Image Encryption Algorithm Based on Bit-Level Permutation and Dynamic Overlap Diffusion

JIA WANG, JINQING LI, XIAOQIANG DI, JIAN ZHOU, AND ZHENLONG MAN

School of Computer Science and Technology, Changchun University of Science and Technology, Changchun 130022, China
Jilin Province Key Laboratory of Network and Information Security, Changchun University of Science and Technology, Changchun 130022, China
Information Center, Changchun University of Science and Technology, Changchun 130022, China
Corresponding author: Jinqing Li (lijinqing@cust.edu.cn)

This work was supported in part by the Science and Technology Department of Jilin Province through the Natural Science Foundation Project under Grant 20190201188JC, and in part by the Education Department of Jilin Province through the Higher Education Teaching Reform Research Project and the Educational Science Planning Project under Grant JLLG685520190725093004 and Grant 20181219140301.

ABSTRACT In order to guarantee the security of digital images, image encryption is a good tool. This article puts forward a novel image encryption/decryption algorithm based on bit-level permutation and dynamic overlap diffusion. Firstly, location information set by users is used to scramble the plain image in the bit-level, changing the pixel position of the image while changing the pixel value. Then, using the plaintext related method to control the iteration parameters of Arnold transform. These two parts together form the permutation process of the algorithm. Thirdly, the overlap steps are set according to the encryption security requirements, and dynamic overlap diffusion is performed on the permutation image. At the same time, the plaintext information entropy and variance are used to update the diffusion key in the five-dimensional hyperchaotic system to achieve the plaintext correlation of the encryption key. Arnold transform and dynamic overlap diffusion can ensure that different key streams are generated when encrypting different images. Finally, experimental analysis and numerical simulation show that our algorithm has high efficiency, high security, high sensitivity, and can effectively resist various attacks.

INDEX TERMS Bit-level permutation, dynamic overlap diffusion, hyperchaotic system, image encryption, plaintext related key.

I. INTRODUCTION

Text information and multimedia information are the main ways of current information transmission. As an important part of multimedia information transmission, images can better reflect the static characteristics of real objects. As important data information for multimedia transmission, the security of digital images cannot be ignored. The biggest difference between images and text is the correlation between image pixels and high redundancy. Therefore, the field of image encryption has become a hot field of current research [1]–[4]. In recent years, a lot of image encryption algorithms have appeared.

An image encryption algorithm is usually processes pixel-level data [5]–[7], block-level data [8]–[10], DNA-level data [11]–[13] or bit-level data [14]–[16]. Encryption algorithms associated with low-level data usually show better performance in image encryption [16]. Therefore, more and more researchers have shifted their research focus to bit-level image encryption.

In 2018, Guodong Ye et al. [5] proposed a new and efficient pixel-level image encryption algorithm, this algorithm enhances the connection between pixel position reorganization and gray value change, and a new type of permutation–rewriting–diffusion structure is proposed to enhance the system’s resistance to attack. Fu et al. [15] used the chaotic system to perform bit-level permutation in the literature, which overcomes the shortcoming of pixel permutation only by disturbing the pixel position. This method can simultaneously perform pixel permutation and diffusion. In 2019, Wang et al. [17] proposed an image encryption algorithm based on bit-combination scrambling and dynamic
diffusion is proposed based on multiple chaotic systems. In their method, they proposed a diffusion method called 

dynamic diffusion, it has \( N \) modes of diffusion, where \( N \) is the number of pixel values. However, in the above encryption 

method, the scrambling process and the diffusion process are independent.

Traditional encryption algorithms such as AES, DES, 

IDEA [18]–[20] take a long time to calculate and have high 

computational strength, so it is not suitable for image encryption [21]–[23]. The word “Chaos” was first proposed by 

Tianyan Li and J.A. Yorke in 1975 [24]. They gave a math- 

ematical definition of chaos in the “Period Three Implies 

Chaos”. The basic characteristics of the chaotic system and 

the natural relationship of cryptography are mentioned in 

Shannon’s 1949 classic “Communication Theory of Secrecy 

Systems” [25]. Chaotic systems are widely used in the field 

of image encryption due to its inherent complexity, sensi-

tivity to initial values, uncertainty and so on. Since Fridrich 

first applied a chaotic encryption method to image encryp-

tion in 1997 [26], chaotic image encryption technology has 

become a hotspot in digital image encryption technology. In 2006, Indian scholars Pareek et al. [4] designed an image 

encryption method based on the chaotic logic mapping. In the 

proposed encryption process, eight different types of opera-

tions were utilized to encrypt the pixels of the image, this way 

can increase the complexity of the encryption process.

Chaotic systems can be divided into two types: low-dimen-

sional chaotic systems and high-dimensional chaotic systems. Low-dimensional systems have a simple 

structure and fewer parameters, so it is difficult to guarantee 

the security of images [4], [22], [27]–[29]. High-dimensional 

chaotic systems have high complexity and are more prone 

to hyperchaotic states, so they have been widely used in 

the field of image encryption [13], [30]–[34]. In the litera-

ture [31], a hyperchaotic system of five-dimensional hyper-

chaotic is proposed, which has complex dynamic behavior, 

so a large encryption scheme with large keyspace can be 

generated using its great number of parameters. Qi Yin and 

coworkers [35] propose a dynamic diffusion method in 

order to enhance the security of the cipher system, which 

can ensure that encrypting each pixel is related to all other 

pixels and can enhance the sensitivity of the cryptosys-

tem. In [36], a novel bit-level image encryption algorithm is 

proposed based on piecewise linear chaotic map. The main innovation is the cyclic shift operation of the diffusion 

stage, which can effectively spread the pixels, this process 

requires repeated operations, so the image encryption is not 

efficient.

Motivated by the above, we propose an image encryption 

algorithm which based on bit-level permutation, Arnold 

transform and dynamic overlap diffusion. Our contributions 

are summarized as follows:

• A new bit-level scrambling scheme is designed, this 

method can make permutation and diffusion processes 

to proceed at the same time, which greatly improving 

encryption efficiency.

• A dynamic overlap diffusion method is proposed, which 

uses the hyperchaotic matrix to realize the dynamic 

overlap diffusion function of pixels.

• To ensure the relevance of the key with plaintext, a set of 

key update schemes are given, including a diffusion key 

stream update scheme for plaintext information entropy 

and variance and a plaintext update scheme for Arnold 

transform parameters.

The rest of this article is organized as follows. In Section 2, the basic theories of Arnold transform and 

the five-dimensional hyperchaotic system are described. In Section 3, the processes of image encryption and 

decryption are described in detail. The simulation results and per-

formance analysis are shown in Section 4. Finally, this article 

is concluded in the last section.

II. PRELIMINARY WORK

A. ARNOLD TRANSFORM

In this article, Arnold transform is employed to change the 

pixel positions in the plain image, the original position of the 

pixel is denoted as \((\alpha, \beta)\), the position of the permutation 

image is denoted as \((\alpha', \beta')\), and the definition of general 

Arnold transform is given in (1) [37]:

\[
\begin{bmatrix}
\alpha' \\
\beta'
\end{bmatrix} = 
\begin{bmatrix}
1 & b \\
 a & ab + 1 
\end{bmatrix}
\begin{bmatrix}
\alpha \\
\beta
\end{bmatrix} \mod(N) \tag{1}
\]

where \(a, b\) are parameters, \(N\) is the height or width of the 

image, and \(t\) is the number of iterations. Equation (1) is 

a classical two-dimensional Arnold map when \(N = 256\) 

and \(a = b = 1\). Perform Arnold transform system (1) 

times to ensure that the permutation process is sufficient. 

The inverse Arnold transform is derived according to (1), 

that is, the corresponding permutation inverse process for 

decryption is shown in (2).

\[
\begin{bmatrix}
\alpha \\
\beta
\end{bmatrix} = 
\begin{bmatrix}
ab + 1 & -b \\
-a & 1
\end{bmatrix}
\begin{bmatrix}
\alpha' \\
\beta'
\end{bmatrix} \mod(N) \tag{2}
\]

B. FIVE-DIMENSIONAL HYPERCHAOTIC SYSTEM

The equation of state of the five-dimensional hyperchaotic 

system is as follows [31]:

\[
\begin{align*}
\dot{x} &= a_1 (y - x) + u \\
\dot{y} &= c_1 - xz + w \\
\dot{z} &= -b_1z + xy \\
\dot{u} &= -h_1u - xz \\
\dot{w} &= -k_1 - k_2y 
\end{align*} \tag{3}
\]

where \(a_1b_1h_1 \neq 0, a_1, b_1 \) and \(c_1 \) are the constant parameters, 

\(h_1, k_1 \) and \(k_2 \) are three control parameters, determining the 

chaotic and hyperchaotic behaviors and bifurcations of the 

system. Numerical simulations have verified that system (3) 

has a hyperchaotic attractor when \((a_1, b_1, c_1, h_1, k_1, k_2) = 

(10, 8/3, 28, -2, 0, 56/5)\), as depicted in Fig.1. Fig.2 shows the 

bifurcation diagram of (3). The spectrum of Lyapunov
exponents illustrated in Fig.3 shows that the five-dimensional chaotic system (3) has two or three positive Lyapunov exponents when $k_2 \in (0, 21)$ and $k_2 \in (29, 61)$, which is a hyperchaotic system [31].

III. ENCRYPTION/DECRYPTION SCHEME

This section is divided into two parts: the encryption process and the decryption process. Due to the symmetry of the algorithm, the decryption process is the reverse of the encryption process, the image encryption algorithm will be described in detail. The encryption process is divided into two aspects. One aspect is based on the scrambling process of bit-level matrix operation and Arnold transform. The other is the dynamic overlap diffusion method which based on chaotic map.

FIGURE 1. Chaotic attractor of hyperchaotic system (3).

FIGURE 2. Bifurcation diagram of hyperchaotic system (3):
(a) $(a_1, b_1, c_1, h_1, k_1, k_2) = (a_1, 8/3, 28, -2, 0, 56/5), a_1 \in (4, 14),$
(b) $(a_1, b_1, c_1, h_1, k_1, k_2) = (10, 8/3, 28, -2, 0, 56/5), b_1 \in (0, 6),$
(c) $(a_1, b_1, c_1, h_1, k_1, k_2) = (10, 8/3, c_1, -2, 0, 56/5), c_1 \in (20, 40),$
(d) $(a_1, b_1, c_1, h_1, k_1, k_2) = (10, 8/3, 28, h_1, 0, 56/5), h_1 \in (-2, 0),$
(e) $(a_1, b_1, c_1, h_1, k_1, k_2) = (10, 8/3, 28, -2, k_1, 56/5), k_1 \in (-5, -1),$
(f) $(a_1, b_1, c_1, h_1, k_1, k_2) = (10, 8/3, 28, -2, 0, k_2), k_2 \in (10, 70).$

FIGURE 3. Lyapunov exponents of hyperchaotic system (3) with $(a_1, b_1, c_1, h_1, k_1, k_2) = (10, 8/3, 28, -2, 0, k_2)$ and $k_2 \in (0.70).$

A. ENCRYPTION ALGORITHM

Take the 8-bit grayscale image as the plain image. The encryption process is shown in Fig.4. In terms of color image encryption, separate the color components of the image, and encrypt each color component separately to obtain the same encryption effect as the grayscale image.

1) PERMUTATION PROCESS

Step 1. Decompose the plain image to bit-level image matrices. The gray value of each pixel in the plain image is represented by an 8-bit binary number. Extract the last bit of each binary number to form a bit-level matrix $blm(1)$. The seventh bit of each binary number is extracted to form a bit-level matrix $blm(2)$. We can derive the rest from each binary number until the first bit of each binary number is used to form a bit-level matrix $blm(8)$:

$$blm(i) = \text{bitget}(P, 9 - i)$$

where $\text{bitget}()$ is the extracted bit value function, and $P$ is the plain image, $i = 1, 2, \ldots, 8$.

Step 2. The bit-level matrix $blm(i)$ performs matrix transformation in order from top to bottom and from left to right, and then obtains a one-dimensional bit sequence $Sblm(i)$.

Step 3. $Sblm(i)$ is reassembled in the order specified by the user, and this order is part of the user’s encryption key.

$$\text{RasM} = [Sblm(ras\_order(i_1)), Sblm(ras\_order(i_2)), \ldots, Sblm(ras\_order(i_8))]$$

where $\text{RasM}$ denotes the reassembled matrix, the size is $(M \times N) \times 8$. $\text{ras\_order}$ represents the reassembly order key set by the user, $i_1 \neq i_2 \neq \ldots, \neq i_8$, $i_1, i_2, \ldots, i_8 \in \{1, 2, 3, 4, 5, 6, 7, 8\}$.

Step 4. Combine every 8 elements in $\text{RasM}$ to form a new 8-bit binary sequence and convert it to decimal. Reshape the decimal sequence into a matrix to an $M \times N$ matrix.

Step 5. Arnold transform is performed on this decimal matrix, and the matrix is scrambled to obtain a pixel-level permutation image $PBimg$. The specific Arnold transform is described in Section 2. The formula for the number of
iterations $t$ of the Arnold transform is defined by (6):

$$t = \text{rem} \left( \sum_{i=1}^{M} \sum_{j=1}^{N} P(i,j), d \right) + 1$$

(6)

where $M$ and $N$ represent the length and width of the image, respectively, $P$ represents the plain image and $\text{rem}(\cdot)$ is the remainder function. Taking into account the following three factors, i.e., the periodicity of the Arnold transform, whether the transformation is sufficient, and the encryption/decryption time, we set $10 \leq d \leq 50$, in which $d$ is a positive integer set by the user to control the number of iterations. If $d$ exceeds the upper limit of the range, excessive encryption/decryption time will be consumed. By controlling the range of parameter $d$, the security performance and encryption time can be balanced. Under the premise of ensuring the encryption security of the application level, the encryption time is not too long. The acquisition process of the bit-level permutation image is shown in Fig. 5.

2) DIFFUSION PROCESS

Step 6. The pixel-level permutation image $PBimg$ of size $M \times N$ is reshaped into a matrix $FM$ of size $m \times (\frac{M \times N}{m})$ according to the method shown in (7):

$$FM = \begin{bmatrix}
PBimg(1) & \cdots & PBimg\left(\frac{M \times N}{m}\right) \\
\vdots & \ddots & \vdots \\
PBimg(m) & \cdots & PBimg(M \times N)
\end{bmatrix}$$

(7)

where $m$ is a partition block size, which is a positive integer divisible by $M \times N$.

Step 7. The matrix $FM$ is divided into $F$ permutation blocks $ZB_j (j = 0, 1, \ldots, F - 1)$ which size is $m \times m$. The segmentation method is as shown in (8):

$$ZB_j = \begin{bmatrix}
FM(1, 1 + ST \times j) & FM(1, 2 + ST \times j) \\
FM(2, 1 + ST \times j) & FM(2, 2 + ST \times j) \\
\vdots & \vdots \\
FM(m, 1 + ST \times j) & FM(m, 2 + ST \times j) \\
\cdots & \cdots \\
FM(1, m + ST \times j) & FM(2, m + ST \times j) \\
\cdots & \cdots \\
FM(m, m + ST \times j)
\end{bmatrix}$$

(8)

$$F = \begin{cases} 
\text{floor} \left( \frac{M \times N}{m \times ST} \right), & \text{kend} \geq m \\
\text{floor} \left( \frac{M \times N}{m \times ST} \right) - 1, & \text{kend} < m
\end{cases}$$

(9)
where $kend = \text{rem} \left( \frac{M \times N}{m}, ST \right) + ST$, $ST$ represents the segment step size of the matrix and $m/2 \leq ST \leq m$, floor() is a round down function.

Step 8. Iterate the five-dimensional hyperchaotic system $(3) Tc$ times to obtain the matrix $HCM$. $Tc$ is the number of iterations of the five-dimensional hyperchaotic system.

$$HCM = \begin{bmatrix} x_1(1) & \cdots & x_5(1) \\ \vdots & \ddots & \vdots \\ x_1(Tc) & \cdots & x_5(Tc) \end{bmatrix} \quad (10)$$

Map the element values of the matrix $HCM$ to a positive integer between 0 and 255, as shown in $(11)$.

$$HCM_{255} = \text{round}((HCM_{255_{\max}} - HCM_{255_{\min}}) \times \frac{HCM - xmin}{xmax - xmin}) + HCM_{255_{\min}} \quad (11)$$

where $HCM_{255_{\max}} = 255$, $HCM_{255_{\min}} = 0$, $xmax = \max(HCM)$, $xmin = \min(HCM)$, round() is a rounding function, $\max()$ is the maximum function, and $\min()$ is the minimum function.

Step 9. Discard the first $t_r$ rows in $HCM_{255}$, starting from the $(t_r + 1) -$th row, obtain $M \times N$ elements from $HCM_{255}$, and then rearrange these elements into a matrix $FCM_r$ of size $m \times \left( \frac{M \times N}{m} \right)$, $r$ is a counter with an initial value of 1. The diffusion process will be executed cyclically $n$ times, each time a different diffusion key will be taken from the $HCM_{255}$ to enhance the security of the encryption algorithm. $FCM_r$ represents the $r$-th diffusion key obtained from $HCM_{255}$.

Step 10. The chaotic matrix $FCM_r$ is divided into $F$ chaotic blocks $BFCM_j$ of size $m \times m$. The segmentation method is shown as $(12)$:

$$BFCM_j = \begin{bmatrix} FCM_r(1, p(r) + \text{st} \times j) & FCM_r(1, p(r) + \text{st} \times j + 1) \\ \vdots & \vdots \\ FCM_r(m, p(r) + \text{st} \times j) & FCM_r(m, p(r) + \text{st} \times j + 1) \\ \cdots & \cdots & \cdots & \cdots \\ FCM_r(1, p(r) + \text{st} \times j + m - 1) & FCM_r(1, p(r) + \text{st} \times j + m) \\ \vdots & \vdots & \vdots & \vdots \\ FCM_r(m, p(r) + \text{st} \times j + m - 1) & FCM_r(m, p(r) + \text{st} \times j + m + 1) \end{bmatrix} \quad (12)$$
where \( st \) is the segment step size of the chaotic matrix, \( st \leq ST \), \( p(r) \) is the starting position of the matrix \( FCM_r \) intercepted from the matrix \( HCM255 \). The calculation method of \( p(r) \) is as follows:

\[
p(r) = \text{rem(floor}(\frac{v(img) \times 10^4}{v(img) + r \times h(img) + 1}), \frac{M \times N}{m})
\]

(13)

where \( h(img) \) is the information entropy of image \( img \), \( v(img) \) is the variance of the histogram of the image \( img \). Since \( p(r) \) represents the starting position of the matrix, when \( p(r) = 0 \), its value should be reassigned to 1. The information entropy and variance calculation formula are shown as (14) and (15) [38]:

\[
H(s) = \sum_{i=0}^{2^l-1} p(s_i) \log_2 \frac{1}{p(s_i)}
\]

(14)

\[
v(img) = \frac{-1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} (img_i - img_j)^2
\]

(15)

where \( L \) is the total number of pixels in the cipher image, \( s_i \) indicates the i-th pixel value, \( p(s_i) \) represents the probability of \( s_i \), \( img \) is the vector of the histogram values and \( img = \{img_1, img_2, \ldots, img_{256}\} \), \( img_i \) and \( img_j \) represent the gray values of the i-th and j-th pixels in the image \( img \), respectively.

Here \( img \) indicates the plain image. Since the information entropy and variance of different images are different, we introduce them in the diffusion key, making the key related to the plaintext. In this way, even if the user uses the same encryption keys, the diffusion keys generated when encrypting different images are not the same, and the security of “one key one time” is truly achieved.

When the matrix block \( BFCM_j \) reaches the end of the matrix \( FCM_r \) and the number of \( FCM_r \) columns is insufficient, continue traversing from the first column of \( FCM_r \), record this column number as \((M \times N)/(m+1)\), and the number of subsequent columns increases sequentially.

Step 11. The permutation blocks \( ZB_j \) and the chaotic block \( BFCM_j \) are diffused to obtain the diffusion image block \( KB_j \) as shown in (16):

\[
KB_j = ZB_j \oplus BFCM_j
\]

(16)

where \( \oplus \) means bitwise XOR operation.

Step 12. The diffusion image block \( KB_j \) is arranged in a matrix segmentation step size \( ST \) to get a diffusion matrix \( KC \):

\[
\begin{bmatrix}
KB_0(1, 1) & KB_0(1, 2) & \cdots & KB_0(1, ST) & KB_1(1, 1) \\
KB_0(2, 1) & KB_0(2, 2) & \cdots & KB_0(2, ST) & KB_1(2, 1) \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
KB_0(m, 1) & KB_0(m, 2) & \cdots & KB_0(m, ST) & KB_1(m, 1)
\end{bmatrix}
\]

The diffusion image blocks \( KB_j \) are overlapped in the matrix \( KC \) with a step size of \( ST \), and the part covered in the previous block by the next image block is automatically discarded. When \( \frac{M \times N}{m} \) is not equal to \( m + ST \times (F - 1) \), the number of columns \( res \) at the end of \( FM \) is not enough to form a matrix block; add the remaining matrix directly at the end of the \( KC \) matrix. Finally, a diffusion matrix \( KM \) of size \( m \times (\frac{M \times N}{m}) \) is obtained.

\[
res = \frac{M \times N}{m} - (m + ST \times (F - 1))
\]

(18)

Step 13. The \( KC \) is rearranged in order from top to bottom, left to right, and we get the \( FNimg \) of size \( M \times N \). Determine whether \( r \) and \( n \) are equal. If not, replace \( PBimg \) with \( FNimg \), add 1 to \( r \) and return to Step 6. If \( r \) is equal to \( n \), the \( FNimg \) is the final cipher image of size \( M \times N \). \( n \) is the number of diffusion cycles set by the user. The detailed process of diffusion is shown in Fig.6.

**B. DECRYPTION PROCESS**

The algorithm proposed in this article is a symmetric encryption algorithm, so the decryption process is the inverse of the encryption process. The specific decryption process is shown in Fig.7.

**IV. EXPERIMENTAL SIMULATION AND NUMERICAL ANALYSIS**

**A. RESULTS**

In order to facilitate analysis and comparison of the security performance of the proposed algorithm, we used three grayscale images “Baboon”, “Peppers”, “Lena”, and three color images “Cornfield”, “Flower”, “Lena” with the size of 256 × 256 as test images. The constant parameter values of the scrambling process are \( i_1 = 7, i_2 = 5, i_3 = 6, i_4 = 8 \), \( i_5 = 3, i_6 = 1, i_7 = 2, i_8 = 4, d = 15 \). The constant parameters of the diffusion process are \( c_1 = 28, m = 16, n = 1 \). The 10 sets of variable keys used in the experiments are shown in Table 1. The following simulation experiments run on Windows 10 operating system, Intel Core 1.7G CPU, 8GB RAM, 480G hard disk and the experiment is performed using MATLAB 2015. The encryption and decryption results of the two grayscale images “Baboon”, “Peppers”, and the two color images “Cornfield”, “Flower” are shown in Fig.8.

**B. ANALYSIS OF ENCRYPTION/DECRYPTION SPEED**

We use 10 different encryption keys to encrypt 8 different grayscale images, and calculate that the average encryption time of the encryption algorithm for the grayscale image is 0.178s. The average encryption time of a color image
is three times of a grayscale image, which is 0.535s. Table 2 lists the encryption/decryption time and encryption/decryption speed of our algorithm. The encryption speed = the size of the image × 8bits/encryption time, and the decryption speed = the size of image × 8bits/decryption time.

It can be calculated from the table that the average encryption speed for the grayscale image is 2.95Mbps and
From the data in Table 2, it can be seen that the encryption algorithm proposed in this article is faster than literature [7], [14], and [39].

### C. KEY SPACE ANALYSIS

The secret key of our proposed encryption algorithm is composed of Arnold transform parameters \(a\), \(b\), \(d\), the initial keys, and the control parameters \(x_0\), \(y_0\), \(z_0\), \(u_0\), \(w_0\), \(h_1\), \(k_1\), \(k_2\) of five-dimensional hyperchaotic system, segment step size \(ST\) and \(st\), matrix block size \(m\), and the number of diffusion times \(n\). If the precision of each initial key is \(10^{-16}\), the key space of our algorithm is \((10^{16})^{15} = 10^{240} \gg 2^{100}\). According to the average encryption speed, an exhaustive attack takes about \(1.03 \times 10^{232}\) years to traverse the entire key space. Similarly, an exhaustive attack takes about \(1.03 \times 10^{232}\) years to try all the key space according to the average decryption speed. So the cryptographic system can resist brute-force attacks effectively [38], [40].

### D. STATISTICAL ANALYSIS

The cipher image requires certain randomness between the pixels of the image. The important indicator for statistical analysis is the histogram analysis and the correlation between adjacent pixels.

#### 1) HISTOGRAM ANALYSIS

The frequency at which each gray level appears can be clearly seen in the histogram. The more irregular the pixel distribution, the more uniform the corresponding histogram distribution should be. The histogram of the plain image and the histogram of the cipher image of the two grayscale images and one color image tested in this article are shown in Fig.9 and Fig.10.

To further analyze the histogram, the variance of the encrypted image histogram is used to measure the uniformity of image encryption. When the keys are different, the closer the two variance values are, the higher uniformity of the encrypted image. The variance of the histogram is shown in (15).
TABLE 3. Values of $v$.

| Test image     | Plain image  | $key_1$ | $key_2$ | $key_3$ | $key_4$ | $key_5$ |
|----------------|--------------|---------|---------|---------|---------|---------|
| Gray Baboon    | 53576.0938   | 473.1638| 522.2226| 508.6052| 516.5291| 512.1006|
| Gray Peppers   | 51263.0937   | 501.3025| 551.4553| 545.0843| 541.7548| 494.5994|
| Cornfield-red  | 66889.1258   | 529.4814| 518.1511| 513.3193| 560.6732| 527.3789|
| Cornfield-blue | 59132.3091   | 531.1719| 475.4125| 474.5664| 451.6853| 493.7911|
| Flower-red     | 19630.3569   | 503.2400| 504.5603| 528.4177| 506.8359| 507.5381|
| Flower-green   | 42093.6953   | 525.4414| 531.0078| 518.4219| 490.5110| 493.8748|
| Flower-blue    | 76302.5466   | 539.0219| 590.4060| 511.4067| 212.6245| 573.8110|

FIGURE 9. Histogram analysis: (a) (d) are histograms of plain images "Baboon" and "Peppers", (b) (e) are histograms of encrypted images "Baboon" and "Peppers", (c) (f) are histograms of decrypted images "Baboon" and "Peppers".

FIGURE 10. Histogram analysis: (a) Histogram of red component of plain image "Flower" (b) Histogram of red component of encrypted image "Flower" (c) Histogram of red component of decrypted image "Flower" (d) Histogram of green component of plain image "Flower" (e) Histogram of green component of encrypted image "Flower" (f) Histogram of green component of decrypted image "Flower" (g) Histogram of blue component of plain image "Flower" (h) Histogram of blue component of encrypted image "Flower" (i) Histogram of blue component of decrypted image "Flower".

To verify the uniformity of the encrypted image, we use the five keys $key_1$, $key_2$, $key_3$, $key_4$, and $key_5$ in Table 1 to encrypt four test images respectively. The analysis of the histogram variance is performed on the encrypted images, and the analysis results are listed in Table 3.

The variance stability rate is used to measure the stability of the histogram variance of different images encrypted by different keys: $\zeta(z) = \frac{| v - \text{AVG}_v |}{\text{AVG}_v} \times 100\%$ (19)

where $v$ is the variance of the image histogram, $\text{AVG}_v$ is the average of the histogram variance of the corresponding image encrypted with different keys. List the value of $\zeta(z)$ in Table 4.

Experimental results show that the value of $\zeta(z)$ will stabilize around 3.3%, and the histogram level of encrypted images will remain at a relatively stable level, so the algorithm can effectively resist statistical attacks [41].

TABLE 4. Values of $\zeta(z)$.

| Test image     | $key_1$(%) | $key_2$(%) | $key_3$(%) | $key_4$(%) | $key_5$(%) |
|----------------|------------|------------|------------|------------|------------|
| Gray Baboon    | 6.5        | 3.0        | 0.4        | 1.9        | 1.9        |
| Gray Peppers   | 4.8        | 4.6        | 3.4        | 2.8        | 6.1        |
| Cornfield-red  | 0.0        | 2.1        | 3.1        | 5.8        | 0.4        |
| Cornfield-blue | 5.5        | 5.5        | 5.7        | 7.6        | 1.8        |
| Flower-red     | 1.3        | 1.0        | 3.5        | 0.6        | 0.5        |
| Flower-green   | 2.6        | 3.7        | 1.2        | 4.1        | 3.5        |
| Flower-blue    | 2.1        | 0.8        | 3.6        | 7.7        | 1.1        |
| Flower-blue    | 1.5        | 7.8        | 6.5        | 4.6        | 4.8        |

2) CORRELATION ANALYSIS

There is a certain correlation between adjacent pixels in an image. An attacker can attack the encrypted image by analyzing the correlation of the horizontal, vertical, and diagonal adjacent pixels of the image. Therefore, it is necessary to reduce the pixel correlation in the encrypted image to ensure the security of the cipher image.

An excellent encryption algorithm should make the correlation of adjacent pixels of the encrypted image as low as possible. The correlation coefficient formula of adjacent pixels is calculated by (20).

$$r_{xy} = \frac{\text{cov}(x, y)}{\sqrt{D(x) \times D(y)}}$$

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

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

$$E(x) = \frac{1}{N} \sum_{i=1}^{N} x_i$$ (20)
where $r_{xy}$ represents the correlation coefficient between $x$ and $y$, $\text{cov}(x, y)$ is the covariance of $x$ and $y$, $D(x)$ is the variance of $x$, $D(y)$ is the variance of $y$, $E(x)$ is the average value of $x$, $E(y)$ is the average value of $y$, $x$ and $y$ are grayscale levels of two neighboring pixels in an image, and $N$ denotes the total number of pixels in an image.

In this test, we randomly select 5000 pairs of adjacent pixels and test the correlation between the plain images and their corresponding encrypted images in the horizontal, vertical, and diagonal directions. Fig. 11(a)-(f) and Fig. 12(a)-(i) illustrate the correlation analysis of plain images and encrypted images of “Baboon”, “Peppers”, and “Flower”, respectively. In order to compare with other references, we also encrypted the grayscale “Lena” image, and the correlation coefficient results are listed in Table 5. Obviously, our encryption algorithm reduces the pixel correlation of the encrypted image, and the correlation coefficient of adjacent pixels approaches zero. The attacker cannot obtain any useful information from the correlation of the adjacent pixels of the encrypted image.

### 3) INFORMATION ENTROPY

Image entropy is a statistical form of features that reflects the average amount of information in an image. The information entropy of an image represents the amount of information contained in the aggregated features of the gray distribution in the image. The gray level entropy of the gray image is defined as (14).

According to the calculation method shown in (14), the ideal information entropy of an 8-bit grayscale image is 8. The closer the value of the information entropy is to the ideal value, the more the encrypted image looks like white noise, and the higher the degree of confusion. The information entropy of the encrypted images of the grayscale images “Baboon”, “Peppers”, “Lena”, and the color images “Cornfield”, “Flower”, “Lena” are given in Table 6, compared with other similar algorithms, it can be seen that the entropy of the encrypted image of our encryption algorithm is very close to the ideal value, which can effectively resist the entropy attack.

In this section, we not only calculated the information entropy of the encrypted image, but we also analyzed and calculated the relative entropy, redundancy, and local entropy of the encrypted and decrypted image.

Relative Entropy, also known as Kullback-Leibler divergence, is a measure of the asymmetry of the difference between two probability distributions [44].

$$KL(P||Q) = \sum_{i=0}^{255} P(i) \log_2 \frac{P(i)}{Q(i)} \quad (21)$$
TABLE 6. Information entropy.

| Encrypted image      | Information entropy |
|----------------------|---------------------|
| Gray Baboon          | 7.9976              |
| Gray Peppers         | 7.9973              |
| Gray Lena            | 7.9972              |
| Cornfield-red        | 7.9970              |
| Cornfield-green      | 7.9973              |
| Cornfield-blue       | 7.9969              |
| Flower-red           | 7.9970              |
| Flower-green         | 7.9969              |
| Flower-blue          | 7.9974              |
| Lena-red             | 7.9970              |
| Lena-green           | 7.9970              |
| Lena-blue            | 7.9970              |
| Ref. [7]             | 7.9970              |
| Ref. [16]            | 7.9972              |
| Ref. [42]-red        | 7.986856            |
| Ref. [42]-green      | 7.987956            |
| Ref. [42]-blue       | 7.988398            |
| Ref. [43]-red        | 7.9899              |
| Ref. [43]-green      | 7.9894              |
| Ref. [43]-blue       | 7.9896              |

FIGURE 13. Key sensitivity test of “Peppers”: (a) Plain image of “Peppers”, (b) (c) (d) (e) (f) represent encrypted images C1, C2, C3, C4, C5 respectively.

For an 8-bit grayscale image, its information entropy divided by 8 is called its relative entropy. If the probability distribution of the encrypted image and the ideal white noise image is closer, the relative entropy is closer to 1.

redu(P||Q) = (1 − KL(P||Q)) × 100% (22)

The difference between 1 and relative entropy is called redundancy, and the smaller the redundancy value, the lower the information redundancy of the image. At the same time, we also calculated the values of the local entropy of 16 × 16 and 32 × 32 image blocks, respectively.

The information entropy, the relative entropy and the redundancy of four images “Baboon”, “Peppers”, “Cornfield”, and “Flower” are shown in Table 7 and Table 8. It is noticeable that the cipher image encrypted by the proposed algorithm has good statistical characteristics.

4) PEAK SIGNAL-TO-NOISE RATIO

In this section, to provide more detailed verification information, we calculated the peak signal-to-noise ratio (PSNR) to measure the similarity between plain images and encrypted images. The PSNR value of two identical images is infinite. If the value of PSNR is low depicts that there is a significant difference between the plain and encrypted images. We record the plain image as P and the encrypted image as C. The mean square error (MSE) between them is defined as for formula (23) [7]:

\[ MSE = \frac{1}{M \times N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |P(i, j) - C(i, j)|^2 \] (23)

PSNR is defined as follows [32]:

\[ PSNR = 10 \times \log_{10} \left( \frac{MAX^2_i}{MSE} \right) \] (24)

where \( MAX_i \) represents the maximum value of the pixel. The PSNR results of four tested images are listed in Table 9.
Structural similarity (SSIM) is an index that measures the similarity of two images. Given two images $x$ and $y$, the structural similarity of the two images can be obtained as formula (25):

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)}$$

(25)

where $\mu_x$ is the mean of $x$, $\mu_y$ is the mean of $y$, $\sigma_x^2$ is the variance of $x$, $\sigma_y^2$ is the variance of $y$, $\sigma_{xy}$ is the covariance of $x$ and $y$, $c_1 = (k_1L)^2$ and $c_2 = (k_2L)^2$ are constants used to maintain stability, $k_1 = 0.01$, and $k_2 = 0.03$ are the default values, $L$ is the range of pixel values. The value range of SSIM is [0,1], the larger the value, the smaller the image distortion. When the two images are exactly the same, the SSIM value is equal to 1. Record the value of SSIM between the plain images and the encrypted images as SSIM (a), the value of SSIM between the plain images and the decrypted images as SSIM (b), and the calculation results are listed in Table 10.

It can be seen from Table 10 that the algorithm proposed in this article is lossless encryption.
TABLE 9. MSE and PSNR results of test images.

| Test image  | MSE   | PSNR  |
|-------------|-------|-------|
| Gray Baboon | 7088.6528 | 9.6308 |
| Gray Peppers | 8678.0549 | 8.7022 |
| Cornfield-red | 7772.5447 | 9.2252 |
| Cornfield-green | 7827.7898 | 9.1977 |
| Cornfield-blue  | 11250.1259 | 7.6192 |
| Flower-red      | 9246.6632 | 8.4710 |
| Flower-green    | 9542.5801 | 8.3341 |
| Flower-blue     | 12134.1900 | 7.2907 |

TABLE 10. SSIM results of test images.

| Test image  | SSIM(a) | SSIM(b) |
|-------------|---------|---------|
| Gray Baboon | 0.0102 | 1.0000 |
| Gray Peppers| 0.0086 | 1.0000 |
| Cornfield-red | 0.0108 | 1.0000 |
| Cornfield-green | 0.0112 | 1.0000 |
| Cornfield-blue   | 0.0093 | 1.0000 |
| Flower-red      | 0.0093 | 1.0000 |
| Flower-green    | 0.0099 | 1.0000 |
| Flower-blue     | 0.0087 | 1.0000 |

E. DIFFERENTIAL ATTACK

Differential attack attacks the encryption algorithm by comparing and analyzing the propagation of different plaintext after encryption. An excellent image encryption algorithm should be sensitive to the plain image, that is, changing one pixel or one bit in the plain image can get two completely different encrypted images. Three important indicators of differential attack analysis are the number of pixels change rate (NPCR), the unified average changing intensity (UACI), and block average changing intensity (BACI). NPCR refers to the rate of change between two encrypted images after a small change in the plain image. That is the ratio of grayscale values of different encrypted images at the same position.

\[
NPCR = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} |\text{Sign}(P_1(i, j) - P_2(i, j))|}{M \times N} \times 100\% \tag{26}
\]

where \(P_1(i, j)\) and \(P_2(i, j)\) are the ciphered images before and after one pixel of the plain image is changed, \(\text{Sign}(\cdot)\) is a symbolic function, as shown in (27):

\[
\text{Sign}(x) = \begin{cases} 
1, & x > 0 \\
0, & x = 0 \\
-1, & x < 0 
\end{cases} \tag{27}
\]

UACI indicates the degree to which the two encrypted images correspond to pixel differences.

\[
\text{UACI} = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{|P_1(i, j) - P_2(i, j)|}{255} \times 100\% \tag{28}
\]

BACI is the average intensity of change in block-wise. In theory, BACI is more illustrative than UACI. The definition of BACI is as follows [45]:

\[
\text{BACI} = \frac{1}{(M-1)(N-1)} \sum_{i=1}^{m_i} \frac{m_i}{255} \tag{29}
\]

where \(m_i = (|d_{11} - d_{21}| + |d_{11} - d_{31}| + |d_{11} - d_{41}| + |d_{21} - d_{31}| + |d_{21} - d_{41}| + |d_{31} - d_{41}|)/6.\) Let \(D = |P_1 - P_2|\), divide \(D\) into image blocks of size \(2 \times 2\), the \(i\)-th block is
shown in (30):

$$D_i = \begin{bmatrix} d_{1i} & d_{2i} \\ d_{3i} & d_{4i} \end{bmatrix}$$  \hspace{1cm} (30)$$

where \( i = 1, 2, \ldots, (M-1)(N-1).$$

The ideal values for NPCR, UACI, and BACI are 99.6094\%, 33.4635\%, 26.7712\%, respectively [40], [46]. The NPCR, UACI, and BACI results of four test images used in this article are listed in Table 11. The closer the calculation result is to the ideal value, the more sensitive the encryption algorithm is to the slight changes in plaintext. Otherwise, the encryption algorithm is insensitive to the change in plaintext. The NPCR and UACI results of this algorithm are compared with the grayscale images in literature [1] and [7] and the color images in literature [42] and [47]. It can be seen from Table 11 that the proposed algorithm can effectively resist the differential attack.

### F. KEY SENSITIVITY ANALYSIS

A good encryption and decryption algorithm should be highly sensitive to the key. Given key = \( \{ x_0, y_0, z_0, u_0, w_0, h_1, k_1, k_2, a, b, d, ST, st \} \). For the value of key, see the key\(_i\) column in Table 1.
TABLE 12. Key sensitivity (NPCR) between $10^{-16}$ mismatch keys encrypted images with the same plain image (unit: %).

|               | (C,C1) | (C,C2) | (C,C3) | (C,C4) | (C,C5) |
|---------------|--------|--------|--------|--------|--------|
| Gray Baboon   | 98.7381| 99.1318| 99.0784| 99.1516| 99.1730|
| Gray Peppers  | 98.7534| 99.1348| 99.0768| 99.1837| 99.1928|
| Cornfield-red | 98.8754| 99.0204| 99.0372| 99.0463| 99.0799|
| Cornfield-green| 98.8922| 99.0189| 99.0280| 99.0433| 99.1028|
| Cornfield-blue | 98.8831| 99.0448| 99.0326| 99.0570| 99.0875|
| Flower-red    | 98.8998| 99.0448| 99.0067| 99.0707| 99.0982|
| Flower-green  | 98.8968| 99.0448| 99.0067| 99.0616| 99.0997|
| Flower-blue   | 98.8876| 99.0402| 99.0219| 99.0677| 99.0875|

TABLE 13. Key sensitivity (UACI) between $10^{-16}$ mismatch keys encrypted images with the same plain image (unit: %).

|               | (C,C1) | (C,C2) | (C,C3) | (C,C4) | (C,C5) |
|---------------|--------|--------|--------|--------|--------|
| Gray Baboon   | 27.6875| 29.7788| 28.6014| 29.2604| 30.3942|
| Gray Peppers  | 27.8173| 30.0814| 28.0646| 29.5463| 30.6199|
| Cornfield-red | 29.2313| 29.6798| 31.3011| 30.3823| 30.3308|
| Cornfield-green| 29.3938| 29.5772| 31.2306| 30.4823| 30.2286|
| Cornfield-blue | 29.2060| 29.5572| 31.1656| 30.4108| 30.1033|
| Flower-red    | 29.4148| 29.5356| 31.3031| 30.6919| 30.1956|
| Flower-green  | 29.1734| 29.3315| 31.2348| 30.5592| 29.8955|
| Flower-blue   | 28.9919| 29.2117| 30.9076| 30.3950| 29.9450|

Assume that the parameters in key are slightly changed to obtain the mismatch key $Ekey1$, $Ekey2$, $Ekey3$, $Ekey4$, $Ekey5$:

$$Ekey1 = \{x_0 + \Delta, y_0, z_0, u_0, w_0, h_1, k_1, k_2, a, b, d, ST, st\}$$

$$Ekey2 = \{x_0, y_0 + \Delta, z_0, u_0, w_0, h_1, k_1, k_2, a, b, d, ST, st\}$$

$$Ekey3 = \{x_0, y_0, z_0 + \Delta, u_0, w_0, h_1, k_1, k_2, a, b, d, ST, st\}$$

$$Ekey4 = \{x_0, y_0, z_0, u_0, w_0 + \Delta, h_1, k_1, k_2, a, b, d, ST, st\}$$

$$Ekey5 = \{x_0, y_0, z_0, u_0, w_0 + \Delta, h_1, k_1, k_2, a, b, d, ST, st\}$$

where $\Delta = 10^{-16}$.

To verify the key sensitivity of this algorithm, we conduct the following experiments:

$$C = encrypt(key, P)$$
$$C1 = encrypt(Ekey1, P)$$
$$C2 = encrypt(Ekey2, P)$$
$$C3 = encrypt(Ekey3, P)$$
$$C4 = encrypt(Ekey4, P)$$
$$C5 = encrypt(Ekey5, P)$$

(31)
TABLE 15. Key sensitivity (NPCR) between $10^{-16}$ mismatch keys decrypted images with the same encrypted image (unit: %).

| Gray Baboon | R1      | R2      | R3      | R4      | R5      |
|-------------|---------|---------|---------|---------|---------|
| D           | 98.7381 | 99.1318 | 99.0784 | 99.1516 | 99.1730 |
| R1          | -       | 99.0753 | 99.0860 | 99.1364 | 99.0479 |
| R2          | -       | -       | 98.6328 | 98.1110 | 99.0845 |
| R3          | -       | -       | -       | 99.0738 | 99.1730 |
| R4          | -       | -       | -       | -       | 99.1760 |

| Gray Peppers | R1      | R2      | R3      | R4      | R5      |
|--------------|---------|---------|---------|---------|---------|
| D            | 98.7354 | 99.1348 | 99.0768 | 99.1837 | 99.1928 |
| R1           | -       | 99.0646 | 99.0692 | 99.1196 | 99.0585 |
| R2           | -       | -       | 98.6740 | 98.1369 | 99.0982 |
| R3           | -       | -       | -       | 99.0875 | 99.1730 |
| R4           | -       | -       | -       | -       | 99.1608 |

| Cornfield-red | R1      | R2      | R3      | R4      | R5      |
|---------------|---------|---------|---------|---------|---------|
| D             | 98.8754 | 99.0204 | 99.0372 | 99.0463 | 99.0799 |
| R1            | -       | 99.0814 | 99.0921 | 99.1180 | 99.0662 |
| R2            | -       | -       | 98.6465 | 98.1308 | 99.0662 |
| R3            | -       | -       | -       | 99.0799 | 99.1760 |
| R4            | -       | -       | -       | -       | 99.1455 |

| Cornfield-green | R1      | R2      | R3      | R4      | R5      |
|-----------------|---------|---------|---------|---------|---------|
| D               | 98.8922 | 99.0189 | 99.0280 | 99.0453 | 99.1028 |
| R1              | -       | 99.0997 | 99.0936 | 99.1241 | 99.0494 |
| R2              | -       | -       | 98.6908 | 98.1277 | 99.0814 |
| R3              | -       | -       | -       | 99.0784 | 99.1714 |
| R4              | -       | -       | -       | -       | 99.1653 |

| Cornfield-blue | R1      | R2      | R3      | R4      | R5      |
|----------------|---------|---------|---------|---------|---------|
| D              | 98.8831 | 99.0448 | 99.0326 | 99.0570 | 99.0875 |
| R1             | -       | 99.0646 | 99.0845 | 99.1241 | 99.0646 |
| R2             | -       | -       | 98.6888 | 98.1247 | 99.0997 |
| R3             | -       | -       | -       | 99.0784 | 99.1653 |
| R4             | -       | -       | -       | -       | 99.1638 |

| Flower-red     | R1      | R2      | R3      | R4      | R5      |
|----------------|---------|---------|---------|---------|---------|
| D              | 98.8998 | 99.0448 | 99.0067 | 99.0707 | 99.0982 |
| R1             | -       | 99.0768 | 99.0814 | 99.1379 | 99.0453 |
| R2             | -       | -       | 98.6206 | 98.1125 | 99.0875 |
| R3             | -       | -       | -       | 99.0707 | 99.1644 |
| R4             | -       | -       | -       | -       | 99.1699 |

| Flower-green   | R1      | R2      | R3      | R4      | R5      |
|----------------|---------|---------|---------|---------|---------|
| D              | 98.8968 | 99.0448 | 99.0067 | 99.0616 | 99.0997 |
| R1             | -       | 99.0768 | 99.0829 | 99.1379 | 99.0463 |
| R2             | -       | -       | 98.6252 | 98.1125 | 99.0891 |
| R3             | -       | -       | -       | 99.0707 | 99.1699 |
| R4             | -       | -       | -       | -       | 99.1699 |

| Flower-blue    | R1      | R2      | R3      | R4      | R5      |
|----------------|---------|---------|---------|---------|---------|
| D              | 98.8876 | 99.0402 | 99.0219 | 99.0677 | 99.0875 |
| R1             | -       | 99.0631 | 99.0646 | 99.1165 | 99.0601 |
| R2             | -       | -       | 98.6710 | 98.1369 | 99.0982 |
| R3             | -       | -       | -       | 99.0891 | 99.1745 |
| R4             | -       | -       | -       | -       | 99.1638 |

\[
\begin{align*}
D &= \text{decrypt}(\text{key}, C) \\
R1 &= \text{decrypt}(Ekey1, C) \\
R2 &= \text{decrypt}(Ekey2, C) \\
R3 &= \text{decrypt}(Ekey3, C) \\
R4 &= \text{decrypt}(Ekey4, C) \\
R5 &= \text{decrypt}(Ekey5, C)
\end{align*}
\]

where encrypt() and decrypt() represent the encryption and decryption algorithms function proposed in this article. \( P \) is the plain image. \( C, C1, C2, C3, C4, \) and \( C5 \) are encrypted images using \( \text{key}, Ekey1, Ekey2, Ekey3, Ekey4, Ekey5 \) to encrypt the image “Peppers”, respectively. \( D \) is the correct decrypted image decrypted using the legal key. \( R1, R2, R3, R4, \) and \( R5 \) are the error decrypted images obtained by using the mismatched keys \( Ekey1, Ekey2, Ekey3, Ekey4, Ekey5 \) to perform the decryption operation. Fig.13 shows the plain image “Peppers” and encrypted images \( C1 – C5 \). Fig.14 shows the histogram of the plain image “Peppers” and encrypted images \( C1 – C5 \). Fig.15 shows the plain image “Flower” and encrypted images \( C1 – C5 \). Fig.16 shows the histogram of the three color components of the encrypted images \( C1 – C5 \). Fig.17 illustrates correctly decrypted image \( D \) and incorrectly decrypted images \( R1 – R5 \) of the encrypted image “Peppers”. Fig.18 illustrates the histogram of the encrypted image “Peppers”’s correctly decrypted image \( D \) and incorrectly decrypted images \( R1 – R5 \). Fig.19 illustrates correctly decrypted image \( D \) and incorrectly decrypted images \( R1 – R5 \) of the encrypted image “Flower”.

VOLUME 8, 2020
Table 16. Key sensitivity (UACI) between $10^{-16}$ mismatch keys decrypted images with the same encrypted image (unit: %).

| Gray Baboon | R1 | R2 | R3 | R4 | R5 |
|-------------|----|----|----|----|----|
| D           | 27.8436 | 25.7418 | 27.8297 | 27.5105 | 28.3818 |
| R1          | - | 33.9123 | 33.6249 | 33.6402 | 34.7334 |
| R2          | - | - | 33.3379 | 33.8437 | 33.7116 |
| R3          | - | - | - | 34.6837 | 33.8981 |
| R4          | - | - | - | - | 33.4315 |

| Gray Peppers | R1 | R2 | R3 | R4 | R5 |
|--------------|----|----|----|----|----|
| D            | 30.3654 | 29.3499 | 30.1880 | 30.1143 | 30.7350 |
| R1           | - | 34.0411 | 33.4832 | 33.6325 | 34.3251 |
| R2           | - | - | 33.4833 | 33.1842 | 33.7117 |
| R3           | - | - | - | 34.5386 | 33.7415 |
| R4           | - | - | - | - | 33.1357 |

| Cornfield-red | R1 | R2 | R3 | R4 | R5 |
|---------------|----|----|----|----|----|
| D             | 28.5940 | 28.4918 | 28.4592 | 28.4601 | 28.5149 |
| R1            | - | 34.1036 | 33.3811 | 33.6254 | 34.1524 |
| R2            | - | - | 33.4746 | 33.2565 | 33.8889 |
| R3            | - | - | - | 34.4290 | 33.5819 |
| R4            | - | - | - | - | 33.0417 |

| Cornfield-green | R1 | R2 | R3 | R4 | R5 |
|-----------------|----|----|----|----|----|
| D               | 28.3811 | 28.6222 | 28.5351 | 28.5880 | 28.5998 |
| R1              | - | 34.0891 | 33.3274 | 33.6027 | 34.0663 |
| R2              | - | - | 33.5614 | 33.1868 | 33.8482 |
| R3              | - | - | - | 34.4142 | 33.5742 |
| R4              | - | - | - | - | 33.0644 |

| Cornfield-blue | R1 | R2 | R3 | R4 | R5 |
|----------------|----|----|----|----|----|
| D              | 34.0620 | 34.0289 | 34.0004 | 33.8360 | 34.0687 |
| R1             | - | 34.1445 | 33.3449 | 33.6138 | 34.1570 |
| R2             | - | - | 33.3940 | 33.1137 | 33.6510 |
| R3             | - | - | - | 34.5414 | 33.5854 |
| R4             | - | - | - | - | 33.0158 |

| Flower-red     | R1 | R2 | R3 | R4 | R5 |
|----------------|----|----|----|----|----|
| D              | 30.9640 | 30.9749 | 30.9163 | 30.8703 | 30.8268 |
| R1             | - | 34.1460 | 33.4664 | 33.7807 | 34.0047 |
| R2             | - | - | 34.4899 | 33.2828 | 33.8123 |
| R3             | - | - | - | 34.6155 | 33.6669 |
| R4             | - | - | - | - | 33.1552 |

| Flower-green   | R1 | R2 | R3 | R4 | R5 |
|----------------|----|----|----|----|----|
| D              | 31.3229 | 31.2712 | 31.2786 | 31.2005 | 31.2917 |
| R1             | - | 34.1402 | 33.3278 | 33.6744 | 34.1616 |
| R2             | - | - | 33.4019 | 33.1159 | 33.8382 |
| R3             | - | - | - | 34.5155 | 33.6349 |
| R4             | - | - | - | - | 33.1883 |

| Flower-blue    | R1 | R2 | R3 | R4 | R5 |
|----------------|----|----|----|----|----|
| D              | 35.3269 | 35.6303 | 35.4391 | 35.3251 | 35.4788 |
| R1             | - | 34.1156 | 33.3265 | 33.6626 | 34.1586 |
| R2             | - | - | 33.4164 | 33.3322 | 33.8719 |
| R3             | - | - | - | 34.6024 | 33.5639 |
| R4             | - | - | - | - | 33.1663 |

Fig. 20 illustrates the histogram of three color components of the encrypted image "Flower"s incorrectly decrypted images $R1 – R5$.

To further analyze the key sensitivity characteristics of our algorithm, we calculated the NPCR, UACI, and BACI values between the encrypted images using different encryption keys based on the same plain image. The calculation results are listed in Table 12 to Table 14.

It can be seen from Table 12 to Table 14 that when we use different keys to encrypt the same plain image, even if the encryption keys are only slightly different from each other by $10^{-16}$, we will get completely different encrypted images. In other words, the tiny difference in the keys of $10^{-16}$ results in an average value of NPCR between two encrypted images is 99.03%. This shows that our encryption algorithm is key sensitive during the encryption process.

In addition, to further illustrate the key sensitivity of the algorithm in the decryption process, Table 15 to Table 17 compare the NPCR, UACI, and BACI values used to decrypt the image using mismatched decryption keys.

It can be seen from the data in the table that when decrypting with $10^{-16}$ mismatched error keys, the average value of NPCR between the error decrypted image and the correct decrypted image is 99.03%, the average value of NPCR between two error decrypted images is 98.96%. Our decryption algorithm is highly sensitive to the decryption key, even if the decryption key only changes $10^{-16}$, it cannot be decrypted correctly. According to Table 11 to Table 17, it can be seen that the proposed image cryptosystem has strong...
TABLE 17. Key sensitivity (BACI) between $10^{-16}$ mismatch keys decrypted images with the same encrypted image (unit: %).

| Gray Baboon | R1      | R2      | R3      | R4      | R5      |
|-------------|---------|---------|---------|---------|---------|
| D           | 20.2863 | 20.3256 | 20.2089 | 20.1342 | 20.4820 |
| R1          | -       | 26.4856 | 26.8721 | 26.7016 | 27.3432 |
| R2          | -       | -       | 26.6476 | 26.5993 | 26.7489 |
| R3          | -       | -       | -       | 26.7401 | 26.9583 |
| R4          | -       | -       | -       | -       | 27.0097 |
| Gray Peppers| R1      | R2      | R3      | R4      | R5      |
| D           | 22.6831 | 22.8798 | 22.7480 | 22.7222 | 22.8042 |
| R1          | -       | 26.5735 | 26.7545 | 26.6338 | 26.8456 |
| R2          | -       | -       | 26.7500 | 26.7715 | 26.7940 |
| R3          | -       | -       | -       | 26.6593 | 26.8964 |
| R4          | -       | -       | -       | -       | 26.7091 |
| Cornfield-red| R1      | R2      | R3      | R4      | R5      |
| D           | 21.3349 | 21.2222 | 21.2135 | 21.1892 | 21.2526 |
| R1          | -       | 26.6189 | 26.7891 | 26.6509 | 26.7586 |
| R2          | -       | -       | 26.6658 | 26.9408 | 26.8558 |
| R3          | -       | -       | -       | 26.5383 | 26.7380 |
| R4          | -       | -       | -       | -       | 26.6729 |
| Cornfield-green| R1     | R2      | R3      | R4      | R5      |
| D           | 21.2004 | 21.1553 | 21.0708 | 21.1512 | 21.1867 |
| R1          | -       | 26.6017 | 26.7028 | 26.6072 | 26.8386 |
| R2          | -       | -       | 26.8088 | 26.8675 | 26.8237 |
| R3          | -       | -       | -       | 26.4737 | 26.6271 |
| R4          | -       | -       | -       | -       | 26.6643 |
| Cornfield-blue| R1     | R2      | R3      | R4      | R5      |
| D           | 26.5729 | 25.4497 | 25.5295 | 25.8673 | 26.4377 |
| R1          | -       | 26.5940 | 26.7600 | 26.6304 | 26.8659 |
| R2          | -       | -       | 26.6659 | 26.8688 | 26.8353 |
| R3          | -       | -       | -       | 26.7134 | 26.6827 |
| R4          | -       | -       | -       | -       | 26.7164 |
| Flower-red | R1      | R2      | R3      | R4      | R5      |
| D           | 24.1874 | 24.1093 | 24.1394 | 24.1383 | 24.0963 |
| R1          | -       | 26.6278 | 26.7267 | 26.7574 | 26.6861 |
| R2          | -       | -       | 26.8945 | 26.9236 | 26.8754 |
| R3          | -       | -       | -       | 26.6934 | 26.6571 |
| R4          | -       | -       | -       | -       | 26.7618 |
| Flower-green| R1     | R2      | R3      | R4      | R5      |
| D           | 23.7647 | 23.6351 | 23.6057 | 23.7025 | 23.7583 |
| R1          | -       | 26.6869 | 26.7192 | 26.7576 | 26.8614 |
| R2          | -       | -       | 26.7264 | 26.8981 | 26.8530 |
| R3          | -       | -       | -       | 26.6356 | 26.7000 |
| R4          | -       | -       | -       | -       | 26.8259 |
| Flower-blue| R1     | R2      | R3      | R4      | R5      |
| D           | 27.1437 | 27.2730 | 27.1405 | 27.1916 | 27.1894 |
| R1          | -       | 26.6586 | 26.7221 | 26.6809 | 26.7982 |
| R2          | -       | -       | 26.7844 | 26.9931 | 26.9277 |
| R3          | -       | -       | -       | 26.6931 | 26.7453 |
| R4          | -       | -       | -       | -       | 26.7839 |

sensitivities to secret key and plaintext images. Therefore, the proposed system can not only resist the differential attack but also has the ability to resist the known-plaintext attack and chosen-plaintext attack.

G. KNOWN PLAINTEXT ATTACK AND CHOSEN PLAINTEXT ATTACK

For an image, its information entropy and variance value are unique. The algorithm proposed in this article uses the information entropy and variance of the plain image as the terms of the calculation formula (13) for the starting position $p(r)$ of the chaotic matrix in the diffusion stage, so the key streams generated when different images are encrypted are different. Therefore, the algorithm proposed in this article realizes that the key is related to the plain image. From the data in Table 11, it can be seen that the proposed image cryptosystem has strong sensitivities to plaintext images. Besides, we performed two rounds of the diffusion process and tested the plaintext sensitivity of all-black and all-white images. The NPCR and UACI values of an all-black image are 99.6017% and 34.7739%, respectively, and the NPCR and UACI of an all-white image are 99.5605% and 34.0198%, respectively. So the proposed system can resist known-plaintext attack and chosen-plaintext attack effectively [45].

H. NIST TEST

NIST SP800-22 [48] is a series of guidelines on information security issued by the National Institute of Standards and Technology (NIST) in the United States. It is a statistical test suite for random and pseudorandom number generators for cryptographic applications. We performed a NIST test for $65536 \times 8$ bits binary numbers of the generated encrypted image “Peppers”. The test results are listed in Table 18. It can be seen from the table that the generated cipher image passed...
all 16 tests, which shows that the sequence generated by our algorithm is random.

### I. CROPPING AND NOISE ATTACKS

If the cipher image is damaged during data transmission, image information will not be completely transmitted. So an excellent encryption algorithm should have good shear resistance. In this experiment, the encrypted images “Peppers” and “Flower” are cropped 1/16, 1/8, 1/4, and 1/2, and the cropped position and size are shown in Fig.21(a)-(d) and Fig.22(a)-(d). The corresponding decryption results are shown in Fig.21(e)-(g) and Fig.22(e)-(g). To further observe the difference between the cropped decrypted image and the normal decrypted image, each picture in Fig.21(e)-(g), Fig.22(e)-(g) and its corresponding normal decrypted image are calculated. The MSE and PSNR values of the image are listed in Table 19 and Table 20. From the experimental results, we can see that our encryption algorithm can recover the cropped image, and has a good ability to resist the cropping attack.

Noise attacks can tamper with the pixels of an image. Although it may only change a few individual pixels in the encrypted image, it also has a certain impact on image recovery. Therefore, a good encryption algorithm needs to have the ability to resist noise attacks. We use the 0.01,


**Table 19.** MSE and PSNR values of the cropped decrypted image and the normal decrypted image "Peppers".

| Decrypted image | MSE  | PSNR  |
|-----------------|------|-------|
| Cropped 1/16    | 535.9013 | 20.8400 |
| Cropped 1/8     | 1078.9898 | 17.8006 |
| Cropped 1/4     | 2168.4729 | 14.7693 |
| Cropped 1/2     | 4353.5334 | 11.7424 |

**Table 20.** MSE and PSNR values of the cropped decrypted image and the normal decrypted image "Flower".

| Decrypted image | Red  | Green | Blue  |
|-----------------|------|-------|-------|
| MSE-Cropped 1/16 | 5476.2284 | 5267.9620 | 14307.0487 |
| MSE-Cropped 1/8  | 5641.4147 | 5435.4230 | 14098.8336 |
| MSE-Cropped 1/4  | 5928.5991 | 5774.3994 | 13665.3269 |
| MSE-Cropped 1/2  | 6531.5672 | 6425.3105 | 12839.1207 |
| PSNR-Cropped 1/16 | 10.7460 | 10.9144 | 6.6753 |
| PSNR-Cropped 1/8  | 10.6160 | 10.7785 | 6.6390 |
| PSNR-Cropped 1/4  | 10.4013 | 10.5157 | 6.7746 |
| PSNR-Cropped 1/2  | 9.9806  | 10.0519 | 7.0455  |

**Table 21.** MSE and PSNR values between the decrypted image after the noise attack and the normal decrypted image "Peppers".

| Decrypted image | MSE  | PSNR  |
|-----------------|------|-------|
| 0.01 noise      | 89.2278 | 28.6258 |
| 0.05 noise      | 422.3311 | 21.8743 |
| 0.1 noise       | 886.7123 | 18.6530 |
| 0.3 noise       | 2550.5561 | 14.0645 |

**Table 22.** MSE and PSNR values between the decrypted image after the noise attack and the normal decrypted image "Flower".

| Decrypted image | Red  | Green | Blue  |
|-----------------|------|-------|-------|
| MSE-0.01 noise  | 5349.6070 | 5139.6822 | 14503.6693 |
| MSE-0.05 noise  | 5446.0230 | 5241.2104 | 14395.6619 |
| MSE-0.1 noise   | 5583.0796 | 5386.5879 | 14216.1748 |
| MSE-0.3 noise   | 6050.7180 | 5945.5947 | 13573.9390 |
| PSNR-0.01 noise | 10.8476 | 11.0214 | 6.5160 |
| PSNR-0.05 noise | 10.7700 | 10.9365 | 6.5485 |
| PSNR-0.1 noise  | 10.6621 | 10.8177 | 6.6030 |
| PSNR-0.3 noise  | 10.3127 | 10.3889 | 6.8037 |

0.05, 0.1, and 0.3 "salt and pepper noise" to attack the "Peppers" and the "Flower" encrypted images, the results are shown in Fig.23(a)-(d) and Fig.24(a)-(d). Fig.23(e)-(h) and Fig.24(e)-(h) are the corresponding decryption results after these noise attacks. To further observe the difference between the decrypted image attacked by noise and the normal decrypted image, each picture in Fig.23(e)-(h) and Fig.24(e)-(h) and its corresponding normal decrypted image are calculated. The MSE and PSNR values are listed in Table 21 and Table 22.

The experimental results demonstrate that the proposed image encryption algorithm is robust against noise and cropping processing, which makes the proposal suitable for harsh transmission conditions.

**V. CONCLUSION**

In this article, a new image encryption algorithm based on bit-level scrambling and dynamic overlap diffusion is proposed. To achieve a better scrambling effect, Arnold transform is used to scramble and change the position and pixel value of image pixels after the bit-level permutation stage. The chaotic matrix generated by the five-dimensional hyperchaotic system is used for the dynamic overlap diffusion step of the image. Because the method of plaintext controlling Arnold iterative times and the method of information entropy and variance updating of diffusion key are used, the attacker cannot crack this algorithm via known-plaintext and chosen-plaintext attacks. We also perform security analyses, such as an information entropy analysis, a statistical analysis, a sensitivity analysis, a differential attack analysis, a noise attack analysis, and so on. The experimental results show that the algorithm proposed in this article has high security, and can be applied to commerce, medical, and other fields to encrypt digital images.

In view of the large amount of image information, when a large number of encrypted images are transmitted or stored, they may face the problem of occupying more bandwidth during the transmission process. Therefore, the future work is to increase the transmission rate of large batches of images while ensuring the security of the cryptographic system, such as reducing the network transmission pressure by compressing cryptographic images before transmission.

**REFERENCES**

[1] S. F. Raza and V. Satpute, “A novel bit permutation-based image encryption algorithm,” *Nonlinear Dyn.*, vol. 95, no. 2, pp. 859–873, Jan. 2019.

[2] H. M. Ghadirli, A. Nodrzi, and R. Enayatifar, “An overview of encryption algorithms in color images,” *Signal Process.*, vol. 164, pp. 163–185, Nov. 2019.

[3] A.-H. Zhu and L. Lian, “Improving for chaotic image encryption algorithm based on Logistic map,” in *Proc. 2nd Conf. Environ. Sci. Inf. Appl. Technol.*, Jul. 2010, pp. 211–214.

[4] N. K. Pareek, V. Patidar, and K. K. Sud, “Image encryption using chaotic logistic map,” *Image Vis. Comput.*, vol. 24, no. 9, pp. 926–934, Sep. 2006.

[5] G. Ye, P. Chen, X. Huang, and Q. Mei, “An efficient pixel-level chaotic image encryption algorithm,” *Nonlinear Dyn.*, vol. 94, no. 4, p. 745, 2018.

[6] Y. Xiong, C. Quan, and C. J. Tay, “Multiple image encryption scheme based on pixel exchange operation and vector decomposition,” *Opt. Lasers Eng.*, vol. 101, pp. 113–121, Feb. 2018.

[7] L. Gong, K. Qiu, C. Deng, and N. Zhou, “An image compression and encryption algorithm based on chaotic system and compressive sensing,” *Opt. Laser Technol.*, vol. 115, pp. 257–267, Jul. 2019.

[8] L. Xu, X. Gou, Z. Li, and J. Li, “A novel chaotic image encryption algorithm using block scrambling and dynamic index based diffusion,” *Opt. Lasers Eng.*, vol. 91, pp. 41–52, Apr. 2017.

[9] Z. Hua and Y. Zhou, “Design of image cipher using block-scrambling and image filtering,” *Inf. Sci.*, vol. 396, pp. 97–113, Aug. 2017.

[10] X. Wang, L. Liu, and Y. Zhang, “A novel chaotic block image encryption algorithm based on dynamic random growth technique,” *Opt. Lasers Eng.*, vol. 66, pp. 10–18, Mar. 2015.

[11] S. Sun, “A novel hyperchaotic image encryption scheme based on DNA encoding, pixel-level scrambling and bit-level scrambling,” *IEEE Photon. J.*, vol. 10, no. 2, pp. 1–14, Apr. 2018.

[12] X. Chai, Z. Gan, K. Yang, Y. Chen, and X. Liu, “An image encryption algorithm based on the memristive hyperchaotic system, cellular automata and DNA sequence operations,” *Signal Process., Image Commun.*, vol. 52, pp. 6–19, Mar. 2017.

[13] X. Ouyang, Y. Luo, J. Liu, L. Cao, and Y. Liu, “A color image encryption method based on memristive hyperchaotic system and DNA encryption,” *Int. J. Mod. Phys. B*, vol. 34, no. 4, Feb. 2020, Art. no. 2050014.

[14] Y. Zhou, W. Cao, and C. L. Philip Chen, “Image encryption using binary bitplane,” *Signal Process.*, vol. 100, pp. 197–207, Jul. 2014.

[15] C. Fu, B.-B. Lin, Y.-S. Miao, X. Liu, and J.-J. Chen, “A novel chaos-based bit-level permutation scheme for digital image encryption,” *Opt. Commun.*, vol. 284, no. 23, pp. 5415–5423, Nov. 2011.

[16] X. Li, Z. Xie, J. Wu, and T. Li, “Image encryption based on dynamic filtering and bit cuboid operations,” *Complexity*, vol. 2019, pp. 1–16, Feb. 2019.
Z. Hua, Y. Zhou, C.-M. Pun, and C. L. P. Chen, “2D sine logistic modulation,” in Proc. Nat. Comput. Conf., 1997.

H. Lipmaa, “IDEA: A cipher for multimedia architectures?” in Proc. Int. Workshop Sel. Areas Cryptogr., 2000, pp. 248–263.

A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, Handbook of Applied Cryptography, Boca Raton, FL, USA: CRC Press, 1997.

S. Li, G. Chen, A. Cheung, B. Bhargava, and K.-T. Lo, “On the design of perceptual MPEG-video encryption algorithms,” IEEE Trans. Circuits Syst. Video Technol., vol. 17, no. 2, pp. 214–223, Feb. 2007.

T. Li, M. Yang, J. Wu, and X. Jing, “A novel image encryption algorithm based on a fractional-order hyperchaotic system and DNA computing,” Complexity, vol. 2017, pp. 1–13, Jan. 2017.

Z.-L. Zhu, W. Zhang, K.-W. Wong, and H. Yu, “A chaos-based symmetric image encryption scheme using a bit-level permutation,” Inf. Sci., vol. 181, no. 6, pp. 1171–1186, Mar. 2011.

T.-Y. Li and J. A. Yorke, “Period three implies chaos,” Amer. Math. Monthly, vol. 82, no. 10, p. 985, Dec. 1975.

C. E. Shannon, “Communication theory of secrecy systems,” Bell Syst. Tech. J., vol. 28, no. 4, pp. 656–715, Oct. 1949.

J. Fridrich, “Image encryption based on chaotic maps,” in Proc. IEEE Int. Conf. Syst., Man., Cybern. Comput. Cybern. Simulation, Oct. 2002, pp. 1105–1110.

J. Fridrich, “Symmetric ciphers based on two-dimensional chaotic maps,” Int. J. Bifurcation Chaos, vol. 8, no. 6, pp. 1259–1264, Jun. 1998.

H. Gao, Y. Zhang, S. Liang, and D. Li, “A novel chaotic algorithm for image encryption,” Chaos, Solitons Fractals, vol. 29, no. 2, pp. 393–399, 2006.

Z. Hua, Y. Zhou, C.-M. Pun, and C. L. P. Chen, “2D sine logistic modulation map for image encryption,” Inf. Sci., vol. 297, pp. 80–94, Mar. 2015.

W. Xiaoyuan, X. Bing, and Z. Huaquang, “A multi-ary number communication system based on hyperchaotic system of 6th-order cellular neural network,” Commun. Nonlinear Sci. Numer. Simul., vol. 15, no. 1, pp. 124–133, Jan. 2010.

Q. Yang and C. Chen, “A 5D hyperchaotic system with three positive Lyapunov exponents,” Int. J. Bifurcation Chaos, vol. 23, no. 6, Jun. 2013, Art. no. 1350109.

C. Zhu, “A novel image encryption scheme based on improved hyper-chaotic sequences,” Opt. Commun., vol. 285, no. 1, pp. 29–37, Jan. 2012.

H. Zhao, S. Xie, J. Zhang, and T. Wu, “Efficient image encryption using two-dimensional enhanced hyperchaotic Henon map,” J. Electron. Imaging., vol. 23, no. 1, pp. 013033, Mar. 2014.

S. Behnia, A. Akhshani, H. Mahmodi, and A. Akhavan, “A novel algorithm for image encryption based on mixture of chaotic maps,” Chaos, Solitons Fractals, vol. 35, no. 2, pp. 408–419, Jan. 2008.

Q. Yin and C. Wang, “A new chaotic image encryption scheme using breadth-first search and dynamic diffusion,” Int. J. Bifurcation Chaos, vol. 28, no. 4, Apr. 2018, Art. no. 1850047.

L. Xu, Z. Li, J. Li, and W. Hua, “A novel bit-level image encryption algorithm based on chaotic maps,” Opt. Lasers Eng., vol. 78, pp. 17–25, Mar. 2016.

C. Jin and H. Liu, “A color image encryption scheme based on Arnold scrambling and quantum chaotic,” J Netw. Secur., vol. 19, no. 3, pp. 347–357, 2017.

G. Alvarez and S. Li, “Some basic cryptographic requirements for chaos-based cryptosystems,” Int. J. Bifurcation Chaos, vol. 16, no. 8, pp. 2129–2151, Aug. 2006.

X. Wang, X. Zhu, X. Wu, and Y. Zhang, “Image encryption algorithm based on multiple mixed hash functions and cyclic shift,” Opt. Lasers Eng., vol. 107, pp. 370–379, Aug. 2018.

Y. Zhang, A. Chen, Y. Tang, J. Dang, and G. Wang, “ Plaintext-related image encryption algorithm based on perceptron-like network,” Inf. Sci., vol. 526, pp. 180–202, Jul. 2020.

Y. Li, Y. Zhang, and X. Wang, “A symmetric image encryption algorithm based on mixed linear–nonlinear coupled map lattice,” Inf. Sci., vol. 273, pp. 329–351, Jul. 2014.

H. Liu, A. Kadir, and X. Sun, “ Chaos-based fast colour image encryption scheme with true random number keys from environmental noise,” IET Image Process., vol. 11, no. 5, pp. 324–332, Apr. 2017.

X. Wu, C. Bai, and H. Kan, “A new color image cryptosystem via hyper-chaos synchronization,” Commun. Nonlinear Sci. Numer. Simul., vol. 19, no. 6, pp. 1884–1897, Jun. 2014.