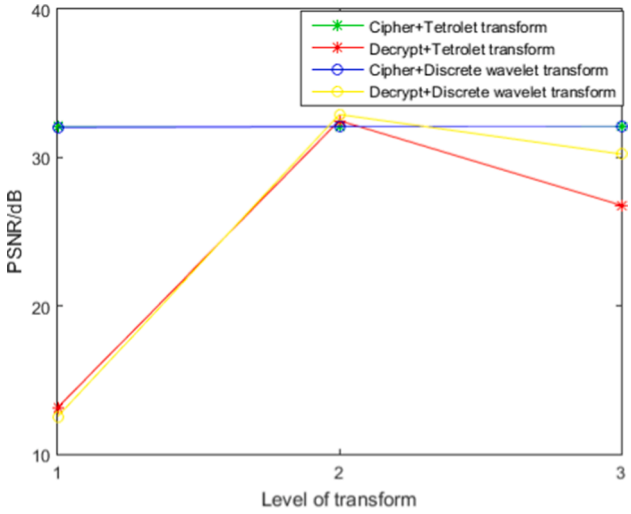

Fig. 12. PSNR vs. level.

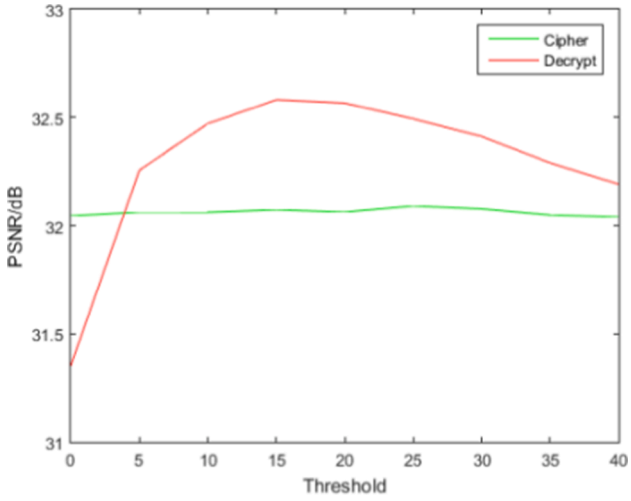

Fig. 13. PSNR vs. threshold.

3.3. Decryption process

The decryption process requires the keys (r_0, xx_0) to generate the measurement matrix and the keys $(x_0, y_0, z_0, u_0, v_0, w_0)$ to generate the random sequences required in the scrambling, diffusion, and embedding processes; embedding strength α; matrix maximum value *Max* and minimum value *Min* for quantization; the number of block samples of the image; the scrambling sequence for optimized matrix scrambling; and tetrominoes masks for the plain image and carrier image. The decryption process is the reverse process of the encryption process and is divided into two stages. The first stage is to extract the secret image from the cipher image, and the second stage is to decrypt the secret image to obtain the decrypted image.

3.3.1. Extract the secret image from the cipher image

The process of extracting the secret image is the inverse process of the embedding process.

Step 1: Divide the plain image and the carrier image into blocks, and perform IWT on the image blocks.

Step 2: By performing NMF on the high-frequency coefficient matrix HH_i of the carrier image block matrix, W_i, H_i are obtained. Then, the orthonormal basis OH_i of H_i is obtained.

Step 3: After obtaining the high-frequency coefficient matrix $HH_i^{'}$ of the block matrix of the cipher image, the sub-matrix $W_i^{'}$ containing the secret image information is obtained.

$$W_i^{'} = HH_i^{'} \times OH_i^{T} \tag{73}$$

Step 4: After obtaining the block matrix $secImg_i$ of the secret image according to Eq. (74), combine the secret image blocks in turn to obtain the complete secret image *secImg*.

$$secImg_i = \mathrm{round}\left(\left(W_i^{'} - W_i\right)/\alpha\right) \tag{74}$$

3.3.2. Decrypt the secret image to the original image

Step 1: First, expand the secret image *secImg* into a 1D vector *YY*8, and perform the inverse diffusion on *YY*8 to obtain *YY*7 according to Eq. (75),

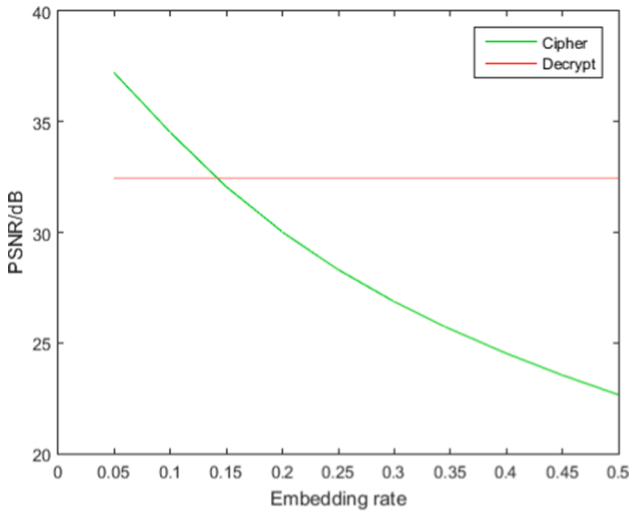

Fig. 14. PSNR vs. embedding rate.

$$YY7(i) = mod\,(256 \times 2 + YY8(i) - YY8(i+1) - S2(i), 256) \tag{75}$$

And then, use the inverse diffusion method based on tetrominoes to inversely diffuse $YY7$ to obtain $YY4$.

Step 2: Use the scrambling method based on the domino knocking problem to inversely transform the matrix $YY4$ to obtain $YY3$.
Step 3: Use the scrambling method based on the domino tiling problem to inversely transform the matrix $YY3$ to obtain $YY2$.
Step 4: Perform the inverse quantization process on the image matrix $YY2$ to obtain $YY1$.

$$YY1 = YY2 \times \frac{Max - Min}{255} + Min \tag{76}$$

where Min and Max are the minimum and maximum values of $Y1$, respectively.

Step 5: The matrix $YY1$ is reconstructed through the SL0 algorithm to obtain $XX3$. Then, the column vectors in $XX3$ are converted into matrices of $B_2 \times B_2$ dimensions and arranged in turn to obtain the matrix $XX2$.
Step 6: Perform the inverse transformation of optimized matrix scrambling on the matrix $XX2$ to obtain $XX1$, and reorganize the matrix to obtain the coefficient matrix XX.
Step 7: Perform inverse Tetrolet transform on XX to generate the decrypted image.

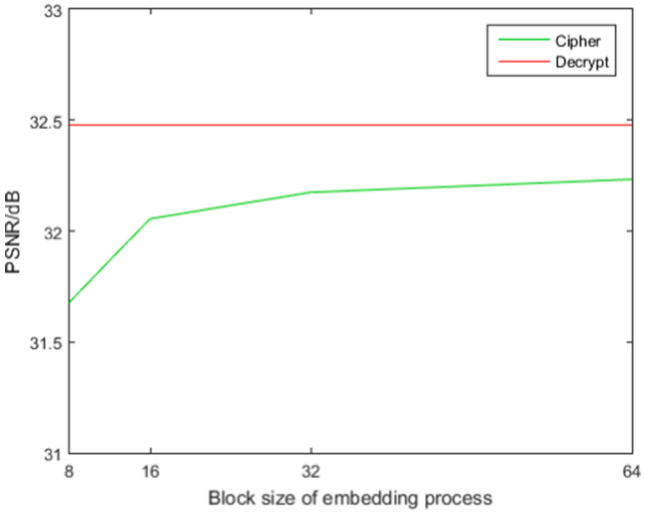

Fig. 16. PSNR vs B_3.

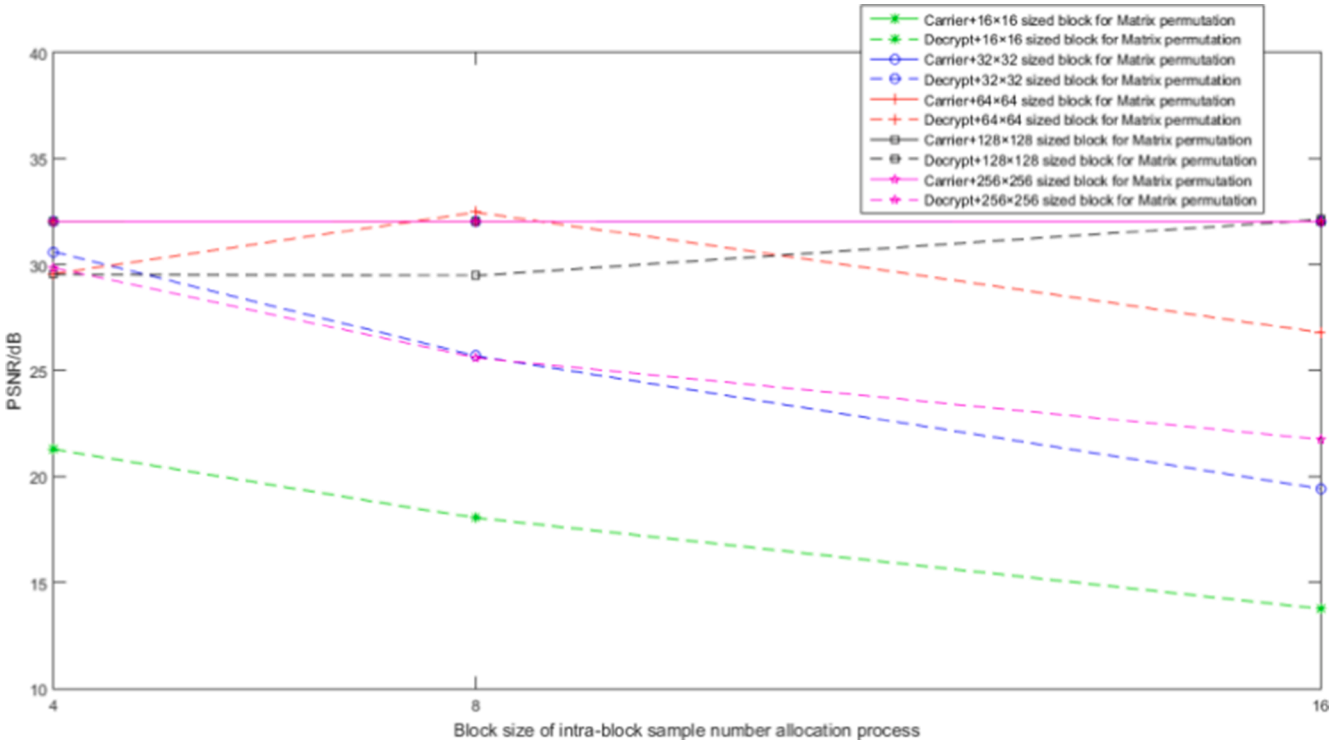

Fig. 15. PSNR vs B_2.

4. Simulation results

4.1. Experimental environment

The computer configuration used in the experiments was Intel Core I5-10500 CPU with 16 GB memory, operating system Windows 10, and the experimental platform MATLAB R2016a. The parameters used in this study are the following: sampling rate $SR = 0.25$, threshold $TS = 11$, block size $B_1 = 64$, $B_2 = 8$, $B_3 = 16$, sampling interval of the chaotic system $d = 4$, number of additional iterations of the chaotic system $t = 800$, and embedding strength factor $\alpha = 0.15$.

4.2. Encryption and decryption results

In this experiment, we selected "Lena", "Woman", "Cameraman" and "Peppers" of 512×512 size as the plain images, and "Private", "Mandril", "Livingroom" and "Lake" of 512×512 size as the carrier images. Fig. 11 shows the experimental results, and Table 2 lists the PSNR and MSSIM values of the corresponding cipher images and decrypted images. The experimental results show that the PSNR values of the cipher images and decrypted images are all above 30 dB, and the MSSIM values are above 0.9, indicating that the proposed scheme has excellent encryption and decryption performance. Table 3 tests the effects of different carrier images on the PSNR and MSSIM values of the cipher images. The experimental results show that the PSNR and MSSIM values of the cipher images corresponding to the different carrier images are very close, indicating that the proposed scheme has good stability. Table 4 lists the PSNR values of different decrypted images and the comparison with other schemes, and the results show that the proposed scheme has advantages. Table 5 lists the comparison of PSNR values of different cipher images with Ref. [34] when embedding strength factor is 0.05, and the results show that our scheme can better maintain the quality of the carrier images.

4.3. Influence of relevant parameters on the encryption and decryption processes

4.3.1. Influence of the wavelet transform type and series on the cipher image and decrypted image

In order to analyze the influence of different wavelet transform types and series on the cipher image and decrypted image, we perform the first-level, second-level, and third-level Tetrolet transform and DWT on the plain image Lena. It can be seen from Fig. 12 that (1) The influence of Tetrolet transform and DWT at all levels on the cipher image can be ignored. (2) Both transformation types have the best effect at the second level, and the change trend is consistent.

4.3.2. Effect of the threshold (TS) on the cipher image and decrypted image

In order to test the effect of different thresholds on the cipher image and decrypted image, we choose Lena as the plain image and Private as the carrier image, and set the threshold from 0 to 40 and the interval as 5. Fig. 13 shows the changes of the PSNR of the cipher image and decrypted image. It can be found that the influence of the change of the threshold on the PSNR values of the cipher image is basically negligible, whereas the influence on the PSNR values of the decrypted image is small. The overall trend of change is first rising and then decreasing.

Table 6
Key space.

Algorithm	Ours	Ref. [19]	Ref. [26]	Ref. [22]	Ref. [30]
Key space	$> 2^{511}$	2^{197}	2^{210}	10^{75}	10^{80}

4.3.3. The effect of the embedding strength (α) on the cipher image and decrypted image

In order to test the effect of different embedding strengths on the cipher image and decrypted image, we set the embedding strength (α) to vary from 0.05 to 0.5 and the interval as 0.05. Fig. 14 is the change of the PSNR of the cipher image and decrypted image. It can be found that the change of the embedding strength has no effect on the PSNR value of the decrypted image, whereas for the cipher image, with the increase of the embedding strength, the PSNR value gradually decreases, and the rate of decrease gradually becomes slower.

4.3.4. The effect of block size in the encryption process on the cipher image and decrypted image

After Tetrolet transform is performed on the image and before the image is measured, there are two processes of segmenting the image. The first is matrix scrambling to equalize the sparsity of the image with the block size of B_1, and the second is to allocate the number of samples required for image compression into each block and to block the measurement matrix with the block size of B_2. Within a range of values, we combine the block size values of the two block operations one by one to obtain the PSNR values of the cipher image and decrypted image under different block combinations. Fig. 15 is drawn based on the experimental results. According to the experimental results, it can be found that different block combinations have little effect on the PSNR value of the cipher image. When the first block size B_1 is 64 and the second block size B_2 is 8, the PSNR value of the decrypted image is the largest. It is worth noting that when the first block size is set to 128 or 256, the running time of the scheme is too long, and when the value is 16, the experimental result is poor. On the whole, the first block size B_1 should take the value of 64 as the best choice.

4.3.5. Influence of block size on the cipher image and decrypted image in the embedding process

In order to study the effect of the block size B_3 used in the embedding process on the cipher image and decrypted image, we select 8, 16, 32, and 64 as the block size values. Fig. 16 shows the experimental results. It can be seen that the block size B_3 has no effect on the PSNR value of the decrypted image, but with the increase of B_3, the PSNR value of the cipher image increases slowly.

4.4. Key space

A sufficiently large key space can make the cryptosystem effectively resistant to brute force attacks. The keys required for the proposed scheme are the following: (1) The hash value of the plain image generated by the SHA512 function; (2) The parameters r_0 and initial state xx_0 of the hybrid chaotic map; (3) The initial state of the 6D hyperchaotic system $(x_0, y_0, z_0, u_0, v_0, w_0)$; (4) Embedding strength α, threshold TS, and sample rate SR. Assuming the computer precision is 10^{-14}, the total key space is

$$2^{512} \times \left(10^{14}\right)^{11} = 2^{512} \times 10^{154} > 2^{511} \tag{77}$$

Table 6 lists the comparison results of this study with other schemes. The experimental results show that the experimental scheme proposed in this study has a large enough key space and can resist exhaustive attacks.

4.5. Scrambling effect analysis

In Ref. [20], a method for evaluating the effect of scrambling algorithms is proposed. The implementation process is to cut the encrypted image, and the cut part is $\left\{clip_{i,j} | (i \in [1,512], j \in [1,50]) \vee (i \in [1,50], j \in [51,512]) \vee (i \in [463,512], j \in [51,512]) \vee (i \in [51,462], j \in [463,512])\right\}$. After decrypting the cropped image, theoretically, the

Fig. 17. Scrambling effect analysis. (a) is the cipher image obtained by scrambling Lean. (b) is the decrypted image of (a). (c) is the cipher image after cutting (a). (d) is the decrypted image of(c).

Table 7
Comparison of the scrambling effects of different algorithms.

Scrambling method	D
Arnold-based scrambling	0.6975
Circular shift scrambling	0.7069
Zigzag scrambling	5.5980
3D zigzag scrambling	0.2227
Ours	0.0079

Table 8
Comparison of the GDD values of two scrambling algorithms.

Image	Ours	Ref. [36]
Lena	0.9505	0.9099
Cameraman	0.9673	0.8858
Livingroom	0.8954	0.8965
Mandril	0.8474	0.7009
Peppers	0.9571	0.9059

Table 9
Comparison of the information entropy of different encryption schemes.

Image	Ref. [4]	Ref. [5]	Ref. [22]	Ref. [37]	Ours
Lena	7.9970	7.9973	7.9968	7.9972	7.9973
Barbara	7.9973		7.9970		7.9972
Jet	7.9970		7.9970		7.9973
Baboon		7.9971			7.9973
Peppers				7.9971	7.9973

frequency of the pixels in the cropped part appearing in the decrypted image is

$$\frac{50\times 512\times 2+412\times 50\times 2}{512\times 512}=0.3525 \tag{78}$$

Therefore, the effect of the scrambling method can be evaluated by calculating the actual distribution of the pixels of the clipped part in the decrypted image. The specific method is to divide the decrypted image into 64 blocks on average, record the frequency of the pixels of the clipped part appearing in each block as q_i, and then calculate the variance by Eq. (79).

$$D=\sum_{i=1}^{64}(q_i-0.3525)^2 \tag{79}$$

The smaller the value of D is, the more evenly distributed the pixels in the clipping part are in the decrypted image, that is, the better the scrambling effect is. Taking the Lena image as the test image, Fig. 17 shows the test results. It can be seen from the results that the pixel distribution is uniform and there is no obvious pattern. Next, using the variance D as the test index and compare the proposed scheme with other scrambling methods. Table 7 shows the experimental results. It can be seen that our method can achieve better scrambling than some other traditional scrambling methods.

We also evaluate the gray difference degree (GDD). GDD is a statistical measure to compare the randomness of plain images and encrypted images [9,36], and the calculation process is as follows. First, calculate the gray difference $GD(i,j)$ of two adjacent pixels in the plain image

$$GD(i,j)=\frac{1}{4}\sum_{i',j'}[I(i,j)-I(i',j')]^2 \tag{80}$$

where $(i',j')=\{(i-1,j),(i+1,j),(i,j-1),(i,j+1)\}$, and $I(i,j)$ is the pixel gray value at (i,j). Then, calculate the average gray difference of the entire plain image.

$$E(GD(i,j))=\frac{\sum_{i=2}^{M-1}\sum_{j=2}^{N-1}GD(i,j)}{(M-2)\times(N-2)} \tag{81}$$

where M and N are the height and width of the entire image. Finally, calculate the GDD between the plain image and the scrambled image.

$$GDD=\frac{E'(GD(i,j))-E(GD(i,j))}{E'(GD(i,j))+E(GD(i,j))} \tag{82}$$

$E'(GD(i,j))$ and $E(GD(i,j))$ are the average gray difference of the plain image and the scrambled image, respectively. The value range of the GDD is [-1, 1],and the closer it is to 1, the better is the scrambling effect. It can be seen from Table 8 that the GDDs of the images tested in this study are all close to 1, indicating that the scrambling algorithm in this study has good scrambling performance.

4.6. Information entropy

Information entropy is an important index to evaluate the randomness of encrypted images [9], and its calculation formula is as shown in Eq. (83).

$$H(m_i)=-\sum_{i=0}^{2^n-1}p(m_i)\log p(m_i) \tag{83}$$

where $p(m_i)$ is the probability of m_i, and for a 256-level grayscale image, the maximum theoretical value of $H(m_i)$ is 8. Table 9 lists the comparison results between the encryption scheme in this study and other schemes. The results show that the proposed scheme has advantages over some other image encryption schemes based on block compressive sensing and DNA computing, and the proposed scheme can better resist entropy attacks.

4.7. Correlation coefficient

There is a high correlation between adjacent pixels of a natural image, and the process of encrypting a plain image is a process of weakening the correlation of adjacent pixels. Many attackers try to

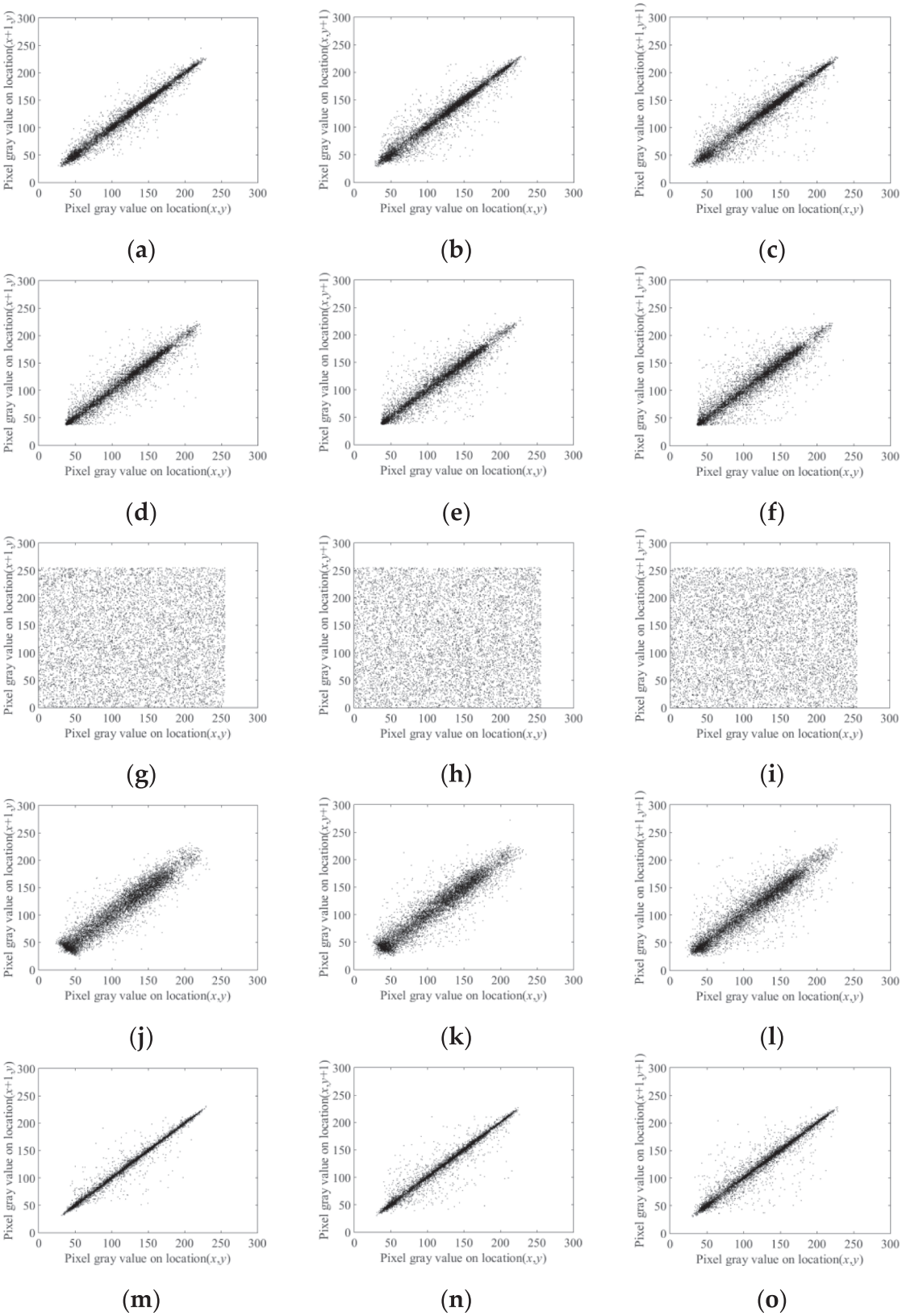

Fig. 18. Correlation analysis of the horizontal, vertical, and diagonal pixel distributions.(a)–(c) are the correlation diagrams of the plain image Lena. (d)–(f) are the correlation diagrams of the carrier image Private. (g)–(i) are the correlation diagrams of the corresponding secret image. (j)–(l) are the correlation diagrams of the corresponding cipher image. (m)–(o) are the correlation diagrams of the corresponding decrypted image.

Table 10
Correlation coefficients of plain images and secret images.

Image	horizontal	vertical	diagonal
Lena			
Plain image	0.9850	0.9718	0.9594
Secret image	−0.0035	0.0003	0.0016
Woman			
Plain image	0.9969	0.9963	0.9946
Secret image	−0.0027	−0.0028	0.0016
Cameraman			
Plain image	0.9900	0.9832	0.9731
Secret image	−0.0049	0.0008	0.0023
Peppers			
Plain image	0.9794	0.9773	0.9643
Secret image	0.0004	0.0069	−0.0011

Table 11
Comparison of the correlation coefficients of encrypted images with different schemes.

	Directions	Ours	Ref. [9]	Ref. [19]	Ref. [30]
Lena	Horizontal	−0.0035	−0.0029	−0.0022	
	Vertical	0.0003	0.0058	0.0023	
	Diagonal	0.0016	−0.0025	0.0034	
Cameraman	Horizontal	−0.0049	−0.0066		0.0041
	Vertical	0.0008	0.01396		0.0043
	Diagonal	0.0023	−0.0063		0.0084

Table 12
NPCR and UACI.

Image	Pixel change	NPCR	UACI	Result
Lena	$P(512,512) = 108 \rightarrow 109$	99.6002%	33.5644%	Pass
Woman	$P(512,512) = 148 \rightarrow 149$	99.6277%	33.5380%	Pass
Cameraman	$P(512,512) = 111 \rightarrow 110$	99.6063%	33.4876%	Pass
Livingroom	$P(512,512) = 159 \rightarrow 158$	99.6323%	33.3846%	Pass
Mandril	$P(512,512) = 89 \rightarrow 88$	99.6231%	33.4854%	Pass

obtain the information of the plain image by analyzing the correlation of adjacent pixels. Therefore, for the encrypted image, the lower the correlation of adjacent pixels, the more difficult it is to be decrypted. The correlation of adjacent pixels can be measured by the correlation coefficient [19]. The calculation formula is as follows.

$$r_{x,y} = \frac{E((x - E(x))(y - E(y)))}{\sqrt{D(x)D(y)}} \tag{84}$$

$$E(x) = \frac{1}{N}\sum_{i=1}^{N} x_i, D(x) = \frac{1}{N}\sum_{i=1}^{N} (x_i - E(x))^2 \tag{85}$$

where x and y represent two adjacent pixels in the image, and N is the number of adjacent pixel pairs randomly selected from the image. The closer the correlation coefficient is to 0, the lower is the correlation between adjacent pixels of the image. In this experiment, we randomly select,8000 pairs of adjacent pixels in the horizontal, vertical, and diagonal directions of different plain images and the corresponding secret images to calculate the correlation coefficients. Fig. 18 lists the test results with Lena as the plain image and Private as the carrier image. It can be seen that the adjacent pixels of the secret image have no apparent correlation. In order to reduce the influence of randomness, we averaged the calculation results in each direction 100 times. Table 10 lists the correlation coefficients of different images in the horizontal, vertical, and diagonal directions. The results show that the correlation coefficients in all directions are very close to 0, indicating that the correlation between the pixels of the encrypted image is greatly reduced. Table 11 lists the comparison of the encrypted images in this study and other schemes in the horizontal, vertical, and diagonal directions, and the results show that the proposed scheme has certain advantages.

Table 13
Key sensitivity of the first group of keys.

Key group	$K_{measurement}$&$K1_{measurement}$	$K_{measurement}$&$K2_{measurement}$
NPCR	99.5499%	99.6140%
UACI	33.3969%	33.3143%

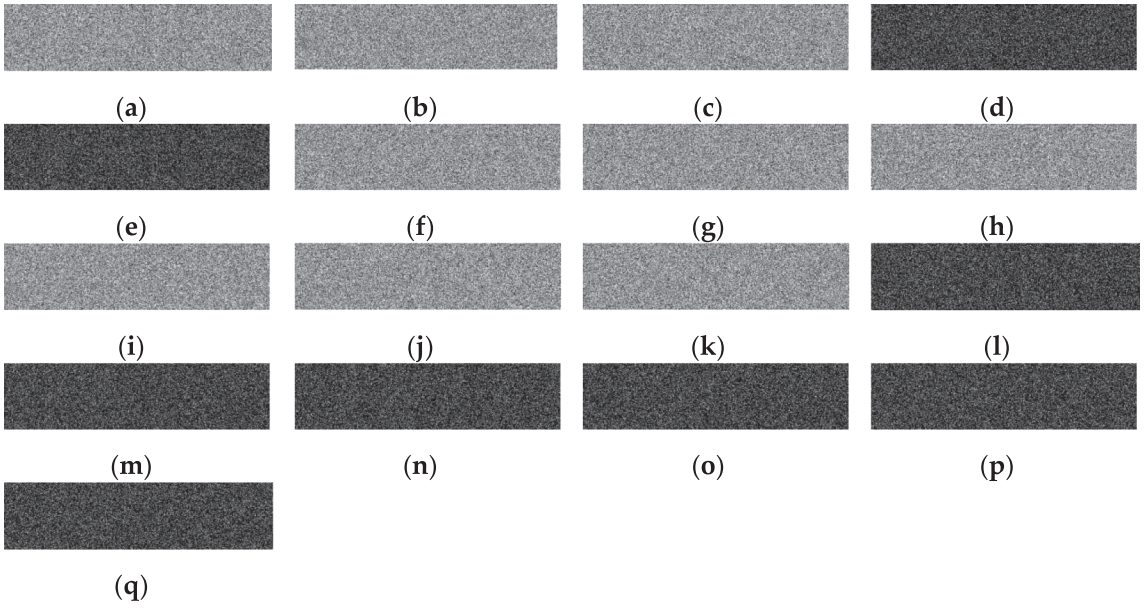

Fig. 19. Key sensitivity of the encryption process. (a)–(c) are the secret images obtained using $K_{measurement}$, $K1_{measurement}$, and $K2_{measurement}$, respectively.(d)–(e) are the differential images of (a) and (b), (a) and (c), respectively; (f)–(k) are the secret images obtained using $K1_{encryption}$, $K2_{encryption}$, $K3_{encryption}$, $K4_{encryption}$, $K5_{encryption}$, and $K6_{encryption}$, respectively; (l)–(q) are the differential images of (a) and (f), (a) and (g), (a) and (h), (a) and (i), (a) and (j), (a) and (k), respectively.

Table 14
Key sensitivity of the second group of keys.

Key group	$K_{encryption}$& $K1_{encryption}$	$K_{encryption}$& $K2_{encryption}$	$K_{encryption}$& $K3_{encryption}$	$K_{encryption}$& $K4_{encryption}$	$K_{encryption}$& $K5_{encryption}$	$K_{encryption}$& $K6_{encryption}$
NPCR	99.6063%	99.5850%	99.5987%	99.6246%	99.5728%	99.6414%
UACI	33.4331%	33.5479%	33.5666%	33.3856%	33.3896%	33.5649%

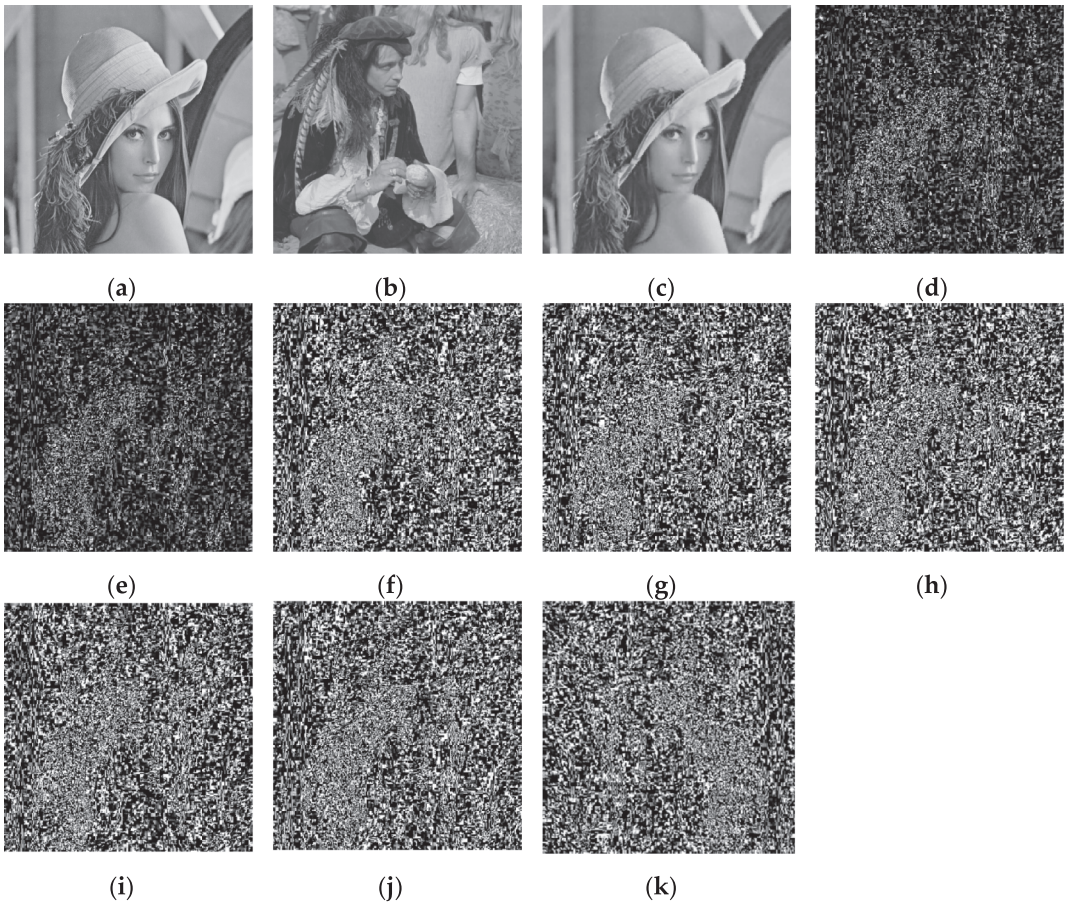

Fig. 20. Key sensitivity of the decryption process. (a) is the plain image Lena. (b) is the carrier image Private. (c) is the decrypted image obtained by the correct key. (d)–(e) are the decrypted images obtained by $K1_{measurement}$ and $K2_{measurement}$, respectively. (f)–(k) are the decrypted images obtained by $K1_{encryption}$, $K2_{encryption}$, $K3_{encryption}$, $K4_{encryption}$, $K5_{encryption}$, and $K6_{encryption}$, respectively.

4.8. Differential attack

Differential attack is a chosen plaintext attack method. Its main idea is to obtain the most likely key by selecting a plaintext pair in a specific form and analyzing the effect of its difference on the ciphertext difference. Generally, the number of pixels' change rate (NPCR) and the unified average change intensity (UACI) are commonly used to measure the resistance of an encryption scheme to differential attacks [38,39]. Eqs. (86)–(88) are the calculation formulas of NPCR and UACI.

$$\mathrm{NPCR}(P_1,P_2)=\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}|\mathrm{Sign}(P_1(i,j)-P_2(i,j))|\times 100\% \tag{86}$$

$$\mathrm{Sign}(x)=\begin{cases}1, x>0\\ 0, x=0\\ -1, x<0\end{cases} \tag{87}$$

$$\mathrm{UACI}(P_1,P_2)=\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\frac{|P_1(i,j)-P_2(i,j)|}{255-0}\times 100\% \tag{88}$$

According to the experimental results in Ref. [40], for a 256×256 image, when the significance level $\alpha=0.05$, the standard value of the NPCR is $N^{*}_{0.05}=99.5693\%$. For two cipher images C_1 and C_2, if their NPCR value $N(C_1,C_2)$ is less than 99.5693%, it is considered that C_1 and C_2 are not random at this significance level. The standard value of UACI is divided into two parts: one is the left value $\mu^{*-}_{0.05}=33.2824$, and the other is the right value $\mu^{*+}_{0.05}=33.6447$. For two cipher images C_1 and C_2, if their UACI value $\mu(C_1,C_2)$ is not within the interval $\left(\mu^{*-}_{0.05},\mu^{*+}_{0.05}\right)$, then it is considered that C_1 and C_2 are not random at this level of significance. In this experiment, we change the pixel value of different plain images at (512, 512) by one bit to obtain the encrypted images before and after the pixel change and then calculate the NPCR and UACI of the encrypted images. Table 12 lists the experimental results of the NPCR and UACI, and the experimental data show that all the test images pass the test.

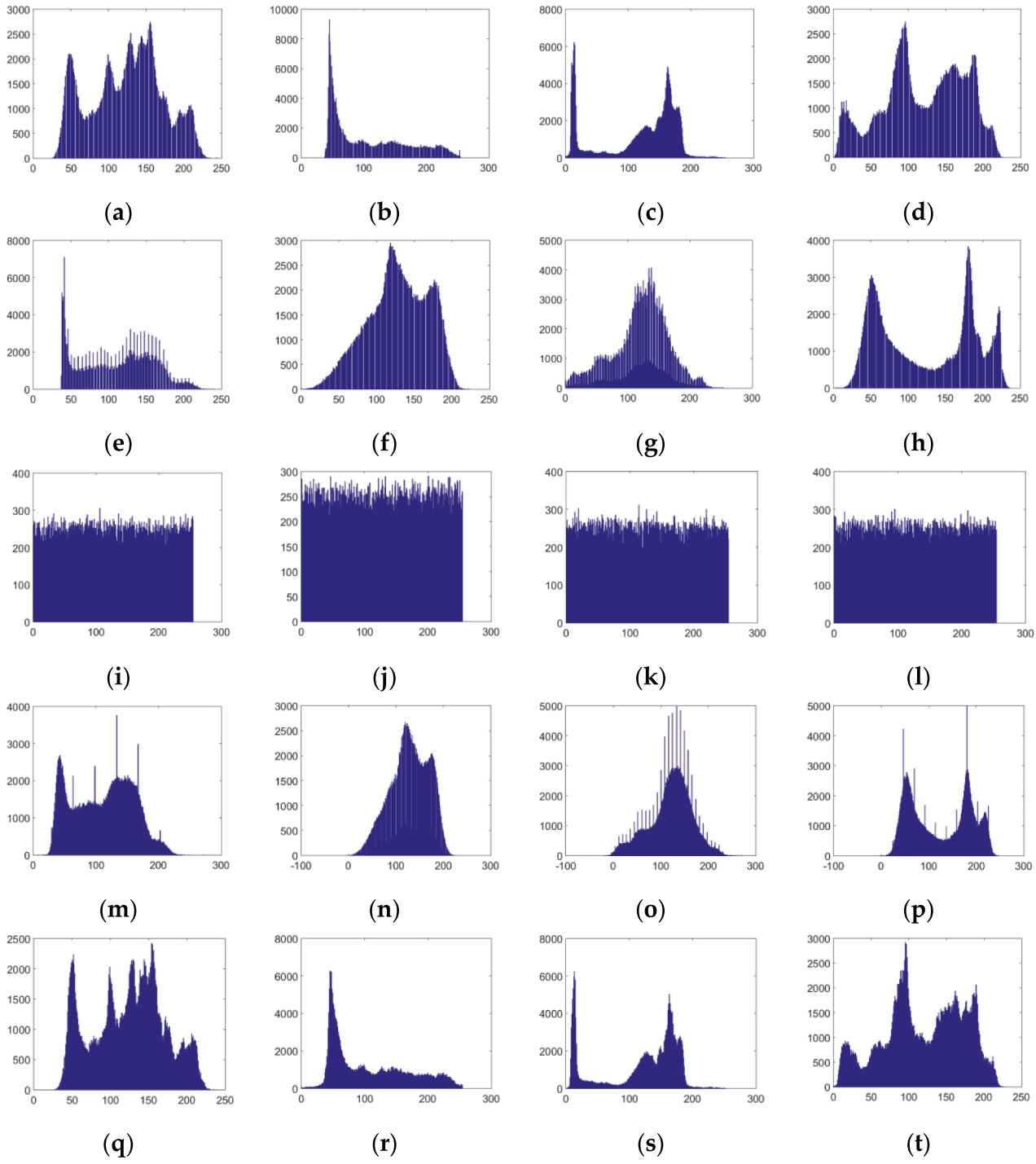

Fig. 21. Histogram analysis. (a)–(d) are the histograms of the plain images Lena, Woman, Cameraman, and Peppers. (e)–(h) are the histograms of the carrier images Private, Mandril, Livingroom, and Lake. (i)–(l) are the histograms of the corresponding secret images. (m)–(p) are the histograms of the corresponding cipher images. (q)–(t) are the histograms of the corresponding decrypted images.

Table 15
χ^2 values of the secret images.

Image	χ^2 value	Result
Lena	248.9375	Pass
Woman	260.4141	Pass
Cameraman	235	Pass
Peppers	246.7734	Pass

4.9. Key sensitivity

Another important indicator of the success of an encryption scheme is key sensitivity. A secure encryption system should be able to guarantee that with small changes to the key, the resulting secret image and decrypted image are as different as possible from the unaltered key. In order to quantify the change of the secret image before and after the key change, NPCR and UACI are introduced as the evaluation indicators. The proposed scheme has two sets of keys. By making slight changes to the two sets of keys in the encryption system, the NPCR and UACI values of the

Fig. 22. Decrypted images under different noise attacks.

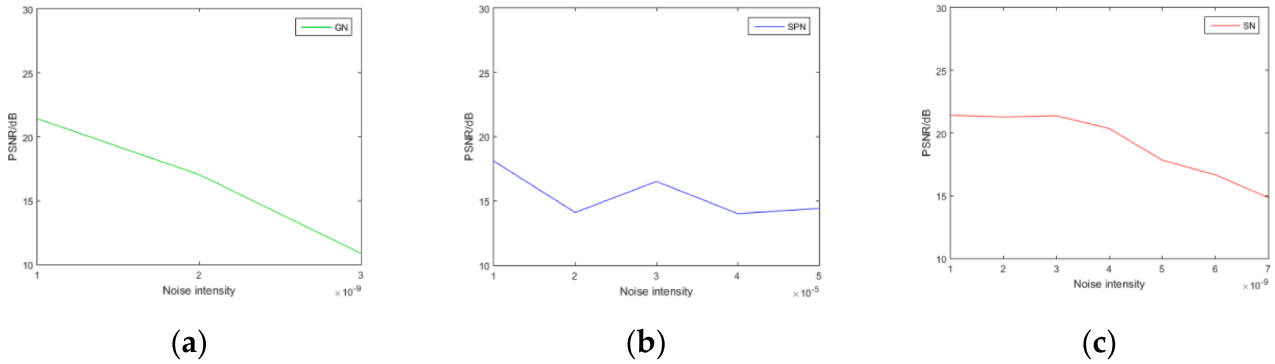

Fig. 23. PSNR values of decrypted images under different noise attacks.

secret image before and after the change are calculated. The first set of keys $K_{measurement} = (r_0, xx_0)$ is used to generate the measurement matrix. Let the wrong keys be $K1_{measurement} = (r_0 + 10^{-14}, xx_0)$, $K2_{measurement} = (r_0, xx_0 + 10^{-14})$. The second set of keys $K_{encryption} = (x_0, y_0, z_0, u_0, v_0, w_0)$ is used for the scrambling, diffusion, and embedding processes. Let the wrong keys be $K1_{encryption} = (x_0 + 10^{-14}, y_0, z_0, u_0, v_0, w_0)$, $K2_{encryption} = (x_0, y_0 + 10^{-14}, z_0, u_0, v_0, w_0)$, $K3_{encryption} = (x_0, y_0, z_0 + 10^{-14}, u_0, v_0, w_0)$, $K4_{encryption} = (x_0, y_0, z_0, u_0 + 10^{-14}, v_0, w_0)$, $K5_{encryption} = (x_0, y_0, z_0, u_0, v_0 + 10^{-14}, w_0)$, $K6_{encryption} = (x_0, y_0, z_0, u_0, v_0, w_0 + 10^{-14})$. Fig. 19 lists the key sensitivity test results of the encryption process. From the results, we can see that even if a parameter in the key changes very slightly, the obtained secret image is far from the

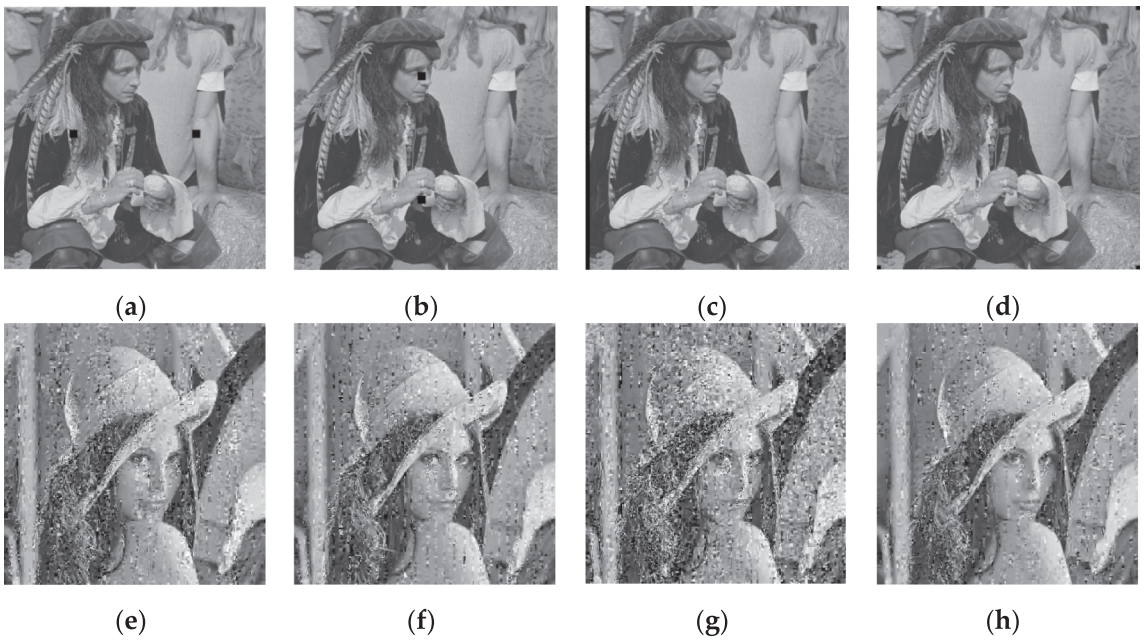

Fig. 24. Crop resistance analysis of the scheme. (a)–(d) are the cipher images of the cropped 16 × 16 × 2, 16 × 16 × 2, 512 × 8, and 8 × 8 × 4 regions. (e)–(h) are the decrypted images of (a)–(d).

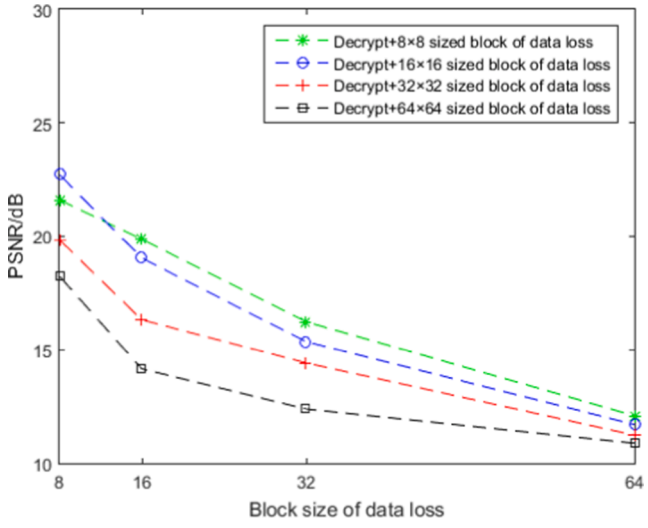

Fig. 25. PSNR vs B_3.

correct secret image. Table 13 and Table 14 list the sensitivity test results for the first and the second sets of keys. From the results, we can see that the NPCR values between the secret images generated by the correct key and the wrong key are between 99.55% and 99.65% and the UACI values are between 33.3% and 33.6%, indicating that the proposed scheme has high key sensitivity in the encryption process. In addition, we also tested the key sensitivity of the decryption process. Fig. 20 shows the test results. As can be seen from the results, the decrypted images obtained with the wrong keys are noise-like, and the relevant information of the original image cannot be obtained from them.

4.10. Histogram analysis

The histogram can reflect the frequency of each pixel. Many attackers often use the feature of histogram imbalance to conduct statistical attacks to obtain the original image. Therefore, a secure encryption system should be able to make the encrypted image histogram as uniform as possible to prevent attackers from obtaining relevant information from it. Fig. 21 lists the histograms of different images. It can be seen that the pixel distributions of the plain images and the decrypted images are very similar, indicating that the scheme can retain the information of the original image to a large extent. The pixel distributions of the secret images are relatively balanced; so, the information of the plain image cannot be effectively obtained from them. The carrier images and the cipher images are also very similar, indicating that the information is well-hidden and does not have a great impact on the carrier image.

The histogram can visually reflect the difference in pixel distribution, and the χ^2 test can quantify the pixel distribution [41]. For $M \times N$ 256-level grayscale images, the calculation formula of the χ^2 test is as follows.

$$\chi^2 = \sum_{i=0}^{255} \frac{(v_i - v_0)^2}{v_0}, v_0 = (M \times N)/256 \tag{89}$$

When the significance level is 0.05, the standard value of the χ^2 test is 293.2478. When the result is less than this value, it is considered to pass

Table 17
Running time of the encryption, embedding and extraction, and decryption processes.

Process	Encryption	Embedding and extraction	Decryption
Time(s)	21.732899	59.599079	23.407382

Table 16
Running time of specific processes.

Process	Process 1	Process 2	Process 3	Process 4	Process 5	Process 6	Process 7
Time(s)	0.048762	0.043251	1.305518	0.539084	0.017343	0.277327	0.611990

the test. From the results in Table 15, we can see that the χ^2 values of all secret image histograms are lower than the standard value, indicating that our scheme can resist statistical attacks.

4.11. Robustness analysis

During the propagation of cipher images, there may be noise and data loss, which hinders their decryption. Therefore, the robustness of an image encryption system is equally important.

4.11.1. Noise attack

Common noise types include Gaussian noise, salt-and-pepper noise, and speckle noise. In order to test the anti-noise ability of the proposed encryption scheme, we add different degrees of noise to the cipher image and then decrypt them. Fig. 22 lists the decrypted images of Lena under different types and intensities of noise attacks, and Fig. 23 lists the PSNR values of the decrypted images under different noise attacks. It can be seen that with the increase of noise intensity, the PSNR value of the decrypted images polluted by Gaussian noise decreases the fastest; the PSNR value of the decrypted images polluted by speckle noise is relatively flat; and the PSNR value of the decrypted images polluted by salt-and-pepper noise presents alternating rise and fall, which shows that the proposed encryption scheme has the strongest resistance to salt-and-pepper noise followed by speckle noise and followed by the weakest resistance to Gaussian noise.

4.11.2. Crop attack

In order to test the resistance of the encryption scheme to data loss, we cut part of the cipher image and decrypt the cut images to obtain the decrypted images. We select the middle position, the edge position, and the four corner positions to crop the $16 \times 16 \times 2$, 512×8, $8 \times 8 \times 4$ regions, respectively. The test results are shown in Fig. 24.It can be seen that within a certain range, after the cipher images lose some information, the decrypted images still retain most of the information.

4.11.3. Effect of block size on decrypted images after data loss

In the process of embedding the secret image into the carrier image, we divide the secret image and the carrier image into blocks so that the scale of the whole embedding process is greatly reduced. In order to test the effect of block size B_3 on the decrypted image obtained after the crop attack, we set four different block sizes of 8, 16, 32, and 64. That is, set the clipping area size of 8×8, 16×16, 32×32, 64×64, and cut the cipher image, and then, decrypt the cut images to obtain the PSNR values of the decrypted images. Fig. 25 presents the experimental results. It can be seen that setting the block size B_3 to 8 can achieve better results. When more data is lost, the gap between different block sizes narrows.

4.12. Running time efficiency analysis

Time consumption is an important indicator to evaluate an encryption scheme. The proposed scheme can be roughly divided into the following specific processes: optimized matrix scrambling (Process 1), sample number allocation (Process 2), measurement matrix optimization and measurement (Process 3), pixel-level scrambling (Process 4), bit-level scrambling (Process 5), diffusion (Process 6), and reconstruction of the decryption process (Process 7). From the running time of each specific process listed in Table 16, it can be seen that the optimization of the measurement matrix and the process of measuring the image take a long time. What is more, the bit-level scrambling algorithm in the proposed encryption scheme has high complexity. Firstly, all elements in the image matrix are expanded into 8-bit binary numbers, then all elements are scrambled by cyclic shift of row vectors and column vectors. Where the time complexity of the cyclic shift of the row vectors is $\Theta(SR \times m \times n \times 8 \times 8 \times n)$, the time complexity of the cyclic shift of the column vectors is $\Theta(SR \times m \times n \times 8 \times SR \times m)$, so the total time complexity is $\Theta(SR \times m \times n \times 8 \times (8n + SR \times m))$. From the time statistics of the entire encryption process in Table 17, we can see that the iteration of the chaotic system and the generation process of the random sequences greatly increase the time consumption. One of the major reasons is that the construction of the measurement matrix needs to be determined according to the number of samples after the image is divided into blocks, which increases the number of random numbers generated by the chaotic system when the specific number of samples is uncertain. In addition, the experimental results in Table 17 also show that the embedding process and extraction process in this study take a long time, which is due to the slow convergence rate of NMF. Especially to obtain higher decomposition accuracy, the number of iterations will increase significantly, making the process time-consuming. In real-time interaction, we can appropriately adjust the relevant parameters in the encryption method to shorten the encryption time. For example, we can reduce the decomposition level of image sparsity, change the threshold for filtering for pixel-level scrambling, adjust the block size of image in the encryption process, and so on. These operations can effectively reduce the encryption time and ensure that the encryption and decryption effects will not be affected too much.

5. Conclusions

Here, we proposed an image encryption and hiding algorithm based on Tetrolet transform, ABCS, and NMF. The approach is as follows. First, Tetrolet transform is performed on the plain image, and the sparsity of the sparse matrix is optimized by performing matrix scrambling, which equalizes the sparsity in each block area of the image matrix. Then, the sampling number of the block areas is calculated according to the image information; the measurement matrix is constructed and optimized; and the optimized measurement matrix is used to compress the image. After that, the compressed image is scrambled and diffused, which completes the encryption process. Finally, the image information is embedded into the carrier image through NMF to obtain a visually secure cipher image. The experimental results show that our scheme achieves good results in both encryption and decryption and has good resistance to various attacks.

The scheme of this study introduces the Tetrolet transform and combines the basic encryption operations to design the scrambling and diffusion processes. The whole process is closely related to the information of the plain image so that the entire encryption system has a strong ability to resist chosen-plaintext and known-plaintext attacks. Based on the non-negative characteristics of the image matrix, this scheme embeds the encrypted image into the carrier image through NMF and uses the keys to generate the initial random matrix, which not only ensures the security of the embedding process but also achieves visual safety.

However, this scheme also has some disadvantages. To obtain better encryption and decryption performance, optimized operations are added to some intermediate processes, which inevitably causes additional time and storage space consumption. More encryption operations and a complex encryption process make the data more sensitive to noise and data loss, which affects the robustness of the scheme. We will continue to study more efficient encryption algorithms to reduce the complexity of the overall scheme and enhance the robustness of the scheme without affecting the encryption effect. The sparsity of the signal is the basis of the compressed sensing technology. Determining the appropriate sparsity can often make the reconstructed signal more similar to the original signal. The sparsity of the proposed scheme is artificially determined and not always optimal, which affects the reconstruction effect of the decrypted images. Therefore, adaptively determining the optimal sparsity according to the image content is an important research direction to better combine compressed sensing theory and image encryption. Like many image encryption schemes that focus on visual security, the information hiding and extraction processes of this scheme require the encryption and decryption parties to have a

common carrier image, and both choose to embed image information into the high frequency of the carrier image, which restricts the transmission of image information and has a degree of security risk. Therefore, how to overcome the limitation of this model will be one of our future research directions.

CRediT authorship contribution statement

Yuandi Shi: Conceptualization, Methodology, Software. **Rongrong Chen:** Validation, Formal analysis, Investigation, Resources, Data curation. **Donglin Liu:** Writing – original draft, Writing – review & editing, Visualization. **Bin Wang:** Supervision, Project administration, Funding acquisition.

Declaration of Competing Interest

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

Data availability

No data was used for the research described in the article.

Acknowledgments

This work is supported by 111 Project (No. D23006), the National Natural Science Foundation of China (Nos. 62272079, 61972266, 61802040), Liaoning Revitalization Talents Program (No. XLYC2008017), Natural Science Foundation of Liaoning Province (Nos. 2021-MS-344, 2021-KF-11-03, 2022-KF-12-14), the Postgraduate Education Reform Project of Liaoning Province (No. LNYJG2022493), the Dalian Outstanding Young Science and Technology Talent Support Program (No. 2022RJ08), the Innovation and Entrepreneurship Team of Dalian University (No. XQN202008).

References

[1] X. Chai, J. Fu, J. Zhang, D. Han, Z. Gan, Exploiting preprocessing-permutation–diffusion strategy for secure image cipher based on 3D Latin cube and memristive hyperchaotic system, Neural Comput. & Applic. 33 (16) (2021) 10371–10402.
[2] N. Rani, S.R. Sharma, V. Mishra, Grayscale and colored image encryption model using a novel fused magic cube, Nonlinear Dyn. 108 (2) (2022) 1773–1796.
[3] S.-X. Nan, X.-F. Feng, Y.-F. Wu, H. Zhang, Remote sensing image compression and encryption based on block compressive sensing and 2D-LCCCM, Nonlinear Dyn. 108 (3) (2022) 2705–2729.
[4] M.A.B. Farah, A. Farah, T. Farah, An image encryption scheme based on a new hybrid chaotic map and optimized substitution box, Nonlinear Dyn. 99 (4) (2019) 3041–3064.
[5] Z. Gan, X. Chai, X. Zhi, W. Ding, Y. Lu, X. Wu, Image cipher using image filtering with 3D DNA-based confusion and diffusion strategy, Neural Comput. & Applic. 33 (23) (2021) 16251–16277.
[6] L. Xiong, F. Yang, J. Mou, X. An, X. Zhang, A memristive system and its applications in red–blue 3D glasses and image encryption algorithm with DNA variation, Nonlinear Dyn. 107 (3) (2022) 2911–2933.
[7] Q. Yin, Y. Zheng, B. Wang, Q. Zhang, Design of Constraint Coding Sets for Archive DNA Storage, IEEE/ACM Trans. Comput. Biol. Bioinf. (2021).
[8] P. Wang, Z. Mu, L. Sun, S. Si, B. Wang, Hidden Addressing Encoding for DNA Storage, Front. Bioeng. Biotechnol. (2022).
[9] Z. Gan, X. Chai, J. Zhang, Y. Zhang, Y. Chen, An effective image compression–encryption scheme based on compressive sensing (CS) and game of life (GOL), Neural Comput. & Applic. 32 (17) (2020) 14113–14141.
[10] X. Chai, X. Fu, Z. Gan, Y. Zhang, Y. Lu, Y. Chen, An efficient chaos-based image compression and encryption scheme using block compressive sensing and elementary cellular automata, Neural Comput. & Applic. 32 (9) (2018) 4961–4988.
[11] P.K. Naskar, S. Bhattacharyya, D. Nandy, A. Chaudhuri, A robust image encryption scheme using chaotic tent map and cellular automata, Nonlinear Dyn. 100 (3) (2020) 2877–2898.
[12] N.A. Azam, I. Ullah, U. Hayat, A fast and secure public-key image encryption scheme based on Mordell elliptic curves, Opt. Lasers Eng. 137 (2021).
[13] B. Abd-El-Atty, A.M. Iliyasu, A. Alanezi, A.A. Abd El-latif, Optical image encryption based on quantum walks, Opt. Lasers Eng. 138 (2021).
[14] J. Chen, L. Chen, Y. Zhou, Cryptanalysis of a DNA-based image encryption scheme, Inf. Sci. 520 (2020) 130–141.
[15] M.S. Khoirom, D.S. Laiphrakpam, T. Themrichon, Cryptanalysis of multimedia encryption using elliptic curve cryptography, Optik 168 (2018) 370–375.
[16] X. Chai, H. Wu, Z. Gan, Y. Zhang, Y. Chen, K.W. Nixon, An efficient visually meaningful image compression and encryption scheme based on compressive sensing and dynamic LSB embedding, Opt. Lasers Eng. 124 (2020).
[17] X. Chai, H. Wu, Z. Gan, D. Han, Y. Zhang, Y. Chen, An efficient approach for encrypting double color images into a visually meaningful cipher image using 2D compressive sensing, Inf. Sci. 556 (2021) 305–340.
[18] Y. Luo, J. Lin, J. Liu, D. Wei, L. Cao, R. Zhou, Y. Cao, X. Ding, A robust image encryption algorithm based on Chua's circuit and compressive sensing, Signal Process. 161 (2019) 227–247.
[19] L. Zhu, H. Song, X. Zhang, M. Yan, T. Zhang, X. Wang, J. Xu, A robust meaningful image encryption scheme based on block compressive sensing and SVD embedding, Signal Process. 175 (2020).
[20] X. Wang, C. Liu, D. Jiang, A novel triple-image encryption and hiding algorithm based on chaos, compressive sensing and 3D DCT, Inf. Sci. 574 (2021) 505–527.
[21] D.L. Donoho, Compressed sensing, IEEE Trans. Inf. Theory 52 (2006) 1289–1306.
[22] H. Wang, D. Xiao, M. Li, Y. Xiang, X. Li, A visually secure image encryption scheme based on parallel compressive sensing, Signal Process. 155 (2019) 218–232.
[23] B. Zhang, Y. Liu, J. Zhuang, K. Wang, Y. Cao, Matrix permutation meets block compressed sensing, J. Vis. Commun. Image Represent. 60 (2019) 69–78.
[24] R. Monika, S. Dhanalakshmi, R. Kumar, R. Narayanamoorthi, Coefficient Permuted Adaptive Block Compressed Sensing for Camera Enabled Underwater Wireless Sensor Nodes, IEEE Sens. J. 22 (1) (2022) 776–784.
[25] H.S. Abed, H.N. Abdullah, Modified Gram Schmidt Based Chaotic Matrix for Compressive Sensing in Cognitive Radio, 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), 2021, pp. 1360-1365.
[26] Z. Gan, X. Chai, J. Bi, X. Chen, Content-adaptive image compression and encryption via optimized compressive sensing with double random phase encoding driven by chaos, Complex & Intelligent Systems 8 (3) (2022) 2291–2309.
[27] X. Chai, H. Wu, Z. Gan, Y. Zhang, Y. Chen, Hiding cipher-images generated by 2-D compressive sensing with a multi-embedding strategy, Signal Process. 171 (2020).
[28] X. Chai, J. Fu, Z. Gan, Y. Lu, Y. Zhang, An image encryption scheme based on multi-objective optimization and block compressed sensing, Nonlinear Dyn. 108 (3) (2022) 2671–2704.
[29] Z. Hua, K. Zhang, Y. Li, Y. Zhou, Visually secure image encryption using adaptive-thresholding sparsification and parallel compressive sensing, Signal Process. 183 (2021).
[30] W. Wen, Y. Hong, Y. Fang, M. Li, M. Li, A visually secure image encryption scheme based on semi-tensor product compressed sensing, Signal Process. 173 (2020).
[31] J. Krommweh, Tetrolet transform: A new adaptive Haar wavelet algorithm for sparse image representation, J. Vis. Commun. Image Represent. 21 (4) (2010) 364–374.
[32] L.d. d, s.h. s,, Learning the parts of objects by non-negative matrix factorization, Nature 401 (6755) (1999) 788–791.
[33] L. Yang, Q. Yang, G. Chen, Hidden attractors, singularly degenerate heteroclinic orbits, multistability and physical realization of a new 6D hyperchaotic system, Commun. Nonlinear Sci. Numer. Simul. 90 (2020).
[34] X. Huang, Y. Dong, G. Ye, Y. Shi, Meaningful image encryption algorithm based on compressive sensing and integer wavelet transform, Front. Comp. Sci. 17 (3) (2022).
[35] D. Wei, M. Jiang, A fast image encryption algorithm based on parallel compressive sensing and DNA sequence, Optik 238 (2021).
[36] C.M. Kumar, R. Vidhya, M. Brindha, An efficient chaos based image encryption algorithm using enhanced thorp shuffle and chaotic convolution function, Appl. Intell. 52 (3) (2021) 2556–2585.
[37] C. Yang, P. Pan, Q. Ding, Image Encryption Scheme Based on Mixed Chaotic Bernoulli Measurement Matrix Block Compressive Sensing, Entropy 24 (2) (2022).
[38] Y. Zhang, A. Chen, Y. Tang, J. Dang, G. Wang, Plaintext-related image encryption algorithm based on perceptron-like network, Inf. Sci. 526 (2020) 180–202.
[39] Z. Liang, Q. Qin, C. Zhou, An image encryption algorithm based on Fibonacci Q-matrix and genetic algorithm, Neural Comput. & Applic. (2022).
[40] Y. Wu, J.P. Noonan, S. Agaian, NPCR and UACI randomness tests for image encryption, Cyber journals: multidisciplinary journals in science and technology, Journal of Selected Areas in Telecommunications (JSAT) (2011) 31–38.
[41] X. Wang, N. Guan, J. Yang, Image encryption algorithm with random scrambling based on one-dimensional logistic self-embedding chaotic map, Chaos, Solitons & Fractals 150 (2021).