Color image encryption based on joint permutation and diffusion

Taiyong Li,* Jiayi Shi, and Duzhong Zhang
Southwestern University of Finance and Economics,
School of Economic Information Engineering, Chengdu, China

Abstract. Image encryption plays an essential role in the community of image security. Most existing image encryption approaches adopt a permutation–diffusion scheme to permute pixel positions and change pixel values separately. One limitation of this scheme is that it has a high risk of being cracked. To solve this problem, we propose an approach that jointly permutates and diffuses (JPD) the pixels in a color image for encryption. First, a 4D hyperchaotic system with two positive Lyapunov exponents is used to generate a sequence for almost all encryption procedures. To enhance security, the plain image's information is introduced to the hyperchaotic system's initial parameters. Then, the hyperchaotic sequence is used to permute and diffuse the pixels in images jointly. More specifically, two index matrices that determine which pixels will be permuted and diffused and one mask matrix that determines how the pixels will be diffused are generated by the hyperchaotic sequence. We test the proposed JPD with several popular images. Experimental results and security analysis demonstrate that the JPD is capable of resisting various types of attacks. Moreover, the JPD can accelerate the encryption process. All these indicate that the JPD is effective and efficient for color image encryption. © *2021 SPIE and IS&T* [DOI: 10.1117/1.JEI.30.1.013008]

Keywords: hyperchaotic system; permutation; diffusion; color image encryption; joint permutation and diffusion.

Paper 200552 received Aug. 13, 2020; accepted for publication Jan. 14, 2021; published online Feb. 10, 2021.

1 Introduction

In the era of the Internet and big data, hundreds of millions of images are produced and transmitted every day. It is a hot topic to prevent the images from access by unauthorized users in the field of privacy preservation and/or image security. A direct way is to encrypt the images to make them uninformative for unauthorized image holders while keep the original contents visible for authorized users by decrypting the images with keys. Therefore, image encryption has drawn much attention from researchers in the last decades. Since chaotic systems have many advantages: pseudo-randomness, synchronization, ergodicity, extreme sensitivity to initial values, and parameters, it is very suitable for encrypting images that are of bulky data, high redundancy, and high correlation. Chaos-based images have become more and more popular in last decades.[1–5]

Chaos-based image encryption usually employs chaotic sequences generated from chaotic systems to determine the rules of encryption.[6–12] There are many operations for image encryption, among which permutation and diffusion are two prevalent ones. Permutation is to change the pixel positions while diffusion is to change the pixel values. Most existing image encryption approaches include both permutation and diffusion for image encryption, and they usually perform encryption by permutation and then diffusion or vice versa. That is to say, they conduct image encryption via permutation and diffusion separately. Tian and Lu[13] used chaotic dynamic S-box to permute image pixels and then applied DNA sequences to diffuse the image. The experimental results demonstrated its good image encryption effect. To resist chosen/known plain-text attacks, a simple chaotic system with plaintext related parameters are used for both permutation and diffusion.[14] Chai et al.[15] introduced block permutation and diffusion operations to improve

*Address all correspondence to Taiyong Li, litaiyong@gmail.com

the speed of image encryption. Chen et al.[16] employed a pixel-swapping mechanism and a chaotic orbit perturbing mechanism in the phases of permutation and diffusion, respectively. Hua and Zhou[17] introduced image filtering to image encryption as a diffusion operation. The experimental results showed that filtering along with block permutation can significantly improve the performance of image encryption. After that, Li et al.[18,19] and Wu et al.[20] extended the filtering with variable kernel parameters and kernel shapes for diffusion. In addition to the pixel block-level, DNA-level, and pixel-level image encryption, bit-level image encryption shows good security performance.[21,22] Zhan et al.[23] presented an approach for image encryption by utilizing a hyperchaotic sequence, global bit-level permutation, and DNA-level diffusion. The extensive experiments demonstrated the quality, security, and robustness of the approach.

Although the separate permutation and diffusion schemes have attracted so much attention from researchers, it was reported that they are of high risk because they lead attackers to crack the two processes separately.[24–27] To address this issue, several researchers studied how to permute and diffuse images simultaneously. Liu et al.[24] proposed a color image encryption scheme that uses Hopfield chaotic neural network (NN) to generate a diffusion matrix and then conducts permutation and diffusion simultaneously several rounds. Diab[25] put forward a novel simultaneous permutation and diffusion image encryption to process the pixels in a dynamic order fashion. In the simultaneous permutation and diffusion approach proposed by Liu et al.,[27] the authors associated Sine–Sine chaotic map's initial values with both the secret keys and the corresponding pixel values and then performed row-level and column-level encryption. The results show that the approach can resist various attacks and is fast. Compared with the existing image encryption techniques, simultaneous permutation and diffusion are still in its infancy.

To improve the performance of image encryption, this paper proposes a simple but effective scheme for color image encryption based on a hyperchaotic system and joint permutation and diffusion (JPD). A 4D hyperchaotic system with 11 terms, 2 positive Lyapunov exponents (LEs), and wide attractors are employed to generate a hyperchaotic sequence. To enhance security, the JPD introduces the information of the plain color image into the initial keys for the hyperchaotic system. So each image has its own initial keys for the hyperchaotic sequence to resist known-plaintext and chosen-plaintext attacks. With the hyperchaotic sequence, the JPD jointly permutes and diffuses the color image to obtain the corresponding cipher image.

The main contributions of the proposed JPD are three-fold: (1) a newly proposed hyperchaotic system is introduced to generate the hyperchaotic sequence; (2) a novel joint permutation and diffusion scheme is proposed to encrypt color images; and (3) extensive experiments and analysis demonstrate that the proposed JPD is fast and can resist various types of attacks.

The rest of our paper is structured as follows. In Sec. 2, we introduce the used 4D hyperchaotic system, the generation of the initial values for the hyperchaotic system, and the generation of the hyperchaotic sequence. In Sec. 3, the proposed JPD is described in detail. We report and analyze the experimental results in Sec. 4. Finally, Sec. 5 concludes the paper.

2 Preliminaries

2.1 *Used Hyperchaotic System*

Although some classical chaotic systems such as Hénon map, Logistic map, and Tent map have simple mathematical forms and are easy to implement, they may suffer from small key spaces, predictable orbits, limited ranges, and so on. Therefore, we employ a newly proposed hyperchaotic system to generate chaotic sequences for encryption. It can be formulated as Eq. (1):[28]

$$\begin{cases} \dot{x} = a(y-x) + w \\ \dot{y} = bx - xz + w \\ \dot{z} = xy - z - w \\ \dot{w} = -c(x+y) \end{cases}. \tag{1}$$

In the equation, a, b, and c are the positive constants, and x, y, z, and w are the state variables. The hyperchaotic system is solved by the fourth-order Runge–Kutta method. When the constants $(a,b,c) = (10,76,3)$ and initial values $IV = (x_0, y_0, z_0, w_0) = (0.4549, 0.3490, 0.2275, 0.2275)$,

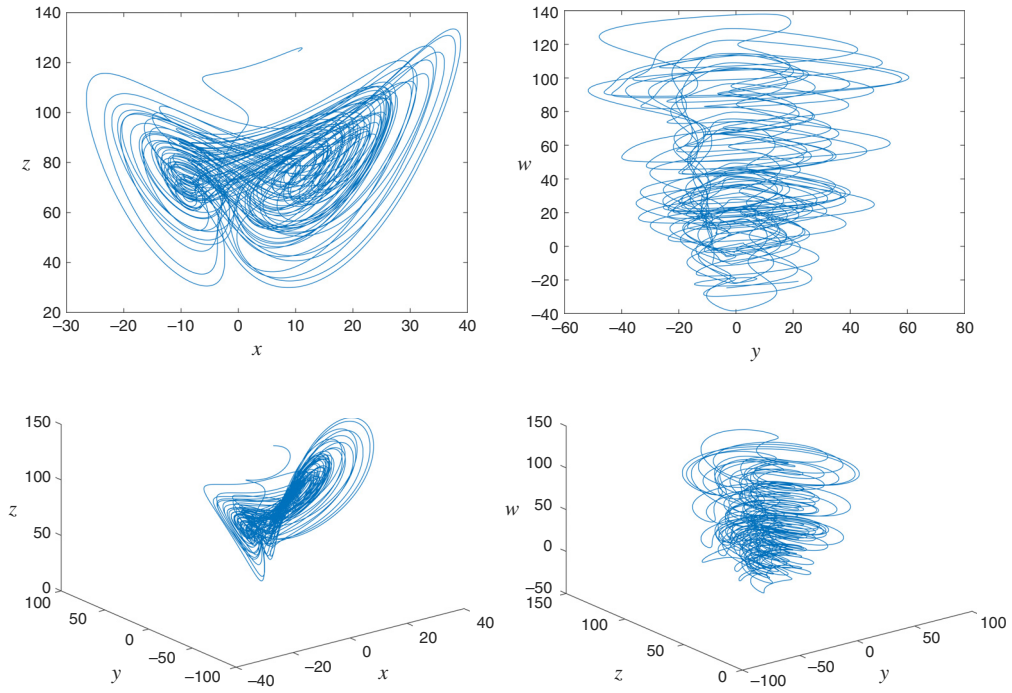

Fig. 1 The attractors of the used 4D hyperchaotic system. The first row shows $x-z$ plane and $y-w$ plane, from left to right. The second row shows $x-y-z$ space and $y-z-w$ space, from left to right.

the attractors of the hyperchaotic system are shown in Fig. 1, and it has two positive LEs: $L_1 = 1.5146$ and $L_2 = 0.2527$.[28]

The reasons for choosing this hyperchaotic system are summarized as follows: (1) It has 11 terms and 4D, but the form is still simple and it is easy to implement by hardware or software. (2) Its chaotic ranges are wider than some low-dimensional maps, so it has a larger secret key space. (3) It has two positive LEs and exhibits hyperchaotic attributes, making it hard to crack.

2.2 *Generation of Initial Values for the Hyperchaotic System*

To improve the ability of resisting known-plaintext and chosen-plaintext attacks, we introduce the hash value of the plain image to generate the initial values for the hyperchaotic system. In this way, each image has a set of specific initial values. After calculating the hash value from the plaintext image by SHA-256 algorithm, we divide the hash value to four parts to generate the initial values of the 4D hyperchaotic system. The detailed procedure can be described as follows:

Step 1: Calculate the SHA-256 hash value of the plain image K.

Step 2: Divide K into 32 blocks, which can be expressed as $K = \{k_1, k_2, \cdots, k_{32}\}$, and each block contains an 8-bit integer.

Step 3: By using the integers obtained from Step 2, four intermediate parameters d_1, d_2, d_3, and d_4 can be calculated by Eq. (2)

$$\begin{cases} d_1 = b_1 + \frac{1}{256}(k_1 \oplus k_2 \oplus \cdots \oplus k_8) \\ d_2 = b_2 + \frac{1}{256}(k_9 \oplus k_{10} \oplus \cdots \oplus k_{16}) \\ d_3 = b_3 + \frac{1}{256}(k_{17} \oplus k_{18} \oplus \cdots \oplus k_{24}) \\ d_4 = b_4 + \frac{1}{256}(k_{25} \oplus k_{26} \oplus \cdots \oplus k_{32}) \end{cases}, \tag{2}$$

where b_1, b_2, b_3, and b_4 are user-defined parameters that can be treated as security keys, and $\oplus$ represents the bitwise XOR operation.

Step 4: The initial values x_0, y_0, z_0, and w_0 of the 4D hyperchaotic system can be obtained by d_1, d_2, d_3, and d_4 via Eq. (3)

$$\begin{cases} x_0 = \frac{mod((d_1+d_2+d_3)\times 10^8, 256)}{255} \\ y_0 = \frac{mod((d_2+d_3+d_4)\times 10^8, 256)}{255} \\ z_0 = \frac{mod((d_1+d_2+d_3+d_4)\times 10^8, 256)}{255} \\ w_0 = \frac{mod(mean(d_1+d_2+d_3+d_4)\times 10^8, 256)}{255} \end{cases}, \tag{3}$$

where mod and $mean$ denote the module and average operation, respectively.

2.3 *Hyperchaotic Sequence Generation*

We use the 4D hyperchaotic system introduced in Sec. 2.1 to generate the hyperchaotic sequence for encryption. To be specific, this procedure contains three steps:

Step 1: The 4D hyperchaotic system uses the initial values calculated in Eq. (3) (x_0, y_0, z_0, and w_0) to iterate to generate sequences long enough for the subsequent encryption operations. In the j'th iteration, it can obtain four state values described as $s^j = \{x_j, y_j, z_j, w_j\}$.
Step 2: Discard the state values of the first I_0 iterations to remove the adverse effects.
Step 3: After the iteration terminates, a hyperchaotic sequence S can be obtained by concatenating all the $s^j (j = 1,2, \cdots, N)$ as

$$\begin{aligned} S = \{s^1, s^2, \cdots, s^N\} &= \{x_1, y_1, z_1, w_1, \cdots, x_N, y_N, z_N, w_N\} \\ &= \{s_1, s_2, s_3, s_4, \cdots, s_{4N-3}, s_{4N-2}, s_{4N-1}, s_{4N}\}. \end{aligned} \tag{4}$$

3 JPD: Joint Permutation and Diffusion for Color Image Encryption

3.1 *Generation of the Auxiliary Matrices*

In the proposed scheme, two types of auxiliary matrices are required. One is used to determine which pixels are to be processed and the other is for diffusion. Given an image of size $h \times w$ (h and w denote the height and width, respectively), for the first purpose, we use four random sequences r_1, r_2, r_3, and r_4 cut from S to generate two index matrices I and T. The detailed steps are described below:

Step 1: Sort r_1, r_2, r_3, and r_4 by ascending order to get the sort index si_1, si_2, si_3, and si_4, respectively.
Step 2: Generate I and T using si_1, si_2, si_3, and si_4.

```
for i = 1 → h
    for j = 1 → w
        I(i, j) = si1(mod(i + si2(j) − 1, w) + 1);
        T(i, j) = si3(mod(i + si4(j) − 1, w) + 1);
    end for
end for
```

The obtained I and T have the same size as the plain image.[4]

Subsequently, we use another random sequence r_5 containing $h \times w$ values cut from S to calculate the mask matrix (M) for diffusion by Eq. (5):

$$M = reshape(mod((r_5 - \lfloor r_5 \rfloor) \times 2^{32}), 256), [h, w]), \tag{5}$$

where reshape is to transform the sequence into a matrix.

Figures 2 and 3 show the generation of the index matrices and the mask matrix respectively, where the images have the same height and width, i.e., $h = w = 4$.

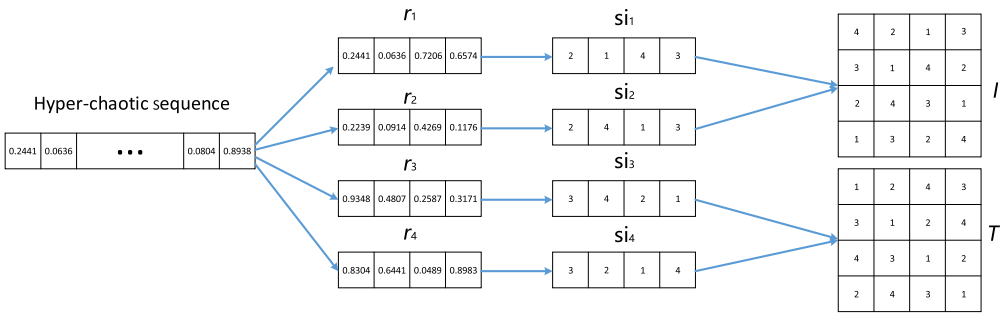

Fig. 2 An illustration of the generation of I and T.

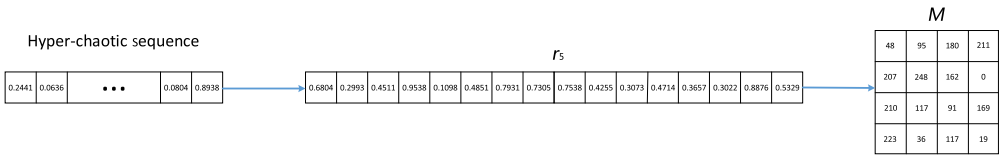

Fig. 3 An illustration of the generation of M.

3.2 *Joint Permutation and Diffusion*

The classical permutation and diffusion process can be easily cracked separately because the operations are independently performed. To solve this problem, we propose a novel JPD scheme to improve the security of the encryption algorithm. Given the index matrices I and T, the mask matrix M, and a plain image with one channel P, the JPD can be described as Eq. (6).

$$\mathbf{C}_{I_{i,j},j} = \begin{cases} mod(M_{i,j} \oplus (P_{T_{j,I_{i,j}},I_{i,j}} + P_{I_{h,w},w}), F), & \text{if } i = 1, j = 1 \\ mod(M_{i,j} \oplus (P_{T_{j,I_{i,j}},I_{i,j}} + C_{I_{i-1,w},w}), F), & \text{if } i \neq 1, j = 1 \\ mod(M_{i,j} \oplus (P_{T_{j,I_{i,j}},I_{i,j}} + C_{I_{i,j-1},j-1}), F), & \text{if } j \neq 1 \end{cases}, \tag{6}$$

where mod and $\oplus$ denote module operation and bitwise XOR operation, respectively. In the proposed JPD, the position of the pixel to be processed is determined by both I and T, whereas the value of the processed pixel is determined by the corresponding pixel in the mask M, the corresponding pixel in the plain image, and the previously processed pixel in the cipher image.

Here, we take an example to illustrate the operations of JPD in one round. Suppose both the height and width of the plain image P are 4, and the I, T, and M are identical to the corresponding matrices in Figs. 2 and 3. The illustration is shown in Fig. 4, where the cipher image is represented by C. The first three pixels to be processed (shown in green, purple and blue, respectively) by JPD are obtained by the following steps and the remaining pixels in C are directed given in the figure. The red arrows in C indicate the order of the pixels to be processed.

1. When $i = 1$ and $j = 1$, we have $I_{1,1} = 4$ and $T_{1,4} = 3$. The source pixel is at $P_{3,4}$, and the destination pixel $C_{4,1}$ is obtained by $C_{4,1} = mod(M_{1,1} \oplus (P_{3,4} + P_{4,4}), 256) = mod(48 \oplus (12 + 16), 256) = 44$.
2. When $i = 1$ and $j = 2$, we have $I_{1,2} = 2$ and $T_{2,2} = 1$. The source and the destination pixels are at $P_{1,2}$ and $C_{2,2}$ respectively, and $C_{2,2} = mod(M_{1,2} \oplus (P_{1,2} + C_{1,1}), 256) = mod(95 \oplus (2 + 44), 256) = 113$.
3. When $i = 1$ and $j = 3$, we have $I_{1,3} = 1$ and $T_{3,1} = 4$. The source and the destination pixels are at $P_{4,1}$ and $C_{1,3}$ respectively, the result of $C_{1,3}$ can be calculated by $mod(M_{1,3} \oplus (P_{4,1} + C_{2,2}), 256) = mod(180 \oplus (13 + 113), 256) = 202$.

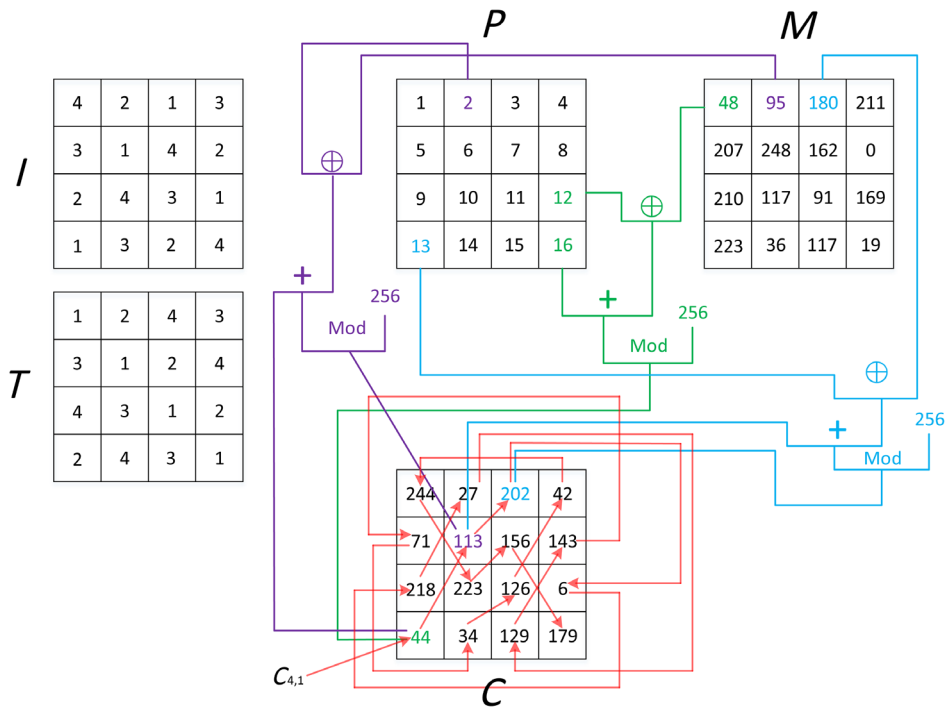

Fig. 4 An illustration of the JPD.

On the contrary, the decrypted image D can be obtained from I, T, M, and C by Eq. (7).

$$\mathbf{D}_{T_{j,I_{i,j}},I_{i,j}} = \begin{cases} mod(M_{i,j} \oplus C_{I_{i,j},j} - D_{I_{h,w},w}, F), & \text{if } i = 1,\ j = 1 \\ mod(M_{i,j} \oplus C_{I_{i,j},j} - C_{I_{i-1,w},w}, F), & \text{if } i \neq 1,\ j = 1 \\ mod(M_{i,j} \oplus C_{I_{i,j},j} - C_{I_{i,j-1},j-1}, F), & \text{if } j \neq 1 \end{cases}. \tag{7}$$

3.3 *Color Image Encryption using Joint Permutation and Diffusion*

Color images have three channels in the RGB color space. To encrypt color images, the simplest way is to encrypt each channel separately. The detailed JPD algorithm to encrypt a color image is described as Algorithm 1.

The JPD mainly consists of two parts: (1) generating the hyperchaotic sequence; and (2) performing JPD on each channel of the plain color image with the corresponding auxiliary matrices determined by the hyperchaotic sequence. In the first part, we use the pixel values of the plain color image to generate the initial values for the hyperchaotic system so that each plain color image has its own hyperchaotic sequence. In this way, the proposed JPD can resist known plaintext attacks. To perform encryption, the plain color image is first divided into three components, each of which is encrypted by the JPD according to the corresponding auxiliary matrices. Then the divided components are merged as the cipher image. To improve the effectiveness of encryption, the JPD can be performed several rounds. The JPD divides the task of color image encryption into several subtasks of encrypting single-channel images, and it is a typical strategy of divide and conquer. We also show the flowchart of the proposed JPD for color image encryption in Fig. 5.

For decryption, it only needs to run the steps in Algorithm 1 inversely.

3.4 *Discussion*

The proposed JPD has the following major characteristics.

First, a 4D hyperchaotic system with 11 terms, wide chaotic ranges, and two positive LEs is used to generate the hyperchaotic sequence for encryption. The hyperchaotic sequence is applied

Algorithm 1 Joint Permutation and Diffusion for a Color Image.

Input: plain color image P, the iteration number l_0 that produces the state values to be discarded, the round of encryption N, $keys = (b_1, b_2, b_3, b_4)$ for Eq. (2)

Output: Cipher Image C

function JPD(P, N, b_1, b_2, b_3, b_4)
  Generate the initial values $IV = \{x_0, y_0, z_0, w_0\}$ from P with b_1, b_2, b_3 and b_4, as described in Sec. 2.2;
  Generate the hyperchaotic sequence S with IV and l_0, as described in Sec. 2.3;
  Get the height (h), weight (w) and number of channels (c) by $h, w, c = size(P)$;
  $C \leftarrow zeros(h, w, c)$;
  $i \leftarrow 1$;
  for $n = 1 \rightarrow N$ **do**
    for $k = 1 \rightarrow c$ **do**
      Extract the pixels in the k'th channel of P to construct a plane p_k;
      Sort $S(i : i + w - 1)$ and $S(i + w : i + 2w - 1)$ to build the auxiliary matrix I, as described in Sec. 3.1;
      Sort $S(i + 2w : i + 3w + 1)$ and $S(i + 3w : i + 4w - 1)$ to build the auxiliary matrix T, as described in Sec. 3.1;
      Map $S(i + 4w : i + 4w + hw - 1)$ to the mask matrix M by Eq. (5) in Sec. 3.1;
      Perform JPD operations on p_k with I, T, and M to obtain a cipher plane e, as described in Sec. 3.2;
      $C(:, :, k) \leftarrow e$;
      $i \leftarrow i + 4w + hw$;
    end for
    $P \leftarrow C$;
  end for
  return C
end function

to almost all the procedures of encryption. The advantages of the used hyperchaotic system will benefit the encryption, such as a large key space, sensitivity to initial values, and being easy to implement.

Second, a plain color image's information is considered to generate the initial values of the hyperchaotic system, making each image have a unique hyperchaotic sequence. Different from some other encryption schemes that use the security keys as initial values of chaotic system only, the proposed JPD also considers the plain image's information to calculate the initial values. In this way, even for the same security keys, different initial values will be generated for different plain images and hence different images have different hyperchaotic sequences. This attribute makes the JPD have a good security level.

Third, three matrices are generated from the hyperchaotic sequence, and joint permutation and diffusion are conducted according to the matrices. In a general image encryption scheme that performs permutation and diffusion separately, it usually has to scan the image at least twice in one round process and the diffusion operation is usually conducted between two adjacent pixels. We know that in plain images, two adjacent pixels usually have a high correlation and hence the pixels in a cipher image may appear some specific regularity, which may lose some security. In the proposed JPD, the next pixel to be encrypted is decided by an index matrix and it may be

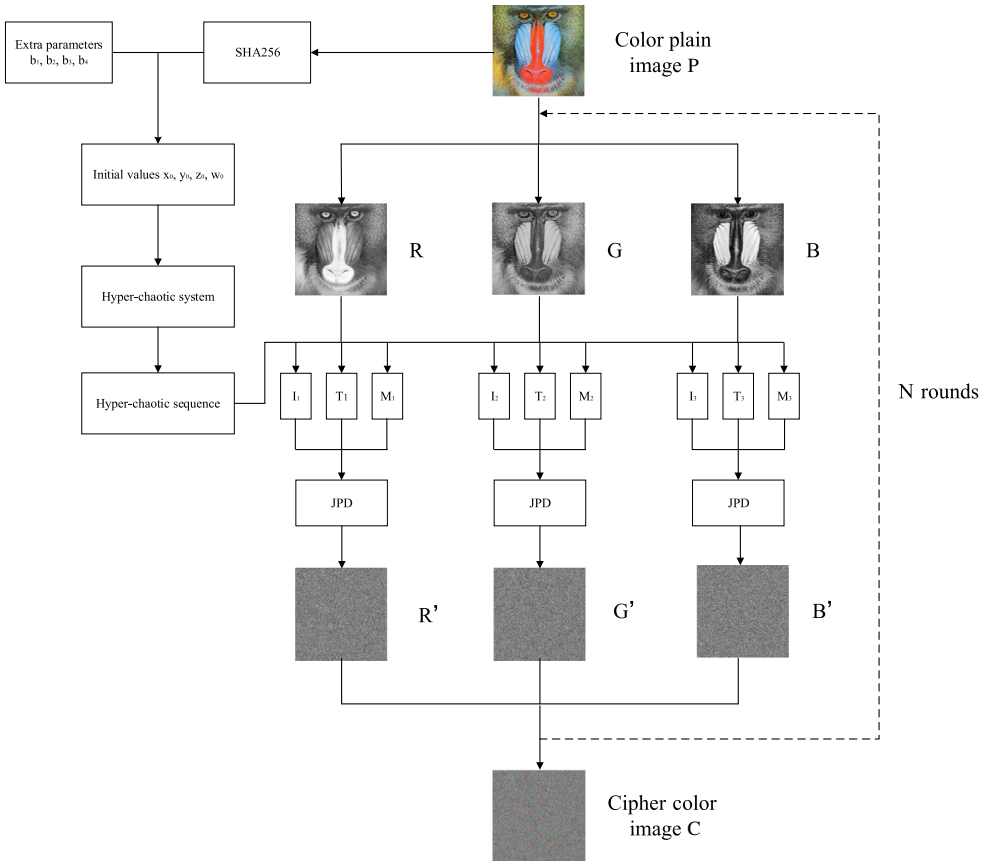

Fig. 5 The flowchart of the proposed JPD.

far from the current pixel and hence the correlation between the current pixel and the next one is usually low. At the same time, the JPD conducts permutation and diffusion jointly, it only needs to scan the whole image once in one round of encryption, and hence it reduces the time of encryption.

All these characteristics contribute to the effectiveness and efficiency of the proposed JPD.

4 Experimental Results and Security Analysis

4.1 *Experimental Settings*

The b_1, b_2, b_3, and b_4 in Eq. (2) are selected as security keys. We set I_0 in Sec. 2.3 to 500 to remove the adverse effects and run JPD 2 rounds for color image encryption. More specifically, the parameters used in the proposed JPD are listed in Table 1. To demonstrate the performance of the proposed JPD, we compare it with four state-of-the-art color image encryption schemes in most experiments: two schemes with Hopfield chaotic NN,[24,29] a scheme using the difference of

Table 1 Experiment parameters.

Parameter description	Value
Security keys	$b_1 = 1$, $b_2 = 1$, $b_3 = 2$, $b_4 = 2$
System parameters of the hyperchaotic system	$a = 10$, $b = 76$, $c = 3$
Iteration numbers of the discarded sequence	$I_0 = 500$
Encryption rounds	$N = 2$

Table 2 Testing images.

Image	Size ($h \times w \times c$)
Airplane	$512 \times 512 \times 3$
Baboon	$512 \times 512 \times 3$
Peppers	$512 \times 512 \times 3$
Sailboat	$512 \times 512 \times 3$
Splash	$512 \times 512 \times 3$

two 1D chaotic maps,[30] and a scheme using chaos to encrypt R, G, and B channels at the same time.[31]

Five popular RGB color images are used as testing images, as listed in Table 2. It is worth noting that all the images have identical sizes, i.e., $512 \times 512 \times 3$.

We conduct all the experiments by MATLAB R2019b on 64-bit Windows 10, and the main hardware includes an i7-4790 CPU @3.60 GHz CPU and 32 GB RAM. The source code of the JPD can be accessed at https://github.com/ltyong/jpd.

4.2 *Security Key Analysis*

The security keys are of importance for image encryption, which should have a large key space and be very sensitive for an ideal encryption scheme.

4.2.1 *Key space*

A well-designed encryption system should have an enough large key space to make brute-force attacks infeasible. In the proposed JPD, the user-defined parameters of b_1, b_2, b_3, and b_4 can be taken as the security keys. If each floating point number has a precision of 10^{-14}, the key space would be as large as $10^{14\times4} = 10^{56} \approx 2^{186}$. According to the existing research, if the size of key space of a cryptosystem is larger than 2^{100}, it can effectively resist brute-force attacks by modern computers.[32] So the proposed JPD has a large enough key space to resist brute-force attacks. In addition, the key space can be easily extended by considering the iteration times I_0, the number of encryption rounds, and/or the Hash value of the plain color image as security keys. For a specific plain color image, the security keys can be optimized by some evolutionary algorithms to achieve higher security.[33,34]

4.2.2 *Sensitivity to security keys*

For an ideal encryption scheme, it is an essential characteristic that the security keys are very sensitive for changes. That is to say, the correct keys can be used to fully recover the plain images while any tiny changes in the keys will result in completely different decrypted images. The keys of the JPD are associated with plain images. Here, we use the exact keys (b_1, b_2, b_3, b_4) of each test image and slightly different keys ($b_1 + 10^{-14}$, b_2, b_3, b_4) to decrypt the corresponding cipher image. The decrypted results are shown in Fig. 6. The images in the second row show that the tiny change (10^{-14}) in the security keys produce completely random-like images, and we cannot see any informative objects from them. Hence we can conclude that the proposed JPD is so sensitive that a tiny change of the keys cannot recover the correct color plain image.

4.3 *Statistical Analysis*

Typical statistical information includes information entropies, histograms, and correlations. And they can be used to crack encryption schemes. The cipher images by a well-designed encryption scheme should have high entropies, flat histograms, and low correlations.

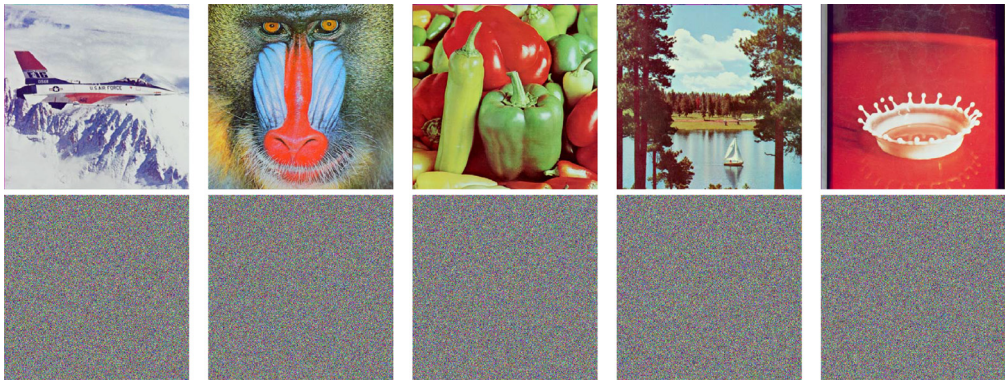

Fig. 6 Sensitivity to security keys. The first row and the second row are the decrypted images with the exact security keys (b_1, b_2, b_3, b_4) and the corresponding decrypted images with the keys ($b_1 + 10^{-14}$, b_2, b_3, b_4), respectively.

4.3.1 *Information entropy analysis*

Information entropy (IE) is a popular metric to measure the randomness and uncertainty of a complex system. For a single-channel C of a color image that has $2^8 = 256$ gray-scale levels, the IE of C can be formulated as Eq. (8):

$$IE(C) = -\sum_{g=0}^{255} p_g \log_2 p_g, \tag{8}$$

where p_g denotes the probability of gray-scale level g occurs in the whole channel. It can be concluded that when the channel has only 1 gray-scale level, it obtains the minimal value 0. If each gray-scale has an identical probability, i.e., $\frac{1}{256}$, the IE reaches its maximal value 8. For each channel of a natural color image, its IE is usually <8. Well-designed image encryption should make the pixel values of each channel of the cipher image distribute as evenly as possible so that its IE is very close to 8.

Table 3 gives the entropies of all channels of the plain color images and the corresponding cipher images by the proposed JPD and four compared schemes. The plain images' entropies are between 6.1265 and 7.7522, which are far below the maximum theoretical value 8. On the contrary, the entropies of all the cipher images concentrated between 7.9992 and 7.9994, which are very close to 8. All these results demonstrate that the pixel values of the cipher images distribute very evenly. Among the encryption schemes, JPD achieves the highest entropies in 6 out of 15 cases, which slightly underperforms Refs. 24 and 29 but outperforms Ref. 30.

4.3.2 *Histogram analysis*

A histogram can reflect the number of times each pixel occurs in an image. Generally speaking, the histogram of a natural image is irregular, and some shapes such as peaks and/or valleys exhibit in the histogram. A well-designed encryption scheme will break the shapes, and the histograms of cipher images should be as flat as possible. The histograms of the plain color images and the cipher images by the proposed JPD are shown in Fig. 7.

From this figure, we can see that the histograms of all the channels of the plain images are very different. For instance, the blue channel of Baboon distributes almost the whole range of pixel values while all channels of Airplane mainly distribute in a narrow range of pixel values around 200. When we investigate the histograms of all the channels of cipher images, we can find that they look almost the same. Particularly, the heights of all the bars in the histograms of cipher images are roughly the same, showing that the pixel values in each channel of the cipher images distribute very evenly. We can conclude that the proposed JPD can effectively resist histogram attacks.

Table 3 The IEs of the testing images.

Image	Channel	Input	Cipher images				
			JPD	Ref. 24	Ref. 29	Ref. 30	Ref. 31
Airplane	R	6.7178	7.9993	**7.9994**	**7.9994**	7.9993	7.9993
	G	6.7990	7.9993	7.9993	7.9993	7.9993	**7.9994**
	B	6.2138	**7.9993**	**7.9993**	**7.9993**	**7.9993**	7.9992
Baboon	R	7.7067	7.9992	**7.9994**	**7.9994**	7.9993	7.9993
	G	7.4744	**7.9994**	**7.9994**	7.9993	7.9993	7.9992
	B	7.7522	7.9992	7.9992	**7.9993**	7.9992	**7.9993**
Peppers	R	7.3388	7.9992	**7.9994**	7.9993	7.9992	**7.9994**
	G	7.4963	**7.9994**	7.9993	7.9992	7.9993	7.9993
	B	7.0583	**7.9994**	7.9993	7.9992	**7.9994**	7.9993
Sailboat	R	7.3124	7.9992	**7.9994**	7.9993	7.9993	7.9993
	G	7.6429	**7.9993**	**7.9993**	**7.9993**	7.9992	**7.9993**
	B	7.2136	7.9993	**7.9994**	**7.9994**	7.9993	**7.9994**
Splash	R	6.9481	**7.9993**	**7.9993**	**7.9993**	**7.9993**	**7.9993**
	G	6.8845	7.9993	7.9993	7.9993	**7.9994**	7.9993
	B	6.1265	7.9992	7.9993	**7.9994**	7.9992	7.9992

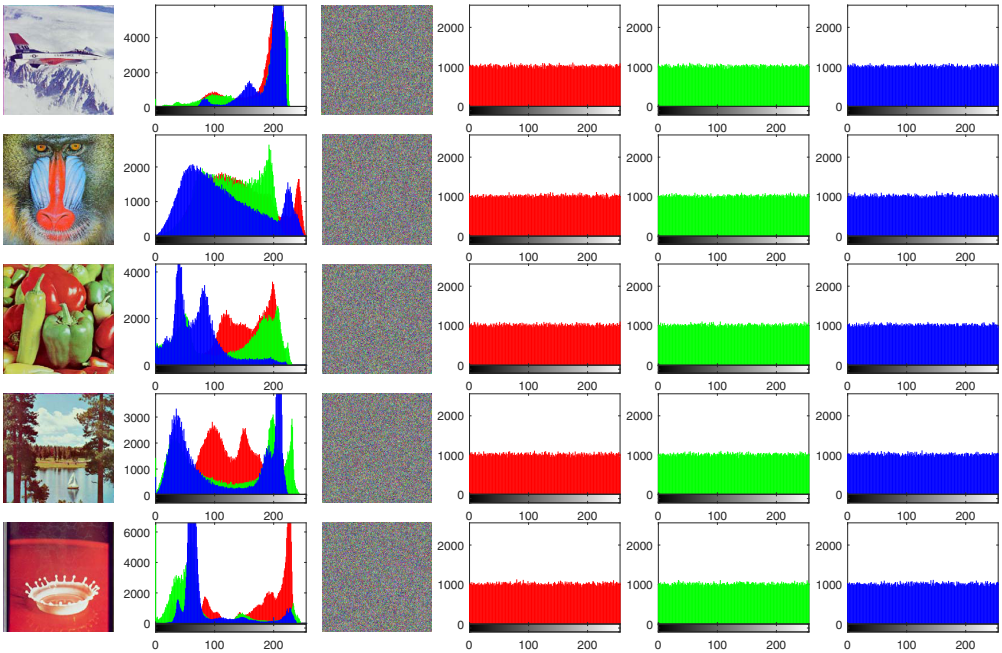

Fig. 7 Images and histograms. The first and the second columns are the plain images and their histograms, respectively. The third to the last columns are the corresponding cipher images and their histograms on R channel, G channel, and B channel, respectively.

4.3.3 *Correlation analysis*

Adjacent pixels in a natural image are usually very similar or even identical, so their correlations are usually very high. This attribute can be used to crack the image encryption system. Therefore, it is necessary for a well-designed encryption system to reduce the correlations existing in plain images. Given a sequence of pixels $X = \{x_1, x_2, \cdots, x_L\}$ and the sequence of its corresponding adjacent pixels $Y = \{y_1, y_2, \cdots, y_L\}$ in an image, the correlation γ between X and Y, denoted by $\gamma_{X,Y}$, can be computed by Eq. (9):

$$\begin{aligned} E(X) &= \frac{\sum_{i=1}^{L} x_i}{L} \\ D(X) &= \frac{\sum_{i=1}^{L} (x_i - E(X))^2}{L} \\ \gamma_{X,Y} &= \frac{\sum_{i=1}^{L} (x_i - D(X))(y_i - D(Y))}{L\sqrt{D(X)D(Y)}}, \end{aligned} \tag{9}$$

where $E(X)$ denotes X' mathematical expectation and $D(X)$ is its standard deviation. It is obvious that when X and Y are identical, $\gamma_{X,Y}$ achieves the maximal value 1. When X and Y have few correlation, $\gamma_{X,Y}$ will be very close to 0. An ideal encryption system should produce correlations as close to zero as possible.

Table 4 gives the correlations of the plain images and cipher images, where γ_h, γ_v, and γ_d represent the correlation at the horizontal, vertical, and diagonal directions, respectively. The lowest correlations are shown in bold for each case. It is worth noting that all the pixels in an image are involved when computing the correlations in this table. From the table, we can find that all the correlations of the plain images fall into [0.7299, 0.9946] that is close to 1, showing that there exist strong correlations in the plain images. In contrast, the correlations of all cipher images are very low and many are very close to 0, showing that the encryption schemes can break the strong correlations in the plain images. Specifically, the proposed JPD achieves the best correlations in 33 out of 45 cases, whereas Refs. 24 and 29–31 achieve the best values only in 6, 3, 1, and 3 cases, respectively. Furthermore, the JPD achieves the value 0.0000 3 times. All these demonstrate that the JPD can effectively break the correlations existed in plain images and it significantly outperforms the compared schemes in terms of reducing the correlations.

Taking it a step further, we plot the correlations of 2000 random pairs of adjacent pixels in the horizontal direction from both plain images and cipher images, as shown in Fig. 8. It is worth noting that for each image, the pixels in R, G, and B channels are plotted in Red, Green, and

Table 4 The correlation coefficients γ of the testing images.

				Cipher images				
Image	Channel		Input	JPD	Ref. 24	Ref. 29	Ref. 30	Ref. 31
Airplane	R	γ_h	0.9726	**0.0011**	−0.0056	0.0058	0.0041	0.0118
		γ_v	0.9507	**−0.0000**	−0.0151	0.0255	0.0046	−0.0014
		γ_d	0.9346	−0.0027	**0.0014**	0.0054	0.0086	−0.0027
	G	γ_h	0.9425	**−0.0004**	−0.0095	−0.0126	0.0041	−0.0098
		γ_v	0.9665	**−0.0014**	0.0133	−0.0035	−0.0086	−0.0072
		γ_d	0.9312	0.0005	0.0189	−0.0139	0.0105	**0.0002**
	B	γ_h	0.9633	**−0.0025**	**−0.0025**	0.0088	0.0156	0.0097
		γ_v	0.9162	**−0.0004**	−0.0159	−0.0051	0.0013	−0.0036
		γ_d	0.9110	0.0008	**−0.0001**	−0.0146	0.0013	0.0094

Table 4 (*Continued*).

				Cipher images				
Image	Channel		Input	JPD	Ref. 24	Ref. 29	Ref. 30	Ref. 31
Baboon	R	γ_h	0.9218	0.0033	**0.0018**	0.0060	−0.0073	0.0060
		γ_v	0.8624	−0.0013	0.0038	**−0.0003**	−0.0059	−0.0015
		γ_d	0.8531	**−0.0009**	−0.0016	−0.0122	−0.0136	−0.0291
	G	γ_h	0.8643	**0.0001**	−0.0013	0.0008	0.0046	0.0022
		γ_v	0.7591	**0.0020**	0.0176	0.0141	−0.0077	0.0122
		γ_d	0.7299	−0.0012	0.0040	0.0071	−0.0044	**0.0008**
	B	γ_h	0.9071	**0.0000**	−0.0112	−0.0004	−0.0067	−0.0138
		γ_v	0.8782	**0.0000**	0.0018	−0.0011	−0.0111	−0.0032
		γ_d	0.8411	**0.0004**	0.0082	−0.0115	0.0122	0.0217
Peppers	R	γ_h	0.9618	**0.0002**	0.0054	−0.0090	0.0142	0.0057
		γ_v	0.9640	**−0.0011**	−0.0042	−0.0226	0.0053	0.0121
		γ_d	0.9575	−0.0024	**−0.0017**	−0.0119	−0.0225	−0.0059
	G	γ_h	0.9777	−0.0016	−0.0055	**0.0006**	−0.0122	−0.0050
		γ_v	0.9771	−0.0004	0.0119	−0.0056	0.0195	**0.0000**
		γ_d	0.9698	**−0.0015**	0.0046	−0.0106	−0.0120	0.0025
	B	γ_h	0.9628	**−0.0005**	−0.0021	0.0027	0.0075	−0.0010
		γ_v	0.9619	**0.0001**	0.0104	−0.0044	−0.0173	0.0028
		γ_d	0.9478	**0.0014**	−0.0021	0.0030	0.0187	−0.0085
Sailboat	R	γ_h	0.9544	**−0.0006**	−0.0025	0.0076	−0.0092	0.0031
		γ_v	0.9529	**0.0011**	−0.0092	0.0066	0.0016	−0.0031
		γ_d	0.9396	−0.0023	−0.0095	**0.0008**	−0.0090	0.0138
	G	γ_h	0.9692	**−0.0007**	0.0124	0.0056	−0.0038	0.0097
		γ_v	0.9627	**−0.0001**	0.0102	0.0008	−0.0034	−0.0048
		γ_d	0.9520	**0.0010**	−0.0057	0.0029	−0.0268	0.0076
	B	γ_h	0.9690	**−0.0003**	−0.0073	0.0028	0.0033	−0.0105
		γ_v	0.9688	**0.0004**	0.0135	0.0111	0.0043	−0.0043
		γ_d	0.9521	**−0.0015**	0.0034	0.0055	0.0127	0.0081
Splash	R	γ_h	0.9936	**0.0021**	−0.0073	−0.0160	−0.0065	−0.0022
		γ_v	0.9946	**−0.0012**	0.0136	−0.0176	−0.0017	−0.0068
		γ_d	0.9893	−0.0007	**−0.0006**	0.0047	0.0028	−0.0089
	G	γ_h	0.9796	**0.0035**	−0.0066	−0.0124	0.0038	0.0039
		γ_v	0.9831	**−0.0040**	0.0202	0.0076	−0.0050	0.0083
		γ_d	0.9712	**0.0022**	−0.0075	−0.0075	−0.0035	−0.0089
	B	γ_h	0.9826	0.0045	−0.0026	0.0023	**0.0013**	0.0124
		γ_v	0.9700	**−0.0002**	0.0023	0.0026	0.0014	0.0173
		γ_d	0.9653	**0.0023**	−0.0045	0.0137	−0.0150	−0.0081

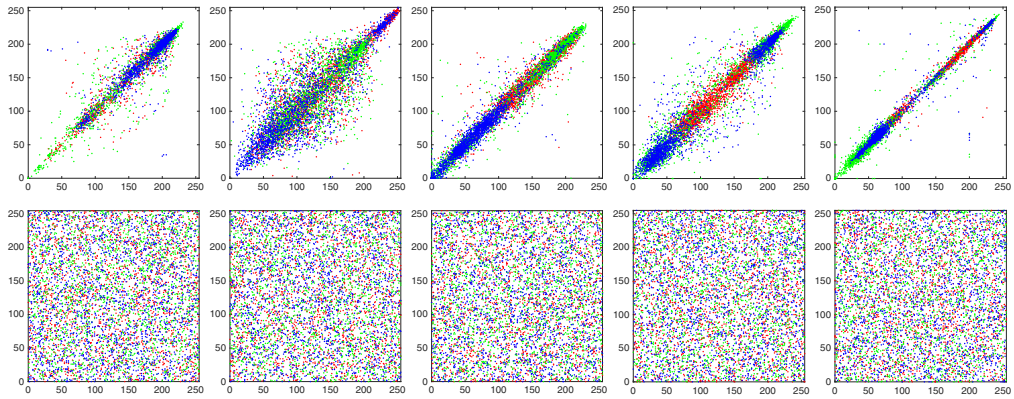

Fig. 8 Correlations of 2000 random pairs of adjacent pixels in the horizontal direction. The first row contains correlations of plain airplane, plain baboon, plain peppers, plain sailboat, and plain splash, from left to right. The second row contains correlations of cipher airplane, cipher baboon, cipher peppers, cipher sailboat, and cipher splash, from left to right.

Blue, respectively. All the results are produced by the proposed JPD. From the first row, we can find that most pairs of pixels, distribute along the diagonal, indicating the strong correlations exist in the plain images. However, from the subplots in the second row, we can see that the points fill in the whole planes, showing few correlations in the cipher images. This figure confirms that the JPD can effectively break the correlations existed in the plain images.

From the above analysis, we can see that the cipher images by the JPD have high entropies, flat histograms, and low correlations. Such attributes enable the proposed JPD to resist statistical attacks effectively.

4.4 *Analysis of Resisting Differential Attacks*

A differential attack is another type of attack by comparing the variations in the plain images with variations in the cipher images to find the plaintext or desired key. If a small variation in a plain image results in a small variation in the corresponding cipher image, it is easy to crack the encryption scheme. In contrast, if the small variation in the plain image produces a completely different cipher image, the encryption scheme is said to be able to resist differential attacks.

Two important indices, the number of pixels change rate (NPCR) and the unified average changing intensity (UACI), can be used to measure the ability to resist differential attacks, and they can be defined as Eqs. (10) and (11):

$$NPCR = \frac{\sum_{i=1}^{h}\sum_{j=1}^{w}\delta_{i,j}}{h \times w}, \tag{10}$$

$$UACI = \frac{\sum_{i=1}^{h}\sum_{j=1}^{w}|C_{i,j} - D_{i,j}|}{255 \times h \times w}, \tag{11}$$

where C and D are the two cipher images, $C_{i,j}$ denotes the pixel value at the position of (x, y) in C, and $\delta_{i,j}$ is used to indicate whether $C_{i,j}$ is identical to $D_{i,j}$ or not, which can be formulated as Eq. (12):

$$\delta(i,j) = \begin{cases} 0, & C_{i,j} = D_{i,j} \\ 1, & C_{i,j} \neq D_{i,j} \end{cases}. \tag{12}$$

According to the prior research,[35] given a significance level $\alpha = 0.05$ and a 256-level (8 bits) image with size of 512×512, if the NPCR is greater than the threshold $\mathcal{N}_{0.05} = 99.5893\%$ and the UACI falls into the range of $(\mathcal{U}^{l}_{0.05}, \mathcal{U}^{u}_{0.05}) = (33.3730\%, 33.5541\%)$, we can say that the encryption scheme passes the NPCR and UACI tests on the test image at the significance level $\alpha = 0.05$.

Table 5 The average NPCR (%) of running the schemes 10 times.

Image	Channel	JPD	Ref. 24	Ref. 29	Ref. 30	Ref. 31
Airplane	R	99.5943	**99.6063**	99.6010	99.6043	*99.5357*
	G	**99.6113**	99.6033	99.6017	99.6055	*99.5151*
	B	**99.6052**	99.6029	99.6014	99.6017	99.6044
Baboon	R	**99.6203**	99.6048	99.5964	99.6037	99.6020
	G	**99.6144**	99.6059	99.6048	99.6017	99.6059
	B	**99.6085**	99.6071	99.6046	99.6043	*99.5231*
Peppers	R	99.6060	99.5975	99.5899	**99.6065**	99.5937
	G	99.6022	**99.6052**	99.6040	99.6007	99.5907
	B	**99.6144**	99.6037	99.6001	99.6005	99.6085
Sailboat	R	99.5953	**99.6089**	99.5995	99.6063	99.6074
	G	**99.6238**	99.5983	99.5995	99.6009	99.6048
	B	99.6061	**99.6174**	99.6005	99.6090	99.6090
Splash	R	**99.6159**	99.6037	99.6003	99.5995	*99.5288*
	G	**99.6033**	99.6020	99.6020	99.6027	*99.5804*
	B	**99.6397**	99.6082	*99.5859*	99.5903	*99.5174*

We first add 1 to a random pixel in the plain image, and then apply an encryption scheme to it to produce a new cipher image D. With the normal cipher image C and the newly produced D, we can compute the NPCR and UACI for one time. To reduce the influence of the random pixel, we repeat the operations 10 times, and the results of the average NPCR and UACI are reported in Tables 5 and 6, respectively. In the tables, we italicize the values to figure out the results that fail the NPCR or UACI test, and we highlight the highest values in bold for each channel of the test images.

When we look at the average NPCR, it can be found that the JPD, Refs. 24 and 30 pass the tests in all cases, whereas Refs. 29 and 31 fail in 1 and 6 out of 15 cases, respectively. Furthermore, the proposed JPD achieves the highest NPCR 10 times, followed by Ref. 24's 4 times. Reference 30 achieves the highest value once while Refs. 29 and 31 never achieve the highest value. Therefore, the proposed JPD significantly outperforms the compared schemes in terms of NPCR.

Regarding the average UACI in Table 6, the values of all the channels associated with the proposed JPD fall into the range of $(33.3730\%, 33.5541\%)$, whereas those of Refs. 24 and 29–31 fall within the range only 6, 6, 2, and 1 times, respectively, showing the compared schemes perform poorly in UACI. What's more, the JPD achieves the highest values 11 out of 15 times, followed by Ref. 29's 3 times. References 24 and 30 completely fail in the competition of achieving the highest values. It is worth noting that although Ref. 31 achieves the highest value 34.1874 with the R channel of Splash, it is out of the scope of $(\mathcal{U}^l_{0.05}, \mathcal{U}^u_{0.05})$ and it still fails the UACI test.

From the above analysis, we can see that the proposed JPD exhibits superpower in NPCR and UACI when compared with the other schemes, and it can pass all the tests at a significance level 0.05. Therefore, the proposed JPD can effectively resist against differential attacks.

4.5 *Robust Analysis*

Data loss and noise are unavoidable for cipher images during transmission and storage. Different from the fact that small changes in plain images result in completely different cipher images,

Table 6 The average UACI (%) of running the schemes 10 times.

Image	Channel	JPD	Ref. 24	Ref. 29	Ref. 30	Ref. 31
Airplane	R	**33.4866**	*31.9665*	*32.0240*	*32.0454*	*32.0379*
	G	**33.4360**	*33.1449*	*33.0564*	*32.9673*	*33.1928*
	B	**33.4773**	*32.7264*	*32.7064*	*32.7126*	*32.3964*
Baboon	R	**33.3946**	*29.9931*	*29.9672*	*29.9630*	*29.9583*
	G	**33.5394**	*28.5822*	*28.6076*	*28.5708*	*28.5502*
	B	**33.4437**	*31.2384*	*31.2565*	*31.2574*	*31.2465*
Peppers	R	**33.4317**	*29.0588*	*29.0127*	*29.0251*	*28.9452*
	G	**33.4702**	33.4382	33.4632	*33.3119*	33.4000
	B	33.4245	33.4001	**33.4603**	*33.3467*	*33.3062*
Sailboat	R	**33.4729**	*27.9264*	*27.9147*	*27.9384*	*27.9175*
	G	33.4602	33.4203	**33.4633**	33.3784	*33.3571*
	B	**33.4857**	33.4025	33.4628	33.3974	*33.3570*
Splash	R	33.4676	33.4427	33.4630	*33.2030*	***34.1874***
	G	33.4493	33.4605	**33.4632**	*33.2232*	*33.2123*
	B	**33.4227**	*31.9747*	*31.8571*	*31.9189*	*31.9259*

contamination in cipher images should not have great impacts on restoring images. A feasible encryption scheme should be robust enough to decrypt contaminated cipher images to some extent.

To demonstrate the robustness of the proposed JPD, we first crop 0.4%, 1.6%, 4%, and 25% pixels at the center of cipher Baboon, and then decrypt the contaminated cipher Baboon using the proposed JPD. The cropped images and decrypted images are shown in Fig. 9. We can see that when the data loss ratio is not greater than 4%, the JPD can restore Baboon very well, and even if it equals 25%, the decrypted image still preserves most visual information of the plain Baboon. Then, we add 0.5%, 1%, 2%, and 5% salt and pepper noise to the cipher Baboon and decrypt

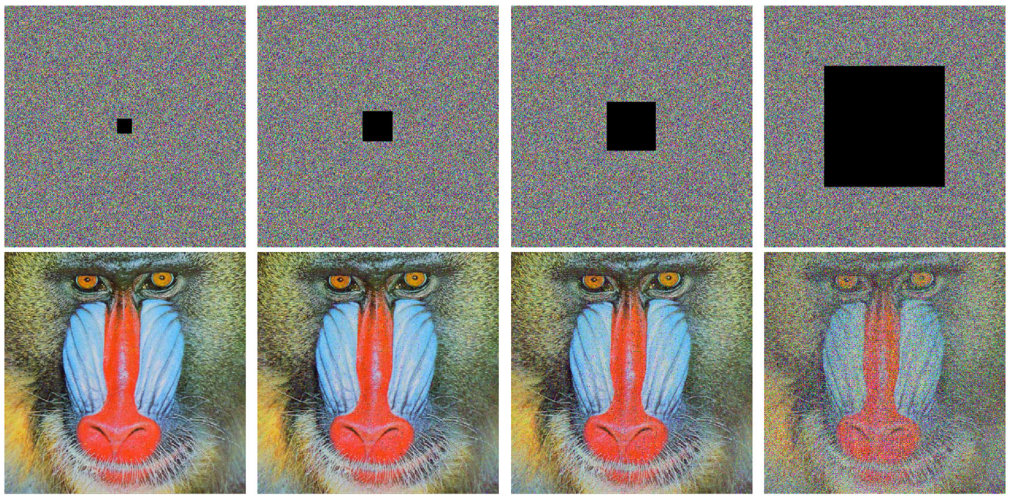

Fig. 9 Cropping attack results. The first row: cipher images with 0.4%, 1.6%, 4%, and 25% data loss. The second row: the decrypted images from the first row.

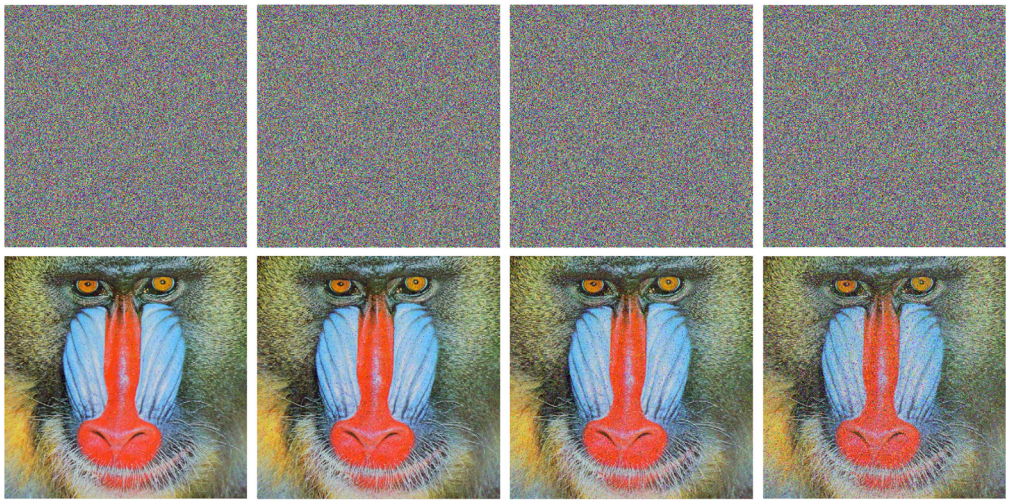

Fig. 10 Noise attack results. The first row: cipher images with 0.5%, 1%, 2%, and 5% salt and pepper noise. The second row: the decrypted images from the first row.

them with the proposed JPD. The results are shown in Fig. 10. Once again, we can observe that the decrypted images are very close to the plain Baboon for 0.5%, 1%, and 2% noise. And even for 5% noise, the decrypted Baboon is still visible.

Based on the analysis, we can conclude that the proposed JPD is robust and can effectively resist cropping attacks and noise attacks.

4.6 *Running Time*

In the proposed JPD, each pixel in each channel of an color image will be processed once in one round. For a given RGB color image with the size of $N \times N \times 3$, suppose we conduct the encryption 2 rounds, the time complexity of the proposed JPD is $6N^2$. It is equal to that of Ref. 24 and is better than that of some other state-of-the-art encryption schemes,[9,36,37] which owes to the fact that the proposed JPD can permutate and diffuse the images jointly rather than separately. In our experimental environment, it takes about 2.1224 s to encrypt an RGB color image of size $512 \times 512 \times 3$ for the JPD and the corresponding decryption time is about 2.1274 s.

4.7 *Summary*

The above-experimental results demonstrate that the proposed JPD can effectively resist brute-force attacks, statistical attacks, differential attacks, and noise and cropping attacks. In addition since plain color images information is introduced into the initial values of the hyperchaotic systems, each image has its own hyperchaotic sequence and the proposed JPD can easily resist known-plaintext attacks. The complexity and running time shows that the JPD is efficient for color image encryption. All these experimental results benefit from the characteristics of the JPD: (1) a hyperchaotic system that generates a hyperchaotic sequence for subsequent encryption operations; (2) introduction of plain color images' information to the initial values of the hyperchaotic system; and (3) a joint permutation and diffusion scheme for encryption.

5 Conclusions

Permutation and diffusion are two typical types of operations for image encryption. Most existing encryption schemes consider these two operations separately. It may increase the risk of being cracked because it is usually easier to crack a system with two separate operations than to crack the mixture of the operations. At the same time, two separate operations increase the times of scanning pixels and hence reduce the efficiency of encryption. To cope with these issues, this paper proposes a simple yet effective encryption scheme for color images using joint

permutation and diffusion. More specifically, the scheme first uses a hyperchaotic system to generate three auxiliary matrices, including two index matrices and one mask matrix, to decide which pixels will be processed and how they will be processed. Then, a practical joint permutation and diffusion operation is proposed to encrypt color images based on the auxiliary matrices. The experimental results demonstrate that the proposed scheme can effectively resist types of attacks and outperforms the compared state-of-the-art encryption schemes, showing it is promising for color image encryption. We will study how to accelerate the proposed scheme using GPUs in the future.

Acknowledgments

This work was supported by the Fundamental Research Funds for the Central Universities (Grant No. JBK2003001), the Ministry of Education of Humanities and Social Science Project (Grant No. 19YJAZH047), and the Scientific Research Fund of Sichuan Provincial Education Department (Grant No. 17ZB0433). The authors declare that they have no conflicts of interest.

References

1. G. Chen, Y. Mao, and C. K. Chui, "A symmetric image encryption scheme based on 3D chaotic cat maps," *Chaos Solitons Fractals* **21**(3), 749–761 (2004).
2. N. K. Pareek, V. Patidar, and K. K. Sud, "Image encryption using chaotic logistic map," *Image Vision Comput.* **24**(9), 926–934 (2006).
3. F. Ozkaynak, "Brief review on application of nonlinear dynamics in image encryption," *Nonlinear Dyn.* **92**, 305–313 (2018).
4. Z. Hua, Y. Zhou, and H. Huang, "Cosine-transform-based chaotic system for image encryption," *Inf. Sci.* **480**, 403–419 (2019).
5. S. Zhu and C. Zhu, "Secure image encryption algorithm based on hyperchaos and dynamic DNA coding," *Entropy* **22**(7), 772 (2020).
6. L. Zhang, X. Liao, and X. Wang, "An image encryption approach based on chaotic maps," *Chaos Solitons Fractals* **24**(3), 759–765 (2005).
7. T. Li et al., "A novel image encryption algorithm based on a fractional-order hyperchaotic system and DNA computing," *Complexity* **2017**, 9010251 (2017).
8. H. Bouslehi and H. Seddik, "Innovative image encryption scheme based on a new rapid hyperchaotic system and random iterative permutation," *Multimedia Tools Appl.* **77**, 30841–30863 (2018).
9. G. Ye et al., "An efficient pixel-level chaotic image encryption algorithm," *Nonlinear Dyn.* **94**, 745–756 (2018).
10. J. Liu et al., "A new simple chaotic system and its application in medical image encryption," *Multimedia Tools Appl.* **77**, 22787–22808 (2018).
11. C. Chen, K. Sun, and Q. Xu, "A color image encryption algorithm based on 2D-CIMM chaotic map," *China Commun.* **17**, 12–20 (2020).
12. H. Li, L. Deng, and Z. Gu, "A robust image encryption algorithm based on a 32-bit chaotic system," *IEEE Access* **8**, 30127–30151 (2020).
13. Y. Tian and Z. Lu, "Novel permutation-diffusion image encryption algorithm with chaotic dynamic S-box and DNA sequence operation," *AIP Adv.* **7**(8), 085008 (2017).
14. L. Huang et al., "A simple chaotic map-based image encryption system using both plaintext related permutation and diffusion," *Entropy* **20**, 535 (2018).
15. X. Chai, Z. Gan, and M. Zhang, "A fast chaos-based image encryption scheme with a novel plain image-related swapping block permutation and block diffusion," *Multimedia Tools Appl.* **76**(14), 15561–15585 (2017).
16. J. Chen et al., "An improved permutation-diffusion type image cipher with a chaotic orbit perturbing mechanism," *Opt. Express* **21**(23), 27873–27890 (2013).
17. Z. Hua and Y. Zhou, "Design of image cipher using block-based scrambling and image filtering," *Inf. Sci.* **396**, 97–113 (2017).
18. X. Li et al., "Image encryption based on dynamic filtering and bit cuboid operations," *Complexity* **2019**, 7485621 (2019).

19. T. Li et al., "Image encryption based on pixel-level diffusion with dynamic filtering and DNA-level permutation with 3D Latin cubes," *Entropy* **21**(3), 319 (2019).
20. J. Wu, J. Shi, and T. Li, "A novel image encryption approach based on a hyperchaotic system, pixel-level filtering with variable kernels, and DNA-level diffusion," *Entropy* **22**(1), 5 (2020).
21. G. Hu et al., "An efficient chaotic image cipher with dynamic lookup table driven bit-level permutation strategy," *Nonlinear Dyn.* **87**(2), 1359–1375 (2017).
22. K. U. Shahna and A. Mohamed, "A novel image encryption scheme using both pixel level and bit level permutation with chaotic map," *Appl. Soft Comput.* **90**, 106162 (2020).
23. K. Zhan et al., "Cross-utilizing hyperchaotic and DNA sequences for image encryption," *J. Electron. Imaging* **26**(1), 013021 (2017).
24. L. Liu et al., "A simultaneous scrambling and diffusion color image encryption algorithm based on Hopfield chaotic neural network," *IEEE Access* **7**, 185796–185810 (2019).
25. H. Diab, "An efficient chaotic image cryptosystem based on simultaneous permutation and diffusion operations," *IEEE Access* **6**, 42227–42244 (2018).
26. L. Huang et al., "On symmetric color image encryption system with permutation-diffusion simultaneous operation," *Opt. Lasers Eng.* **115**, 7–20 (2019).
27. L. Liu, Y. Lei, and D. Wang, "A fast chaotic image encryption scheme with simultaneous permutation-diffusion operation," *IEEE Access* **8**, 27361–27374 (2020).
28. S. Vaidyanathan, A. T. Azar, and A. Boulkroune, "A novel 4-D hyperchaotic system with two quadratic nonlinearities and its adaptive synchronisation," *Int. J. Autom. Control* **12**(1), 5–26 (2018).
29. X. Wang and Z. Li, "A color image encryption algorithm based on Hopfield chaotic neural network," *Opt. Lasers Eng.* **115**, 107–118 (2019).
30. C. Pak and L. Huang, "A new color image encryption using combination of the 1D chaotic map," *Signal Process.* **138**, 129–137 (2017).
31. X. Wang, L. Teng, and X. Qin, "A novel colour image encryption algorithm based on chaos," *Signal Process.* **92**(4), 1101–1108 (2012).
32. G. Alvarez and S. Li, "Some basic cryptographic requirements for chaos-based cryptosystems," *Int. J. Bifurcation Chaos* **16**(8), 2129–2151 (2006).
33. J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc.-Int. Conf. Neural Networks*, IEEE, Vol. 4, pp. 1942–1948 (1995).
34. Y. Song et al., "MPPCEDE: multi-population parallel co-evolutionary differential evolution for parameter optimization," *Energy Convers. Manage.* **228**, 113661 (2021).
35. Y. Wu et al., "NPCR and UACI randomness tests for image encryption," *J. Sel. Areas Telecommun.* **1**(2), 31–38 (2011).
36. Y. Zhang and X. Wang, "A new image encryption algorithm based on non-adjacent coupled map lattices," *Appl. Soft Comput.* **26**, 10–20 (2015).
37. L. Liu, Z. Zhang, and R. Chen, "Cryptanalysis and improvement in a plaintext-related image encryption scheme based on hyper chaos," *IEEE Access* **7**, 126450–126463 (2019).

Taiyong Li is a professor at Southwestern University of Finance and Economics. He received his BS, MS, and PhD degrees in computer science from Sichuan University in 2001, 2004, and 2009, respectively. He is the author of more than 40 journal papers. His current research interests include image processing, machine learning, and artificial intelligence.

Jiayi Shi is pursuing his master's degree at Southwestern University of Finance and Economics. His research interests include image processing and machine learning.

Duzhong Zhang is a lecturer at Southwestern University of Finance and Economics. He received his BS, MS, and PhD degrees in communication engineering from Wuhan University of Technology in 2009, 2012, and 2017, respectively. His current research interests include communication and signal processing.