Secure and Fast Image Encryption Algorithm Using Hyper-chaos based Key Generator and Vector Operation

Bin Ge¹, Xu Chen², Gang Chen², and Zhihua Shen¹
¹Electronic Information Engineering College, Nantong Vocational University, Nantong, 226007, China
²Institute of Semiconductors, Chinese Academy of Sciences, Beijing, 100083, China

Corresponding author: Bin Ge (bge@mail.ntvu.edu.cn)

ABSTRACT To protect image data privacy, a secure and fast image encryption algorithm using hyper-chaos based key generator and vector operation is proposed. Firstly, we introduce a novel post-process method to create a key matrix for significantly reducing the iterations of the hyperchaotic system from \( W \times H/4 \) to \( 2W \) for an image size of \( W \times H \). And within the post-process, a random selector driven by the piecewise linear chaotic map ensures the good randomness and unpredictability of the generated key matrix. Secondly, the vector operation is employed to accelerate the cipher block chaining based diffusion process, so the plain image can be parallel encrypted row by row and column by column with high efficiency. Thirdly, we utilize a Logistic map to quickly produce an initial vector for the vectorized diffusion process. Finally, the proposed algorithm is evaluated by some common security and performance tests. Experimental results show that the cipher image can pass all tests of NIST SP 800-22 with \( P \)-values \( > 0.01 \), its correlation coefficient between pixels is close to \( 0 \), and the entropy is greater than 7.999. Combined with other results of security tests, we can safely conclude that the proposed image encryption algorithm provides adequate protection against statistical, brute-force, chosen-plaintext, and other common types of attacks. In addition, the time complexity is in the order of \( O(W+H) \) and the average encryption time of 512×512 images is only \( 0.023s \). The results indicate that our algorithm with high security and fast speed can meet the requirements of real-time confidential transmission of massive image data.

INDEX TERMS Image encryption, hyperchaotic system, key generator, vector operation.

I. INTRODUCTION

With the vigorous development of information technology and online social media, digital images have become part of our daily living [1], [2]. However, numerous images are produced and transmitted daily without adequate protection over the Internet [3]–[5]. Meanwhile, with the gradual popularization of IoT, devices are creating more and more digital images at an alarming rate, securing image data in complex open transmission channels has become a common concern [6]–[8]. At present, encryption and watermarking techniques have become the two major thrusts of image security research. But the small capacity of data hiding makes the watermarking technique more suitable to protect the copyright of digital images [9], [10]. Certainly, the encryption technique makes it possible for us to transmit massive image data without leaking any useful information whether in wireless or wired communications [7]. Although block ciphers, such as 3DES and AES, are commonly adopted [11], [12], extensive practical results [13], [14] indicate that these algorithms designed for textual data cannot effectively encrypt images partially given the low processing speed due to the large amount and high redundancy of image data [15]. Hence, many novel image orientated encryption algorithms have been proposed [16]–[19] with chaos-based image encryption being the most representative approach.

In 1989, Matthews [20] proposed the first stream cipher algorithm using a Logistic map. Since the development of chaos control and chaos synchronization in 1990 [21], the application of chaotic systems to cryptosystems has been
increasingly explored. In 1998, Fridrich [22] used a two-dimensional Baker’s map to develop a common permutation-diffusion structure to fully hide the relationship between a plain image and its corresponding cipher image. Although various techniques have been introduced to enhance the algorithm security [23]–[26], some inevitable risks, such as the small key space, low linear complexity, and short periodic orbit [27], [28], limit the application of low-dimensional chaotic map to image encryption algorithms. However, the theory of hyper-chaos may be helpful to overcome these limitations [29]–[32]. In general, a hyperchaotic system with more than two positive Lyapunov exponents can exhibit more complex attractors, higher unpredictability, and stronger randomness. Moreover, it typically needs more than four state values for iteration, naturally providing a large key space for the corresponding cryptosystem.

In 2008, Gao et al. [33] proposed the first hyperchaotic image encryption algorithm, but it was vulnerable to the chosen-plaintext attacks [34] because its key sequence was independent of the plain image. Thereafter, Li et al. [35] proposed a plaintext-related image encryption algorithm that obtains the initial values of the hyperchaotic system from nine original pixels. However, the sensitivity to variations in other pixels by the application of one-round diffusion was inadequate. A more secure approach should adopt at least two-round cipher block chaining (CBC) based diffusion for the cipher image to relate to every original pixel. But on the other hand, the CBC based diffusion process was serial in common, which led to a practical problem of low encryption speed [36], [37]. Thus, in 2017, Yuan et al. [38] introduced a pixel classification method to parallelize the CBC based diffusion process. However, the insufficient parallelism degree and the usage of rotation resulted in an unsatisfactory efficiency. Then, in 2019, Çavuşoğlu et al. [39] depended on the multi-thread technique to accelerate encryption after dividing the plain image into several subsequences, but the lack of diffusion between threads would cause security issues. In 2020, Zhou et al. [40] converted the original image into multiple 4×4 blocks to encrypt 16 pixels in parallel using a key mask and XOR operation. But only one-round diffusion rendered the proposed algorithm vulnerable to the chosen-plaintext attack and chosen-ciphertext attack. Also in 2020, Zhao et al. [41] presented a highly parallelized algorithm by extending the CBC technique from pixel level to row level, the diffusion process in column remained serialized, and the overall diffusion process could only let pixel information influence its lower-right corner. In general, there is still no appropriate image encryption algorithm to balance security and speed.

Moreover, few studies are available on the time consumption of hyper-chaos based key generators, but it is quite important to improve the efficiency of the image encryption algorithm. On closer inspection, we find that the key generation process even takes longer than the diffusion process [38]–[41]. In order to solve the above problems, we propose a secure and fast image encryption algorithm using hyper-chaos based key generator and vector operation. Firstly, the key generation process is redesigned to produce a key matrix according to the size of the plain image. By applying a post-process method, the number of iterations in the hyperchaotic system can be substantially reduced from W×H/4 to 2W. Taking the two-dimensional structure of the image into consideration, we introduce the vector operation into the CBC based diffusion process, then pixels on a row or column can be encrypted in parallel. Furthermore, the cipher image is highly sensitive to every pixel change through four-round chained diffusion in the vertical direction and horizontal direction. Detailed results and analyses of the security and efficiency performance show that the proposed algorithm can provide fast encryption and can resist various cryptographic attacks such as statistical, brute-force, and chosen-plaintext attacks.

Briefly, the main contributions of this paper can be summarized as follows.

(1) Based on a hyperchaotic system and vector operation, we propose a novel image encryption algorithm that can provide high security protection and fast encryption speed for massive image data transmission on the public channel.

(2) A novel post-process method for the hyper-chaos based key generator is presented. By using bitwise XOR operation and circular shift operation on key stream, a W×H key matrix can be obtained with a 4D hyperchaotic system only iterating 2W times. More importantly, a random selector within the post-process guarantees the randomness and unpredictability of the key matrix.

(3) We utilize the vector operation to accelerate the CBC based diffusion process. Hence, the plain image can be encrypted by row in the vertical direction and by column in the horizontal direction. Moreover, the update of initial vector between two diffusion rounds during the four-round encryption process ensures adequate diffusion, and thus even minor changes of any pixel can significantly influence the results of the encryption.

The remainder of this paper is organized as follows. Section II presents the preliminaries for this study. In Section III, we introduce the proposed image encryption algorithm. In Section IV, we report and analyze the experimental results of the proposed algorithm. In Section V, we further test the proposed algorithm on color images. Finally, we draw conclusions in Section VI.

II. PRELIMINARIES

In this section, all the chaotic systems for constructing the proposed image encryption algorithm are presented, and the mechanism of vector operation is also described.
A. HYPERCHAOTIC LORENZ SYSTEM

Nowadays, various new types of hyperchaotic systems [42]–[45] have emerged. But as one of the most often used hyperchaotic systems to construct secure image encryption algorithms, the hyperchaotic Lorenz system (HCLS) has been proved to have good inherent randomness, high initial sensitivity, and large key space [46], [47]. Moreover, many observations have verified that HCLS keeps chaos character with long period in the digital system [29], [33], [35]–[37], [46], [47]. In this study, we add a nonlinear controller \( \dot{x}_4 = -x_1 x_3 + d x_4 \) on the original Lorenz system [47], then the HCLS equation is given by

\[
\begin{align*}
\dot{x}_1 &= a(x_2 - x_1) + x_4, \\
\dot{x}_2 &= c x_1 - x_2 - x_1 x_3, \\
\dot{x}_3 &= x_1 x_2 - b x_3, \\
\dot{x}_4 &= -x_1 x_3 + d x_4,
\end{align*}
\]

where \( a, b, c, \) and \( d \) are control parameters of HCLS.

For \( a=10, b=8/3, c=28, \) and \( d=-1, \) the system contains two positive Lyapunov exponents [48]: \( \lambda_1=0.3381 \) and \( \lambda_2=0.1586. \) Hence, the system evolves into a hyperchaotic state and can generate an extremely complex and dense motion trajectory, as shown in Fig. 1. Therefore, we can rely on the HCLS to implement a secure pseudorandom number generator with a long period and high linear complexity.

\[ \lambda_1 = 0.3381, \lambda_2 = 0.1586 \]

\[ a=10, b=8/3, c=28, d=-1 \]

\[ x_4 = -x_1 x_3 + d x_4 \]

\[ \dot{x}_1 = -2 x_2 x_3 - 8 x_1 + 28 x_1 x_3 \]

\[ \dot{x}_2 = 9 x_1 - x_2 - x_1 x_3 \]

\[ \dot{x}_3 = x_1 x_2 - 3 x_3 \]

\[ \dot{x}_4 = -x_1 x_3 + d x_4 \]

\[ \lambda_1 = 0.3381, \lambda_2 = 0.1586 \]

\[ a=10, b=8/3, c=28, d=-1 \]

\[ x_4 = -x_1 x_3 + d x_4 \]

\[ \dot{x}_1 = -2 x_2 x_3 - 8 x_1 + 28 x_1 x_3 \]

\[ \dot{x}_2 = 9 x_1 - x_2 - x_1 x_3 \]

\[ \dot{x}_3 = x_1 x_2 - 3 x_3 \]

\[ \dot{x}_4 = -x_1 x_3 + d x_4 \]

\[ \lambda_1 = 0.3381, \lambda_2 = 0.1586 \]

\[ a=10, b=8/3, c=28, d=-1 \]

\[ x_4 = -x_1 x_3 + d x_4 \]

\[ \dot{x}_1 = -2 x_2 x_3 - 8 x_1 + 28 x_1 x_3 \]

\[ \dot{x}_2 = 9 x_1 - x_2 - x_1 x_3 \]

\[ \dot{x}_3 = x_1 x_2 - 3 x_3 \]

\[ \dot{x}_4 = -x_1 x_3 + d x_4 \]

\[ \lambda_1 = 0.3381, \lambda_2 = 0.1586 \]

\[ a=10, b=8/3, c=28, d=-1 \]

\[ x_4 = -x_1 x_3 + d x_4 \]

B. PIECEWISE LINEAR CHAOTIC MAP

The piecewise linear chaotic map (PWLCM) [49] is given by

\[ y(i+1) = F_p(y(i)) = \begin{cases} 
\frac{y(i)}{p}, & 0 < y(i) < p \\
\frac{(y(i) - p)}{(0.5 - p)}, & p \leq y(i) < 0.5 \\
F_p(1 - y(i)), & 0.5 \leq y(i) < 1
\end{cases} \]

where \( p \) is a control parameter and \( y(i) \in (0,1) \).

The PWLCM evolves into a chaotic state, when \( p \) is in the range of 0 and 0.5. In this paper, we let \( p=0.25 \) for the research. Since the PWLCM has a very simple structure, and can rapidly generate a sequence of uniform distribution, we employ it in a fast random selector for post-processing the pseudorandom numbers to obtain a secure key matrix according to the size of the plain image.

C. LOGISTIC MAP

The Logistic map (LM) [50] is a famous one-dimensional chaotic system given by

\[ z(i+1) = \mu z(i)(1 - z(i)) \]

where \( \mu \) is the control parameter that must be in the range of 3.56 and 4 for chaotic behavior [51].

In this study, we utilize LM, and set \( \mu=4 \) to get the largest Lyapunov exponent, as a producer to quickly generate an initial vector for the vectorized CBC based diffusion process.

D. VECTOR OPERATION

A vector operation refers to the simultaneous application of an operation to a set of values. Thus, it allows to operate on aggregates of data without resorting to loops to handle individual scalar operations. Nowadays most CPUs combined with a modern programming language and single instruction multiple data extension (SIMD) can accelerate the program execution speed by implicit parallelization using vector operation.

Suppose \( i=1,2,\ldots,n \), \( V^1_i = [v_1^i(i)] \), \( V^2 = [v_2^2(i)] \) are two row vectors with length \( n \), \( V^3_i = [v_3^i(i)]^T \), \( V^4 = [v_4^2(i)]^T \) are two column vectors with height \( n \), and \( s \) is a scalar. We define function vec_opr() as a kind of operation between two vectors (or between a vector and a scalar) and function opr() as a kind of operation between two scalars, then the vector operations are described in Eq. (4). At present, SIMD has evolved from SSE 128 bits to AVX 512 bits [52] endowing vector operations with a more powerful parallel computing ability. Furthermore, vectorization is requisite to implement algorithms in parallel computing platforms, such as the graphics processing unit (GPU) and the field-programmable gate array (FPGA). As digital image data are stored in matrix form, many image processing algorithms have been vectorized to improve the operating efficiency [53], [54].


III. THE PROPOSED SECURE AND FAST IMAGE ENCRYPTION ALGORITHM

In this section, the proposed image encryption algorithm is detailed regarding a key matrix generation method, an initial vector generation method, and a four-round vectorized addition, modulus, and bitwise XOR (bitxor). It must be noted that, for the sake of simplification, although we use common operations in the following subsections to calculate vectors (or a vector and a scalar), they represent the vectorized operations.

\[
\{ \text{vec}\_\text{opr}(V_i^1, V_i^2) = \{\text{opr}(v_i^1(i), v_i^2(i))\} \}
\]

\[
\{ \text{vec}\_\text{opr}(V_i^1, s) = \{\text{opr}(v_i^1(i), s)\} \}
\]

\[
\{ \text{vec}\_\text{opr}(V_i^1, s) = \{\text{opr}(v_i^1(i), s)\}^T \}
\]

\[i = 1, 2, ..., n - 1, n\]

\[\text{vec\_opr}(\ , \}) = \{\text{opr}(\ , \)\}
\]

\[\text{vec\_opr}(\ , \}) = \{\text{opr}(\ , \)\}
\]

\[\text{vec\_opr}(\ , \}) = \{\text{opr}(\ , \)\}
\]

\[\text{vec\_opr}(\ , \}) = \{\text{opr}(\ , \)\}
\]

FIGURE 2. Flow diagram of the proposed encryption algorithm.

A. THE KEY MATRIX GENERATION METHOD

To obtain the key stream for encrypting an image, the real-valued chaotic sequence must be converted into an integer sequence. The general process to get the key stream is shown in Fig. 3. Thus the HCLS must perform at least \(WH/4\) iterations to obtain a key stream with enough length for encryption.

FIGURE 3. Flowchart of the general process to obtain the key stream.

However, the high complexity of the hyperchaotic system imposes a tradeoff between security and efficiency in a cryptosystem. By introducing a key schedule method to post-process the key stream, a \(WH\) key matrix can be quickly obtained, as shown in Fig. 4, with the HCLS only iterating \(2W\) times.
FIGURE 4. Flowchart of the post-process to obtain a key matrix.

Then, the Algorithm and descriptions of the key matrix generation method are detailed below:

**Step 1:** Input \(x_1(1), x_2(1), x_3(1), x_4(1)\), then use the fourth-order Runge-Kutta method to iterate the HCLS \(pre_1\) times to overcome the transient effect.

**Step 2:** Initialize an empty sequence \(B\) of length \(8W\).

**Step 3:** Continue to iterate HCLS \(2W\) times and fill new state values into \(B\).

**Step 4:** Transform the real-valued sequence \(B\) into an integer sequence \(I\).

**Step 5:** Input \(y(1)\), then iterate PWLCM \(pre_2\) times to overcome the transient effect.

**Step 6:** Initialize an empty sequence \(S\) of length \(2H\).

**Step 7:** Continue to iterate the PWLCM \(2H\) times and fill new state values into \(S\).

**Step 8:** Transform the real-values sequence \(S\) into integer sequence \(S'\).

**Step 9:** Split \(I\) into eight subsequences of length \(W\): \(I_1\)–\(I_8\).

**Step 10:** Initialize an empty matrix \(K\) with the size of \(W\times H\).

**Step 11:** Randomly choose \(temp_1\) from \(I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8\). Split \(I_1, I_2, I_3, I_4\) and \(temp_2\) from \(I_5, I_6, I_7, I_8\).

**Step 12:** Apply bitxor operation between \(temp_1\) and \(temp_2\), and fill the result into \(K\), then update \(I_n\) using cyclic right shift (circshift) operation.

**Step 13:** Repeat **Step 11** and **Step 12** \(H\) times to obtain key matrix \(K\).

### Algorithm

**Key matrix generation method**

**Input:** Initial values \(x_1(1), x_2(1), x_3(1), x_4(1)\), and \(pre_1\) for HCLS; initial values \(y(1)\), and \(pre_2\) for PWLCM

**Output:** The key matrix \(K\)

Put \(x_1(1), x_2(1), x_3(1), x_4(1)\) into HCLS

for \(i\) from 1 to \(pre_1\)

Iterate HCLS

Drop the state values \(x_1(i), x_2(i), x_3(i), x_4(i)\)

end for

\(B\leftarrow\text{zeros}(1, 8W)\)

\(x_1(1)\leftarrow x_1(pre_1+1); x_2(1)\leftarrow x_2(pre_1+1); x_3(1)\leftarrow x_3(pre_1+1)\)

Put new \(x_1(1), x_2(1), x_3(1), x_4(1)\) into HCLS

for \(i\) from 1 to \(2W\)

Iterate HCLS

end for

**end for**

**end case**

**case 5:** \(temp_p\leftarrow I_5\)

**case 6:** \(temp_p\leftarrow I_6\)

**case 7:** \(temp_p\leftarrow I_7\)

**case 8:** \(temp_p\leftarrow I_8\)

**end case**

\(K(x_{i,:})\leftarrow\text{bitxor}(temp_p, temp_p)\)

\(I_n\leftarrow\text{circshift}(temp_p, 1)\)

**end for**
B. THE INITIAL VECTOR GENERATION METHOD

Similar to most CBC based diffusion algorithms, an initial vector is required in the proposed encryption algorithm to start diffusion. However, the initial vector for the vectorized diffusion process is longer than that for existing algorithms, consequently increasing the storage and transmission costs.

As the LM can rapidly produce many random numbers from few initial values, we use it to generate the required initial vector as follows:

Step 1: Input $z(1)$, then iterate the LM $pre_3$ times to overcome the transient effect.

Step 2: Initialize an empty sequence $V$ with length of $W$.

Step 3: Continue to iterate the LM $W$ times and fill new state values into $V$.

Step 4: Transform real-valued sequence $V$ into the initial vector $IV_1$ consisting of integer numbers by

$$ IV_1 = \text{floor}(V \times 10^{15}) \mod 256 $$

(5)

C. THE FOUR-ROUND VECTORIZED DIFFUSION METHOD

Fig. 5 demonstrates how a 2×2 image is encrypted by the proposed four-round vectorized diffusion method, and the steps are detailed as follows.

Step 1: Input plain image $P$, key matrix $K$, and initial vectors $IV_1$ as shown in Fig. 5(a).

Step 2: Use $K$ for row-by-row encryption of $P$ from top to bottom, as defined in Eq. (6). Let encrypted row to influence the next row and $IV_1$ to encrypt the first row. After the first round of diffusion, temporary cipher image $TC_1$ is created, as illustrated in Fig. 5(b).

$$ \begin{align*}
TC_1(:,0) &= IV_1 \\
\text{temp} &= (P(:,i) + TC_1(:,i-1)) \mod 256 \\
TC_1(:,i) &= \text{bitxor}(K(:,i),\text{temp}) \\
i &= 1,2,...,H-1,H
\end{align*} $$

(6)

Step 3: Use $K$ for column-by-column encryption of $TC_1$ from left to right, as defined in Eq. (7). Let encrypted row to influence the next column and $IV_2=TC_1(:,W)$ encrypt the first column. After the second round of diffusion, temporary cipher image $TC_2$ is created, as illustrated in Fig. 5(c).

$$ \begin{align*}
TC_2(:,0) &= IV_2 = TC_1(:,W) \\
\text{temp} &= (TC_1(:,i) + TC_2(:,i-1)) \mod 256 \\
TC_2(:,i) &= \text{bitxor}(K(:,i),\text{temp}) \\
i &= 1,2,...,W-1,W
\end{align*} $$

(7)

Step 4: Use $K$ for row-by-row encryption of $TC_2$ from top to bottom, as defined in Eq. (8), and $IV_3=TC_2(H,:)$ to encrypt the first row. After the third round of diffusion, temporary cipher image $TC_3$ is created, as illustrated in Fig. 5(d).

$$ \begin{align*}
TC_3(:,0) &= IV_3 = TC_2(H,:) \\
\text{temp} &= (TC_2(:,i) + TC_3(:,i-1)) \mod 256 \\
TC_3(:,i) &= \text{bitxor}(K(:,i),\text{temp}) \\
i &= 1,2,...,H-1,H
\end{align*} $$

(8)

Step 5: Use $K$ for column-by-column encryption of $TC_3$ from left to right, as defined in Eq. (9), and $IV_4=TC_3(:,W)$ to encrypt the first column. After the fourth round of diffusion, temporary cipher image $TC_4$ is created, as illustrated in Fig. 5(e).

$$ \begin{align*}
TC_4(:,0) &= IV_4 = TC_3(:,W) \\
\text{temp} &= (TC_3(:,i) + TC_4(:,i-1)) \mod 256 \\
TC_4(:,i) &= \text{bitxor}(K(:,i),\text{temp}) \\
i &= 1,2,...,W-1,W
\end{align*} $$

(9)

Step 6: Output $TC_4$ as the final cipher image $C$ as shown in Fig. 5(f).
D. THE DECRYPTION PROCESS

The decryption process is the converse of the encryption process, as shown in the flow diagram in Fig. 6, where function vec_sub() represents a vectorized subtraction operation.

![Flow diagram of decryption.](image)

IV. EXPERIMENTAL RESULTS AND ANALYSES

We used common tests to thoroughly evaluate the security and performance of the proposed algorithm. The algorithm and tests were implemented in MathWorks Matlab 2016b, which provides all the vectorized operations for our algorithm, running on a personal computer equipped with an Intel® Core i5-6500 processor at 3.2 GHz, 16 GB memory.
and Windows 7 operating system. For the simulation, we randomly generated a session key consisting of $pre_1=157$, $x_1(1)=0.737$, $x_2(1)=0.962$, $x_3(1)=0.175$, $x_4(1)=0.504$ for HCLS, $pre_2=103$, $y(1)=0.357$ for PWLCM, and $pre_3=54$, $z(1)=0.985$ for LM. The images for testing had a resolution of 512×512 pixels and 8-bit grayscale level. Fig. 7 shows the simulation results of eight images (Lena, Man, Finger, Brain, Aerial, Baboon, Peppers, and Ruler) from different public image datasets.
The first column and second column of Fig. 7 show that the proposed algorithm can encrypt a meaningful image as an unordered image without leaking useful information. The third column of Fig. 7 shows that the encrypted images can be successfully recovered using the decryption algorithm and a correct session key. The simulation results demonstrate the feasibility of the proposed algorithm.

Below, we report the results of common test methods that verify the security and operational efficiency of the proposed image encryption algorithm.

A. HISTOGRAM ANALYSIS

The grayscale histogram shows the distribution of a digital image’s pixels. An ideal cipher image should have a uniformly distributed histogram with all the pixels being equiprobable to hide the pixel frequency of the plain image. Fig. 8 shows the histograms of the plain and cipher images. In the second column of Fig. 8, the histograms of the plain images have noticeable distribution characteristics. After applying the proposed encryption algorithm, the histograms of the corresponding cipher images are uniformly distributed, as shown in the fourth column of Fig. 8.

In addition, we calculate chi-square ($\chi^2$) values of the above images by Eq. (10) to further analyze the distributions of pixels.

$$\chi^2 = \sum_{i=1}^{L} \frac{(f_i - p_i)^2}{p_i}$$

and the pixel level $L$ of grayscale is 256, $f_i$ and $p_i$ are the observed frequency and expected frequency of each pixel value. For an ideal cipher image, the $\chi^2$ values should be 293.24783. Then, we set the significant level $\alpha=0.05$, and obtain the results of $P$-values as shown in Table 1. Since all the $P$-values are greater than 0.05, it indicates that any plain image encrypted by our algorithm has a good random distribution to resist frequency attacks.
FIGURE 8. Histograms of plain images (presented in the first column) in the second column, and cipher images (presented in the third column) in the fourth column.
B. STATISTICAL ANALYSIS

In this subsection, we use the NIST SP 800-22 test suite, which consists of 15 statistical tools to check the randomness of a sequence, to verify the statistical performance of the proposed algorithm. Both the key matrix used in the encryption and final cipher images are converted to binary sequences for the test. And the significant level $\alpha$ is set to 0.01, which means the result of each test must be greater than 0.01 to consider a sequence to be random with a confidence of 99%. In Table 2, it can be observed that all the $P$-values exceed the threshold of randomness statistical tests. Hence, we can conclude that the proposed image encryption algorithm has strong robustness against statistical attacks.

| Test                        | $K$   | Lena | Man | Finger | Brain | Aerial | Baboon | Peppers | Ruler |
|-----------------------------|-------|------|-----|--------|-------|--------|--------|---------|-------|
| Frequency                   | 0.9711| 0.7029| 0.2376| 0.9815 | 0.4187| 0.6915 | 0.6843 | 0.3118  | 0.2695 |
| Block Frequency             | 0.1942| 0.9831| 0.6823| 0.2127 | 0.4089| 0.5349 | 0.2138 | 0.8799  | 0.2592 |
| Cumulative Sums             | 0.7145| 0.1201| 0.1138| 0.8219 | 0.9864| 0.2204 | 0.3271 | 0.1355  | 0.4969 |
| Runs                        | 0.5754| 0.6712| 0.7373| 0.7622 | 0.7192| 0.1688 | 0.4896 | 0.8145  | 0.3382 |
| Longest Runs of Ones        | 0.1344| 0.1895| 0.9812| 0.4638 | 0.7093| 0.7847 | 0.2893 | 0.1865  | 0.6362 |
| Rank                        | 0.4729| 0.8548| 0.3274| 0.8699 | 0.3173| 0.3178 | 0.3118 | 0.5477  | 0.3916 |
| Spectral DFT                | 0.1477| 0.9313| 0.9402| 0.2989 | 0.7023| 0.6531 | 0.2059 | 0.7909  | 0.2916 |
| Nonperiodic Template Matchings | 0.6649| 0.1432| 0.9079| 0.1461| 0.7089| 0.4839 | 0.2718 | 0.7892  | 0.5851 |
| Overlapping Template Matchings | 0.2683| 0.5123| 0.7959| 0.2943| 0.5043| 0.4383 | 0.2465 | 0.205   | 0.5973 |
| Universal Statistical       | 0.7701| 0.8001| 0.7724| 0.9001| 0.9015| 0.2389 | 0.8269 | 0.9812  | 0.1802 |
| Approximate Entropy         | 0.3159| 0.5933| 0.2189| 0.9082| 0.3694| 0.8861 | 0.4809 | 0.8713  | 0.8352 |
| Random Excursions           | 0.9753| 0.0773| 0.8636| 0.7862| 0.5803| 0.3869 | 0.2617 | 0.7301  | 0.2606 |
| Random Excursions Variant   | 0.6124| 0.7555| 0.3385| 0.4897| 0.4432| 0.2465 | 0.7755 | 0.4274  | 0.1541 |
| Serial                      | 0.5465| 0.3276| 0.3721| 0.5498| 0.7791| 0.1867 | 0.8719 | 0.3479  | 0.1711 |
| Linear Complexity           | 0.4811| 0.6048| 0.5412| 0.4927| 0.2917| 0.7556 | 0.7015 | 0.5102  | 0.9563 |

C. KEY SPACE ANALYSIS

Owing to the rapid development of computing devices, exhaustive attacks threaten symmetric encryption algorithms. The key space of a secure encryption algorithm must be larger than $2^{128}$ to resist brute-force attacks [38], [39]. In our algorithm, the session key has two parts: 1) pre-iterations of the chaotic systems with 16-bit unsigned integers, and 2) initial values of the chaotic systems with double-precision floating-point numbers. Thus, the key space of the proposed algorithm is $(2^{16})^3 \times (10^{15})^6 \approx 2^{445}$. Table 3 lists the comparison of key spaces between various image encryption algorithms. Although a larger key space provides higher resistance against brute-force attacks, it also means harder transmission. Since NIST declares the key space more than 256 is enough for most encryption scenarios [12], our algorithm can achieve the same protective effect against brute force attacks as some algorithms with larger key space.

Table 3. Key space of various image encryption algorithms.

| Algorithm          | Ours | Ref. [38] | Ref. [39] | Ref. [40] | Ref. [41] | Ref. [42] | Ref. [43] | Ref. [44] | Ref. [45] |
|--------------------|------|------------|------------|------------|------------|------------|------------|------------|------------|
| Key space          | $2^{445}$ | $2^{297}$ | $2^{399}$ | $2^{216}$ | $2^{256}$ | $2^{465}$ | $2^{212}$ | $2^{812}$ | $2^{418}$ |

D. PLAINTEXT SENSITIVITY ANALYSIS

To resist a differential attack, a secure encryption algorithm must be sensitive to minor changes in the plain image. In other words, even one bit change in any position of a plain image should make the encryption generate a totally different cipher image. There are two common standards given by Eq. (11), NPCR (number of pixel change rate), and UACI (uniform average changing intensity), to measure the sensitivity to one bit change in plain images.

$$\text{NPCR} = \frac{\sum_{i=1}^{W} \sum_{j=1}^{H} D(i, j)}{W \times H} \times 100\%$$

$$\text{UACI} = \frac{1}{255 \times W \times H} \times \left[ \sum_{i=1}^{W} \sum_{j=1}^{H} |C_1(i, j) - C_2(i, j)| \right] \times 100\%$$

where $C_1$ and $C_2$ are cipher images.

We evaluated the ability of the proposed encryption algorithm to resist a differential attack as follows. First, we...
changed one bit of a randomly selected pixel in the original plain image. Then, we used the same session key to encrypt both the original and modified plain images. Thus, we obtained two cipher images, C1 and C2. Finally, we used Eq. (11) to calculate NPCR and UACI between C1 and C2. We evaluated all the test images 100 times and obtained the average NPCR and UACI listed in Table 4. Our algorithm can effectively resist a differential attack because the experimental results are close to the expected values: NPCR = 99.6094% and UACI = 33.4636% [1], [2]. Furthermore, the results of the other algorithms indicate the superior performance of the proposed algorithm. Thus, the proposed encryption algorithm can provide higher security for image communication under strong differential attacks.

### Table 4. Average NPCR and UACI obtained from the plaintext image sensitivity test (%).

| Image  | Standard | Ours | Ref. [38] | Ref. [39] | Ref. [40] | Ref. [41] | Ref. [42] | Ref. [43] | Ref. [44] | Ref. [45] |
|--------|----------|------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| Lena   | 99.6091  | 99.6122 | 99.6093  | 99.6114  | 99.6106  | 99.81     | 99.61     | 99.6185  | 99.63     |
| UACI   | 33.4709  | 33.4573 | 33.5989  | 33.4523  | 33.4669  | 33.40     | 33.43     | 33.4561  | 28.5142  |
| Man    | 99.6112  | 99.6092 | 99.6089  | 99.6081  | -         | -         | 99.61     | 99.6078  | -         |
| UACI   | 33.4638  | 33.4682 | 33.4713  | 33.4662  | -         | -         | -         | 33.4821  | -         |
| Finger | 99.6083  | 99.6118 | -        | -         | -         | -         | -         | 99.6170  | -         |
| UACI   | 33.4705  | 33.4691 | -        | -         | -         | -         | -         | 33.4162  | -         |
| Brain  | 99.6125  | -      | -        | -         | -         | -         | -         | 99.6185  | -         |
| UACI   | 33.4655  | -      | -        | -         | -         | -         | -         | 33.4550  | -         |
| Aerial | 99.6089  | -      | 99.6043  | 99.6091  | -         | 99.61     | 99.6063  | -         | -         |
| UACI   | 66.4643  | -      | 33.2875  | 33.4826  | -         | 33.46     | 33.4969  | -         | -         |
| Baboon | 99.6105  | 99.6108 | 99.6091  | 99.6016  | 99.6097  | -         | 99.61     | 99.6109  | 99.63     |
| UACI   | 33.4661  | 33.4653 | 33.5381  | 33.4039  | 33.4660  | -         | 33.48     | 33.5034  | 27.0542  |
| Peppers| 99.6078  | 99.6102 | 99.6202  | 99.6115  | -         | -         | 99.61     | 99.6044  | 99.62     |
| UACI   | 33.4637  | 33.4691 | 33.6602  | 33.4245  | -         | -         | 33.47     | 33.4117  | 29.4536  |
| Ruler  | 99.6096  | -      | -        | -         | -         | -         | -         | 99.6239  | -         |
| UACI   | 33.4635  | -      | -        | -         | -         | -         | -         | 33.52     | 33.4720  |

### Table 5. Average NPCR and UACI obtained from the encryption key sensitivity test (%).

| Value name | Original value | Modified value | Lena | Peppers | Baboon | Lena | Peppers | Baboon |
|------------|----------------|----------------|------|---------|--------|------|---------|--------|
| pre1       | 157            | 158            | 99.6048 | 99.6158 | 99.6008 | 33.4550 | 33.4742 | 33.4599 |
| x1(1)      | 0.737          | 0.737+10^{-15} | 99.6156 | 99.6032 | 99.6011 | 33.4628 | 33.4588 | 33.4569 |
| x2(1)      | 0.962          | 0.962+10^{-15} | 99.6096 | 99.6115 | 99.6066 | 33.4674 | 33.4596 | 33.4738 |
| x3(1)      | 0.175          | 0.175+10^{-15} | 99.6149 | 99.6123 | 99.6014 | 33.4568 | 33.4736 | 33.4508 |
| x4(1)      | 0.504          | 0.504+10^{-15} | 99.6022 | 99.6124 | 99.6011 | 33.4704 | 33.4749 | 33.4752 |
| pre2       | 103            | 104            | 99.6069 | 99.6128 | 99.6037 | 33.4763 | 33.4577 | 33.4789 |
| y1(1)      | 0.357          | 0.357+10^{-15} | 99.6037 | 99.6080 | 99.6036 | 33.4596 | 33.4853 | 33.4776 |
| pre3       | 54             | 55             | 99.6055 | 99.6086 | 99.6037 | 33.4534 | 33.4761 | 33.4776 |
| z(1)       | 0.985          | 0.985+10^{-15} | 99.6164 | 99.6166 | 99.6131 | 33.4728 | 33.4620 | 33.4694 |

In the second stage, we used the abovementioned session keys to encrypt the decryption key sensitivity of the proposed algorithm. Unlike the previous test, we used the original and modified session keys to decrypt the same cipher image, and then we obtained a correctly decrypted image and an incorrectly decrypted image, as illustrated for the Lena image in Fig. 9. Fig. 9(a) and (b) show that even by only adding 10^{-15} to x1, the cipher image fails to be correctly decrypted and results in salt-and-pepper noise. Only using the exact decryption key allows the cipher image to be recovered as the original plain image, as shown in Fig. 9(c) and (d).
F. ANALYSIS OF THE CORRELATION BETWEEN ADJACENT PIXELS

A high correlation between adjacent pixels hinders the effective hiding of plain image information. Hence, a secure encryption algorithm should minimize the correlation between adjacent pixels, with a low correlation between adjacent pixels in cipher images indicating the high security of an encryption algorithm.

For the pixel correlation test, we randomly selected 10,000 pairs of adjacent pixels along with the horizontal, vertical, and diagonal directions from the plain image of Lena and its corresponding cipher image. Fig. 10 shows the correlation distribution. In Fig. 10(b)–(d), the distributions of adjacent pixels on the plain images are highly concentrated. In contrast, the distributions of adjacent pixels in the cipher images are randomly distributed, as shown in Fig. 10(f)–(h). Therefore, the proposed encryption algorithm can substantially reduce the correlation between adjacent pixels for increased security.

TABLE 6. Average NPCR and UACI obtained from the decryption key sensitivity test (%).

| Value name | Original value | Modified value | NPCR Lena | NPCR Peppers | NPCR Baboon | UACI Lena | UACI Peppers | UACI Baboon |
|------------|----------------|----------------|-----------|--------------|-------------|-----------|--------------|-------------|
| pre1       | 157            | 158            | 99.6058   | 99.6009      | 99.6153     | 33.4554   | 33.4526      | 33.4518     |
| x(1)       | 0.737          | 0.737×10⁻⁴⁵    | 99.6025   | 99.6261      | 99.5975     | 33.4732   | 33.4709      | 33.4619     |
| x(1)       | 0.962          | 0.962×10⁻⁴⁵    | 99.6088   | 99.6174      | 99.6270     | 33.4714   | 33.4588      | 33.4592     |
| x(1)       | 0.175          | 0.175×10⁻⁴⁵    | 99.6093   | 99.6074      | 99.6038     | 33.4615   | 33.4654      | 33.4617     |
| x(1)       | 0.504          | 0.504×10⁻⁴⁵    | 99.6086   | 99.6152      | 99.6059     | 33.4501   | 33.4653      | 33.4550     |
| pre2       | 103            | 104            | 99.6161   | 99.6202      | 99.5936     | 33.4583   | 33.4737      | 33.4521     |
| y(1)       | 0.357          | 0.357×10⁻⁴⁵    | 99.6091   | 99.6142      | 99.6245     | 33.4741   | 33.4701      | 33.4677     |
| pre3       | 54             | 55             | 99.5943   | 99.6252      | 99.6232     | 33.4665   | 33.4656      | 33.4693     |
| z(1)       | 0.985          | 0.985×10⁻⁴⁵    | 99.5909   | 99.5989      | 99.6141     | 33.4592   | 33.4699      | 33.4709     |
We further used correlation coefficient $r_{xy}$ to measure the robustness of the proposed image encryption algorithm against statistical attacks given by

$$
\begin{align*}
\bar{x} &= \frac{1}{N} \sum_{i=1}^{N} x_i, \quad \bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i \\
 r_{xy} &= \frac{\sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{N} (y_i - \bar{y})^2}}
\end{align*}
$$

(12)

where $N = 10,000$ and $x_i$ and $y_i$ ($i = 1, 2, \ldots, N$) are pairs of adjacent pixels randomly selected from the image. The value of $r_{xy}$ in all the plain images is close to one along each direction, and in all the cipher images is close to zero along each direction.

**TABLE 7. Results of correlation coefficients between adjacent pixels.**

| Image   | Horizontal | Vertical | Diagonal |
|---------|------------|----------|----------|
| Lena    | 0.9841     | 0.9691   | 0.91639  |
| Cipher  | -0.0002    | -0.0007  | -0.0016  |
| Man     | 0.9675     | 0.9593   | 0.9432   |
| Ciper   | -0.0015    | 0.0024   | 0.0004   |
| Finger  | 0.9037     | 0.8869   | 0.8119   |
| Ciper   | -0.0013    | -0.0001  | -0.0007  |
| Brain   | 0.9583     | 0.9467   | 0.9185   |
| Ciper   | 0.0009     | 0.0010   | -0.0008  |
| Aerial  | 0.8549     | 0.8993   | 0.8003   |
| Ciper   | -0.0006    | -0.0005  | -0.0007  |
| Baboon  | 0.9100     | 0.9322   | 0.8647   |
| Ciper   | 0.0001     | 0.0003   | 0.0011   |
| Peppers | 0.9764     | 0.9733   | 0.9651   |
| Ciper   | -0.0009    | 0.0017   | -0.0001  |
| Ruler   | 0.4599     | 0.4494   | -0.0291  |
| Ciper   | 0.0003     | -0.0004  | -0.0008  |

**G. INFORMATION ENTROPY ANALYSIS**

Information entropy is an important measure of randomness on an image. A higher entropy indicates more information and more randomness, and the entropy reaches its maximum only if the probabilities of all the information elements are equal. The information entropy is defined as

$$
H(g) = -\sum_{j=0}^{L} p(g_j) \log_2 p(g_j)
$$

(13)

where $L$ denotes the number of possible elements $g_i$ and $p(g_i)$ represents the frequency of element $g_i$ with the entropy of an ideal random 8-bit grayscale image being 8. Table 9 lists the information entropy values for various encryption algorithms. The information entropy of the cipher images is close to eight, indicating that the proposed algorithm can resist entropy attacks. Compared with existing algorithms, our proposal has made some progress regarding information entropy.
TABLE 9. Information entropy of cipher images obtained using various encryption algorithms.

| Image   | Plaintext | Ours  | Ref. [38] | Ref. [39] | Ref. [40] | Ref. [41] | Ref. [42] | Ref. [43] | Ref. [44] | Ref. [45] |
|---------|-----------|-------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| Lena    | 7.4451    | 7.9996 | 7.9973    | 7.9992    | 7.9972    | 7.9977    | 7.9987    | 7.9971    | 7.9939    | 7.9881    |
| Man     | 7.2367    | 7.9998 | 7.9998    | -         | 7.9998    | 7.9998    | -         | -         | -         | 7.9890    |
| Finger  | 7.0153    | 7.9995 | 7.9975    | -         | -         | -         | -         | -         | -         | -         |
| Brain   | 3.1047    | 7.9996 | -         | -         | -         | -         | -         | 7.9967    | -         | -         |
| Aerial  | 6.9939    | 7.9997 | -         | -         | 7.9998    | -         | -         | 7.9972    | 7.9972    | -         |
| Baboon  | 7.2925    | 7.9999 | 7.9998    | 7.9993    | 7.9998    | 7.9998    | -         | 7.9970    | -         | 7.9866    |
| Peppers | 7.5937    | 7.9996 | 7.9993    | 7.9970    | 7.9992    | 7.9992    | 7.9979    | 7.9972    | 7.9973    | 7.9894    |
| Ruler   | 0.5000    | 7.9996 | -         | -         | -         | -         | -         | 7.9974    | -         | -         |

H. CHOSEN-PLAINTEXT ATTACK ANALYSIS

A secure encryption algorithm must resist chosen-plaintext attacks by preventing an attacker from distinguishing two encryptions even if the attacker can elaborate choose the corresponding plain images. An all-black or all-white image is often used to analyze the vulnerability of encryption algorithms [1], [2], [40], [44]. The proposed algorithm uses four-round vectorized diffusion to ensure that any pixel information spreads to the entire cipher image, thus having a high sensitivity to any plain image. Fig. 11 shows the histogram of all-black and all-white images of 512 x 512 pixels, and Table 10 lists the experimental measures. In Fig. 11, the pixel distributions of the all-black and all-white images are uniform after encryption. Thus, an attacker cannot obtain useful information from the cipher images to break the encryption algorithm. More importantly, the plaintext sensitivity, entropy, and correlation results listed in Table 8 indicate that the proposed algorithm provides an excellent performance regarding the corresponding measures, even when using these specific plain images. Therefore, the proposed image encryption algorithm can resist chosen-plaintext attacks.

FIGURE 11. Evaluation of all-black and all-white image encryption. (a) All-white image and its (b) cipher image. (c) Histogram of all-white cipher image. (d) All-black image and its (e) cipher image. (f) Histogram of all-black cipher image.

TABLE 10. Experimental results for all-black and all-white images.

| Test        | \( \chi^2 \) test | NPCR(%) | UACI(%) | Entropy | Correlation coefficient |
|-------------|---------------------|---------|---------|---------|------------------------|
|             |                     | Min Max | Min Max |         | Horizontal Vertical Diagonal |
| All-black   | 0.2342              | 99.6018 | 99.6227 | 33.4559 | 33.4644 | 7.9991 | 0.0014 | -0.0007 | 0.0033 |
| Ref. [40]   | -                   | 99.6328 | 33.4965 | 7.9975  | -0.0049 | 0.0087 | -0.0006 |
| Ref. [44]   | Pass                | 99.6143 | 33.4719 | 7.9976  | -0.0038 | 0.0016 | 0.0028 |
| Ours        | 0.3508              | 99.5993 | 99.6244 | 33.4528 | 33.4684 | 7.9994 | 0.0313 | 0.0214 | 0.0083 |
| All-white   | Ref. [40]           | -       | 99.6129 | 33.5079 | 7.9974  | 0.0033 | -0.0051 | 0.0093 |
| Ref. [44]   | Pass                | 99.6075 | 33.4302 | 7.9973  |         |         |         |

I. ROBUSTNESS ANALYSIS

Noise jamming and data loss may occur when a digital image is transmitted through an unstable public channel. Suitable image encryption should be robust to noise and data loss by...
being able to decrypt a noisy cipher image into a recognizable plain image. Fig. 12 shows the results of noise attacks applied to the proposed image encryption algorithm. The cipher image can be decrypted even when it is polluted by different types and intensities of noise, and the recovered Baboon image is recognizable. Fig. 13 shows the decrypted results of the cipher image with data lost 1/16, 1/8, 1/4, and 1/2. Although the cipher image can be decrypted to a meaningful image, we observe that the noise mainly concentrates somewhere. It means in some special situations, such as satellite image encryption and medical image encryption, the receiver has a certain probability of missing some key information. Alternatively, a solution to this problem is adding a scrambling process before or after the diffusion process, but the time consumption may cost even more than the diffusion process [43]. Therefore, our algorithm can recover a useful image from data loss in most real-time communication scenarios.

Furthermore, decrypted images with pollution and data loss are evaluated by the PSNR (Peak Signal to Noise Ratio).

$$\text{PSNR} = 10\log \left( \frac{255 \times 255}{(1/W \times H) \sum_{i=1}^{W} \sum_{j=1}^{H} (P(i, j) - D(i, j))^2} \right)$$  \hspace{1cm} (14)

In Table 11, the PSNRs of images decrypted from cipher images with different pollution and data loss are presented. By comparison with previous works, our algorithm still has good performance despite the above-mentioned weakness. It indicates that the proposed image encryption algorithm has strong robustness to protect data transmission on the Internet.

FIGURE 12. Decrypted images from cipher images polluted by Gaussian white noise with zero mean and variances of (a) 0.001, (b) 0.003, and (c) 0.005 and by (d) 1%, (e) 3%, and (f) 5% salt-and-pepper noise.

FIGURE 13. Data loss decryption results. (a) 1/16 cropping image and its decrypted image, (b) 1/8 cropping image and its decrypted image, (c) 1/4 cropping image and its decrypted image, and (d) 1/2 cropping image and its decrypted image.
Table 11. PSNR results of decrypted images.

| Algorithm | Cipher image polluted by | Gaussian with variance of 0.001 | Gaussian with variance of 0.003 | Gaussian with variance of 0.005 | Data loss of 1/16 | Data loss of 1/8 | Data loss of 1/4 | Data loss of 1/2 |
|-----------|--------------------------|-------------------------------|-------------------------------|-------------------------------|------------------|----------------|----------------|----------------|
| Ours      | 20.46                    | 13.29                         | 18.21                         | 12.73                         |                  |                |                |                |
| Ref. [44] | 30.59                    | 19.34                         | 15.21                         | 12.23                         |                  |                |                |                |

Table 12. Execution time of various image encryption algorithms (Unit: s).

| Algorithm | Image Size       | 256x256 | 512x512 | 1024x1024 |
|-----------|------------------|---------|---------|-----------|
| Ours      | Worst (s)        | 0.018   | 0.031   | 0.077     |
|           | Best (s)         | 0.011   | 0.023   | 0.061     |
|           | Average (s)      | 0.015   | 0.027   | 0.068     |
|           | Throughput (Mbps)| 1.3     | 9.5     | 14.7      |
| Ref. [38] |                  | 1.217   | 4.893   | 19.891    |
| Ref. [39] |                  | 25kbps  | 210kbps |           |
| Ref. [41] |                  | 0.053   | 0.158   | 0.806     |
| Ref. [43] |                  | 0.133   | -       | -         |
| Ref. [44] |                  | 0.380   | 1.431   | 5.719     |

V. PERFORMANCE ANALYSIS OF COLOR IMAGES

In this section, the proposed algorithm is evaluated by security tested on color images.

A. SIMULATION AND HISTOGRAM RESULTS

The color image consists of pixels in the R channel, G channel, and B channel. Since it is more expressive than the grayscale image, it has more extensive application prosperity. The simulation and histogram results of the Lean color image are shown in Fig. 14. The encryption and decryption results are respectively shown in the third row and fifth row of Fig. 14. And by comparison with the second row and fourth row of Fig. 14, we can conclude that the distributions of pixels in all channels have been hidden after being encrypted by the proposed algorithm. Hence, the results indicate that our algorithm is also suitable to deal with color images.
FIGURE 14. Simulation and histogram results of Lena color image. The first row shows the Lena color image and its R, G, B channels. The second row shows the distribution of pixels of Lena color image in R, G, B channels. The third row shows the cipher image and its R, G, B channels. The fourth row shows the distribution of pixels of cipher image in R, G, B channels. The fifth row shows the decrypted image and its R, G, B channels. The sixth row shows the distribution of pixels of the decrypted image and its R, G, B channels.

B. OTHER ANALYSES

In this part, we utilize $\gamma_{xy}$, $\chi^2$, NPCR, UACI, and information entropy tests to evaluate the security performance of our algorithm on color images, and obtain the time consumption of color image encryption (execution time and throughput are 0.063s and 12.6 Mbps). Table 13 shows the results of all the tests, and also demonstrates the comparisons with the results of other algorithms. Since the results are all close to ideal values and are superior to the previous algorithms, we can safely conclude that the proposed image encryption algorithm also has good security and speed performance for color images, not just for grayscale images.
VI. CONCLUSIONS

We proposed a secure and fast image encryption algorithm using hyper-chaos based key generator and vector operation. Aiming to reduce the time consumption of key stream generation, we present a novel method to post-process key stream generated from the general method. As a result, the HCLS only iterates 2W times rather than W×H times to create a key matrix with the same size as the plain image (W and H are the width and height of the plain image respectively). During the key matrix generation process, a PWLCM is employed as a random selector to control the bitwise XOR operation between the quantified hyperchaotic sequences, which can further enhance the randomness and unpredictability of the key matrix. Then, considering two-dimensional images, we introduce the vector operation into the CBC based diffusion process, so the plain image can be implicit parallel encrypted row by row and column by column with high efficiency. Furthermore, by using an initial vector as a chain between two diffusion rounds, any pixel information and key information can be diffused to the whole cipher image. Thus, even a minor change of the session key or the plain image completely changes the resulting cipher image. In addition, we apply an LM to fast generate an initial vector for the first diffusion round, which can facilitate the storage and transmission of the initial vector. Experimental results show that the proposed encryption algorithm has a large key space of 2^{345}, high plaintext sensitivity with NPCR near 99.6094% and UACI near 33.4636%, and high session key sensitivity to resist statistical, brute-force, differential, and some other common attacks. Besides, it achieves high performance regarding information entropy>7.999, and noise and data loss robustness with PSNR>10db. Moreover, tests on all-black and all-white images verify the chosen-plaintext attack resistance of our algorithm. In addition, the positive results of tests on color images indicate that our algorithm has a wide application perspective. Finally, the time complexity of our algorithm is in the order of O(W+H), which supports the algorithm to encrypt a 512×512 image in 0.03s. Overall, despite some respects, such as key space and data loss resistance, show slightly insufficently, the proposed image encryption algorithm still meets the security, efficiency, and robustness requirements for most daily image confidential communications.

REFERENCES

[1] M. Khan, and T. Shah, “A literature review on image encryption techniques,” 3D Res., vol. 5, no. 4, pp. 5-29, 2014.
[2] M. Kaur, and V. A. Kumar, “Comprehensive review on image encryption techniques,” Arch. Comput. Method E., vol. 27, no. 1, pp. 15-43, 2020.
[3] X. Di, X. Liao, and P. Wei, “Analysis and improvement of a chaos-based image encryption algorithm,” Chaos Soliton. Fract., vol. 40, no. 5, pp. 2191-2199, 2009.
[4] M. Kaur, D. Singh, and R.S. Uppal, “Parallel strength Pareto evolutionary algorithm-II based image encryption,” IET. Image. Process., vol. 14, no. 6, pp. 1015-1026, 2020.
[5] H.M. Ghadirli, A. Nodehi, and R. Enayatifar, “An overview of encryption algorithms in color images,” Signal Process., vol. 164, pp. 163-185, 2019.
[6] N. Rawat et al., “Compressive sensing based robust multispectral double-image encryption,” Appl. Optics, vol. 54, no. 7, pp. 1782-1793, 2015.
[7] E.E. García-Guerrero et al., “Randomness improvement of chaotic maps for image encryption in a wireless communication scheme using PIC-microcontroller via Zigbee channels,” Chaos Soliton. Fract., vol. 133, 2020, Art. no. 107316.
[8] A. Girdhar, H. Kapur, and V. Kumar, “A novel grayscale image encryption approach based on chaotic maps and image blocks,” Appl. Phys. B-Lasers O., vol. 127, no. 3, pp. 1-12, 2021.
[9] E.ED. Hemdan, “An efficient and robust watermarking approach based on single value decomposition, multi-level DWT, and wavelet fusion with scrambled medical images,” Multimed. Tools Appl., vol. 80, pp. 1749-1777, 2021.
[10] M. Nazari, and M. Mehrabian, “A novel chaotic IWT-LSB blind watermarking approach with flexible capacity for secure transmission of authenticated medical images,” Multimed. Tools Appl., vol. 80, pp. 10615-10655, 2021.
[11] Data Encryption Standard (DES), FIPS Standard 46-3, 1999.
[12] Advanced Encryption Standard (AES), FIPS Standard 197, 2001.
[13] C.L. Chowdhary et al., “Analytical study of hybrid techniques for image encryption and decryption,” Sensors-basel., vol. 20, no. 18, 2020, Art. no. 5162.
[14] Y. Bentoutoua et al., “An improved image encryption algorithm for satellite applications,” Adv. Space Res., vol. 66, no. 1, pp.176-192, 2020.

Table 13. The results of security and encryption speed tests.

| Test | Horizontal | Vertical | Diagonal | $\chi^2$ | NPCR | UACI | Information entropy |
|------|------------|----------|----------|---------|-------|------|-------------------|
| R    | Ours       | -0.0016  | -0.0003  | -0.0012 | 0.3409 | 99.6099% | 33.4605% | 7.9992 |
| Ref. [44] | -        |          |          | Pass    | 99.6197% | 33.4423% | 7.9993 |
| G    | Ours       | -0.0005  | 0.0011   | 0.0028  | 0.9911  | 99.6188% | 33.4599% | 7.9994 |
| Ref. [44] | -        |          |          | Pass    | 99.6117% | 33.4539% | 7.9994 |
| B    | Ours       | -0.0017  | -0.0024  | -0.0004 | 0.9392  | 99.6085% | 33.4621% | 7.9993 |
| Ref. [44] | -        |          |          | Pass    | 99.6113% | 33.4647% | 7.9992 |

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3118377, IEEE Access
[15] Y.L. Luo et al., “An image encryption method based on elliptic curve elgamal encryption and chaotic systems,” IEEE Access., vol. 7, pp. 38507-38522, 2019.

[16] B. Deepan et al., “Multiple-image encryption by space multiplexing based on compressive sensing and the double-random phase-encoding technique,” Appl. Optics., vol. 53, no. 20, pp. 4539-4547, 2014.

[17] R. Enayatifar, A.H. Abdullah, and I.F. Isnin, “Chaos-based image encryption using a hybrid genetic algorithm and a DNA sequence,” Opt. Laser Eng., vol. 56, pp. 83-93, 2014.

[18] M. Kaur, V. Kumar, “Beta Chaotic Map Based Image Encryption Using Genetic Algorithm,” Int. J. Bifurcat. Chaos., vol. 28, no. 11, 516-519, 2018, Art. no. 1850132.

[19] Q. Xu et al., “A fast image encryption algorithm based on compressive sensing and hyperchaotic map,” Opt. Laser Eng., vol. 121, pp. 203-214, 2019.

[20] R. Matthews, “On the derivation of a ‘chaotic’ encryption algorithm,” Cryptologia., vol. 13, no. 1, pp. 29-42, 1989.

[21] L.M. Pecora, and T.L. Carroll, “Synchronization in chaotic systems,” Phys. Rev. Lett., vol. 64, no. 8, pp. 821-824, 1990.

[22] J. Fridrich, “Symmetric ciphers based on two-dimensional chaotic maps,” Int. J. Bifurcat. Chaos., vol. 8, no. 6, pp. 1259-1284, 1998.

[23] C. Pak, and L. Huang, “A new color image encryption using combination of the 1D chaotic map,” Signal Process., vol. 138, pp. 129-137, 2017.

[24] H. Wang et al., “Cryptanalysis and enhancements of image encryption using combination of the 1D chaotic map,” Signal Process., vol. 144, pp. 444-452, 2018.

[25] A. Hasheminejad, and M.J. Rostami, “A novel bit level multiphase algorithm for image encryption based on PWLCM chaotic map,” Optik., vol. 184, pp. 205-213, 2019.

[26] S.J. Sheela, K.V. Suresh, and D. Tandur, “Image encryption based on modified Henon map using hybrid chaotic shift transform,” Multimed. Tools Appl., vol. 77, no. 19, pp. 25223-25251, 2018.

[27] M. Ahmad et al., “Cryptanalysis of an image encryption algorithm based on PWLCM and inertial delayed neural network,” J. Intell. Fuzzy Syst., vol. 34, no. 3, pp. 1323-1332, 2018.

[28] H. Liu, A. Kadir, and J. Liu, “Color pathological image encryption algorithm using arithmetic over Galois field and coupled hyper chaotic system,” Opt. Laser Eng., vol. 122, pp. 123-133, 2019.

[29] T. Gao et al., “The generation and circuit implementation of a new hyper-chaos based upon Lorenz system,” Phys. Lett. A., vol. 361, no. 1-2, pp. 78-86, 2007.

[30] T.M. Khlebodarova et al. “Chaos and hyperchaos in simple gene network with negative feedback and time delays,” J Bioinf. Comput. Biol., vol. 15, no. 2, 2016, Art. no. 1650042.

[31] V. Basios, and C.G. Antonopoulos, “Hyperchaos & labyrinth chaos: Revisiting Thomas–Rössler systems,” J. Theor. Biol., vol. 460, pp. 153-159, 2019.

[32] B. Nawe et al., “Image encryption using a combination of Grain-128a algorithm and Zaslavsky chaotic map,” IET Image Process., vol. 14, no. 6, pp. 1120-1131, 2020.

[33] T. Gao, and Z. Chen, “A new image encryption algorithm based on hyper-chaos,” Phys. Lett. A., vol. 372, no. 4, pp. 394-400, 2008.

[34] R. Rhouma, and S. Belghith, “Cryptanalysis of a new image encryption algorithm based on hyper-chaos,” Phys. Lett. A., vol. 372, no. 38, pp. 5973-5978, 2008.

[35] Z. Li et al., “A novel plaintext-related image encryption algorithm using hyper-chaotic system,” Nonlinear Dynam., vol. 94, no. 2, pp. 1319-1333, 2018.

[36] X. L. Chai, Z. H. Gan, and M. H. Zhang, “A fast chaos-based image encryption scheme with a novel plain image-related swapping block permutation and block diffusion,” Multimed. Tools Appl., vol. 76, no. 14, pp. 15561–15585, 2017.

[37] L. Liu, Y. Lei, and D. Wang, “A fast chaotic image encryption algorithm with simultaneous permutation-diffusion operation,” IEEE Access., vol. 8, pp. 27361–27374, 2020.

[38] H.M. Yuan et al., “A new parallel image cryptosystem based on 5D hyper-chaotic system,” Signal Process.-Image., vol. 52, pp. 87-96, 2017.

[39] Ü. Çağuşoğlu, and S. Kaçar, “A novel parallel image encryption algorithm based on chaos,” Cluster Comput., vol. 22, no. 4, pp. 1211–1223, 2019.

[40] M. Zhou, and C. Wang, “A novel image encryption scheme based on conservative hyperchaotic system and closed-loop diffusion between blocks,” Signal Process., to be published. DOI: 10.1016/j.sigpro.2020.107484.

[41] C.F. Zhao, and H.P. Ren, “Image encryption based on hyper-chaotic multi-attractors,” Nonlinear Dynam., vol. 100, no. 1, pp. 679-698, 2020.

[42] L. Oteko Tresor and M. Sumbwanyambe, “A Selective Integration Image Encryption Scheme Based on 2D DWT, Henon Map and 4D Qi Hyper-Chaos,” IEEE Access, vol. 7, pp. 103463-103472, 2019.

[43] X. Gao, “Image encryption algorithm based on 2D hyperchaotic map,” Opt. Laser Eng., vol. 142, 2021, Art. no. 107252.

[44] M.D. Gupta, and R. K. Chauhan, “Secure image encryption scheme using 4D-Hyperchaotic systems based reconfigurable pseudo-random number generator and S-Box,” Integration, vol. 81, pp. 137-159, 2021.

[45] X. Wang, and M. Zhao, “An image encryption algorithm based on hyperchaotic system and DNA coding,” Opt. Laser Eng., vol. 143, 2021, Art. no. 107316.

[46] X. Wang et al., “A one-time pad color image cryptosystem based on SHA-3 and multiple chaotic systems,” Opt. Laser Eng., vol. 103, pp. 1-8, 2018.
[47] S. Rajendran et al., “Fast pre-processing hex Chaos triggered color image cryptosystem,” Multimed. Tools Appl., vol. 79, no. 7, pp. 1-23, 2020.

[48] S. Zhou et al., “A novel method based on the pseudo-orbits to calculate the largest Lyapunov exponent from chaotic equations,” Chaos., vol. 29, 2019, Art. no. 033125.

[49] M. Ghebleh, A. Kanso, and D. Stevanović, “A novel image encryption algorithm based on piecewise linear chaotic maps and least squares approximation,” Multimed. Tools Appl., vol. 77, no. 6, pp. 7305-7326, 2018.

[50] M.J. Rostami et al. “A novel parallel image encryption with chaotic windows based on logistic map,” Comput. Electr Eng., vol. 62, pp. 384-400, 2017.

[51] S. Zhou, and X. Wang, “Identifying the linear region based on machine learning to calculate the largest Lyapunov exponent from chaotic time series,” Chaos., vol. 28, 2018, Art. no. 123118.

[52] Intel®, “Architecture Instruction Set Extensions Programming Reference,” Version 319433-044. Jun. 08, 2021. [Online]. Available: https://software.intel.com/content/www/us/en/developer/download/intel-architecture-instruction-set-extensions-programming-reference.html.

[53] C.R. Harris et al. “Array programming with NumPy,” Nature., vol. 585, no. 7825, pp. 357-362, 2020.

[54] M. Gataric et al. “Reconstruction of optical vector-fields with applications in endoscopic imaging,” IEEE T. Med. Imaging., vol. 38, no. 4, pp. 955-967, 2018.

[55] G.W. Allan, “The USC-SIPI Image Database,” Version 6. Feb. 03, 2018. [Online]. Available: http://sipi.usc.edu/database/SIPI_Database.pdf

---

Bin Ge is a Lecturer in Nantong Vocational University. He received his Ph.D. degree in engineering from the University of Chinese Academy of Sciences in 2016. His current research interests include the theory and application of cryptography and chaotic cryptography.

Xu Chen is an Associate Researcher in the Institute of Semiconductors, CAS. She received his Ph.D. degree in engineering from the Institute of Semiconductors, CAS in 2005. Her current research interests include image processing, artificial intelligent and pattern recognition.

Gang Chen is an Associate Researcher in the Institute of Semiconductors, CAS. He received his Ph.D. degree in engineering from the University of CAS in 2013. His current research interests include image processing and artificial intelligence.

Zhihua Shen is a Lecturer at Nantong Vocational University. He received his Ph.D. degree in engineering from Xi’an Jiao Tong University in 2017. His current research interests include chaos theory and circuit design.