Hindawi
Complexity
Volume 2021, Article ID 8824915, 18 pages
https://doi.org/10.1155/2021/8824915

*Research Article*

# An Effective Chaos-Based Image Encryption Scheme Using Imitating Jigsaw Method

**Zhen Li**,[1,2] **Changgen Peng**,[1] **Weijie Tan**,[1] **and Liangrong Li**[2]

[1]*College of Computer Science and Technology, State Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China*
[2]*College of Big Data and Information Engineering, Guizhou University, Guiyang 550025, China*

Correspondence should be addressed to Changgen Peng; cgpeng@gzu.edu.cn

Received 20 August 2020; Revised 25 September 2020; Accepted 6 February 2021; Published 20 February 2021

Academic Editor: Jesus M. Muñoz-Pacheco

In this paper, an efficient chaos-based image encryption scheme is proposed, which uses the imitating jigsaw method containing revolving and shifting operations. In this scheme, there are three processes in encryption: preprocessing, encryption process, and postprocessing. In the preprocessing, the original image is partitioned into $64 \times 64$ pixel image blocks and then randomly revolved and shifted under control sequences which are generated by the hyperchaotic Lorenz system whose initial conditions are calculated by original image and keys. Therefore, the preprocessing is sensitive to plain image against differential attacks. In the encryption process, the after-preprocessing image is partitioned into $32 \times 32$ pixel image blocks; next they are randomly revolved and encrypted by control sequence and key blocks which are generated by the skew tent map. In postprocessing, the after-encryption image is partitioned into $16 \times 16$ pixels' image blocks, and they are randomly revolved and shifted again under control sequences which are related with encrypted image and keys. The postprocessing further increases the diffusion characteristics. Moreover, the test experiment and security analyses are given; the results show that our proposed cryptosystem has both security and speed performance.

## 1. Introduction

With the rapid development of electronic technology, digital information is being applied in all the fields in the society. Digital image is becoming the most widely used information carrier in our daily life with many advantages, such as intuition and vividness. At the same time, a series of security problems of image information threaten the safety of peoples' life and property. Therefore, the information security of digital image has become a research focus in recent years. Because the image data has many special characteristics which are different from text structure data, such as high redundancy, strong correlation, and bulk data capacity, traditional text encryption algorithms usually have low efficiency on image encryption [1]. Therefore, the image encryption scheme must be designed by considering those characteristics.

Since the chaos theory was first proposed by Lorenz [2], many chaotic phenomena were found in many fields, such as physics, astronomy, chemistry, biology, and medicine. In recent years, the chaotic encryption attracts more and more researcher's attention. After Matthews [3] using the chaotic system to design the first image encryption scheme, many chaotic image encryption schemes are proposed in the recent years [3–38]. Fridrich [26] gave a two-dimensional chaotic symmetric ciphers scheme. Lian et al. [27] proposed a block cipher using a chaotic standard map. At the same period, many chaotic encryption schemes were put forward, such as [19–24]. Many of these schemes have been proved to have some security problems, such as weak plaintext sensitivity, small key space, and easy statistical attack.

In the past five years, many splendid image encryption schemes were presented. Hu et al. [4] proposed a chaotic image cipher scheme by using a plaintext-related permutation mechanism. In [5], an image encryption scheme was presented by using chaos sequence to control the encoding plain image to DNA sequence, and then they were encrypted by cycle operation of DNA sequence. In [6], Hua et al.

proposed a new 2D logistic-sine-coupling map and then designed an image encryption scheme using this chaotic system. In [7], Gao et al. gave a new 2D logistic ICMIC cascade map and then proposed a bit-level image encryption algorithm based on this map. Chai et al. [8] used the memristive chaotic system, elementary cellular automata, and compressive sensing to design an image encryption cryptosystem. Ye et al. [9] proposed an image encryption scheme by using SHA-3 hash function and double Arnold chaotic maps. Luo et al. [10] designed a plain image-related cryptosystem by using the tent map to generate two ergodicity sequences to control the permutation and diffusion processes. In [11], Hua et al. proposed an image encryption scheme for medical image by using high-speed scrambling and pixel adaptive diffusion. Khelifi et al. [12] presented an encryption scheme for image data sharing in cloud. In [13], Wang and Zhang used the hyperchaotic system to design an image encryption scheme which expanded the plain image into two compound images and then diffused them at the bit level. In [14], an image encryption scheme was presented which encrypted the image from four directions by interweaving pairs of rows and columns. In [15], an image encryption scheme was designed by using rows and columns switch. In [16], Liu et al. designed a self-adaptive selective permutation and inter-intra-block feedback diffusion and then used them by designing an image encryption scheme. Wu et al. [17] proposed a color image encryption scheme using the nonlinear chaotic algorithm (NCA) map-based coupled map lattice (CML) and DNA operation. Wang et al. [18] used chaos and simulated annealing algorithm to design the image encryption scheme. Li et al. [29] proposed a plaintext related image encryption scheme based on the hyperchaotic system and hash function. In [30], Wu et al. proposed an image encryption algorithm by using Henon-Sine map and DNA approach. Ye et al. [31] proposed an image encryption scheme by using quaternion discrete fractional Hartley transform and an improved pixel adaptive diffusion. In [32], Zhu et al. use block compressive sensing and singular value decomposition embedding to balance the security, compression, and robustness. In [33], an image encryption scheme is proposed by using 2D multiple parameter fractional discrete Fourier transform and 3D Arnold transform. You et al. [34] proposed a parallel image encryption scheme and implemented by OpenCL. Ouyang et al. [35] proposed a method of impulsive synchronization of coupled delayed neural networks with actuator saturation and designed an image encryption scheme by using this method. In [36], a compressed sensing strategy based on a semitensor product was proposed and a visual security image encryption scheme was designed. In [37], an image encryption scheme was proposed by using compressive sensing and random number insertion. Zhang et al. [38] proposed a plaintext-related image encryption scheme by using perceptron-like network. Li et al. [39] proposed an efficient bit-level permutation method and designed a chaos-based color image encryption scheme. Yavuz et al. [40] proposed an effective image encryption algorithm using two independent chaotic functions and some logical operations. Yavuz [41] proposed a content sensitive dynamic function and used this mechanism to design an image encryption scheme. Amina et al. [42] proposed an image encryption scheme using logistic-tent system. Ravichandran et al. [43] presented an image encryption scheme based on combined logistic-tent system. Lakshmi et al. [44] proposed a Hopfield attractor-based image encryption scheme using neural networks. Banu et al. [45] proposed an image encryption based on chaotic attractors on frequency domain by integer wavelet transform (IWT) and fused with deoxyribonucleic acid (DNA) sequence on the spatial domain. Alawida et al. [46] proposed a new chaotification method which combined two chaotic maps and designed an image encryption scheme using this method. Alawida et al. [47] used a deterministic finite state machine to enhance chaotic properties of tent map and proposed an image encryption scheme using this new chaotic system. Artiles et al. [48] combined the block cipher and chaotic system to design an image encryption scheme; the drawback of this scheme is less efficiency. Wang et al. [49] proposed an image encryption algorithm using logistic-dynamic mixed linear-nonlinear coupled map lattices. Luo et al. [50] presented an image encryption scheme using improved baker map and logistic map.

As we all know, a good image encryption scheme not only should have an excellent security performance but also need to have an outstanding encryption speed performance. The main contributions of our work are as follows:

(1) We proposed an efficient image encryption scheme based on chaos theory and imitating jigsaw operation
(2) We give a complete security analysis and performance analysis
(3) The proposed scheme is proved to have good encryption and decryption efficiencies compared with those of others' works

The organization of this paper is as follows: in Section 2, we give a detailed description of the encryption and decryption algorithms. In Section 3, we simulate our proposed cryptosystem and use some security analyses to prove our work is secure. Section 4 concludes this paper.

## 2. Encryption and Decryption Algorithms

In our proposed image cryptosystem, there are three processes in both encryption and decryption algorithms: preprocessing process, encryption or decryption process, and postprocessing process. In preprocessing and postprocessing processes, we use the hyperchaotic Lorenz system to generate the control sequences of revolving and shifting image blocks. In the encryption or decryption process, we use the skew tent system to generate the control sequence of revolving image blocks and the security key for encryption or decryption.

*Remark 1.* In our proposed scheme, we use the hyperchaotic system and skew tent map to control the encryption process. However, the other chaotic systems can also be extended in our scheme, and the only difference is the size of the key space.

The hyperchaotic Lorenz system [51] is given by

$$\begin{cases} \dot{x} = a(y-x) + w, \\ \dot{y} = cx - y - xz, \\ \dot{z} = xy - bz, \\ \dot{w} = -yz + \gamma w, \end{cases} \quad (1)$$

where $a$, $b$, $c$, and $\gamma$ are the parameters of the system. When $a = 10$, $b = 8/3$, $c = 28$, and $\gamma \in [-1.52, -0.06]$, the system goes into chaos. Figure 1 shows attractors of the hyperchaotic Lorenz system.

The skew tent system [52] is given by

$$t_{n+1} = \begin{cases} \frac{t_n}{p}, & t_n \in (0, p), \\ (1-t_n)(1-p), & t_n \in [p, 1), \end{cases} \quad (2)$$

where $p$ is the parameter of the skew tent system. When$p \in (0, 0.5) \cup (0.5, 1)$, the system can generate a chaotic sequence$t_n \in (0, 1)$.

*2.1. Encryption Algorithm.* The whole encryption scheme as shown in Figure 2 has two parts: image data processing before encryption part and encryption algorithm part. The purpose of data processing before encryption is to make input images suitable for processing rules of encryption algorithms. The details are as follows:

Step 1: we supposed the inputted image **IM** is an $\mathbf{M0} \times \mathbf{N0}$ 8 bit gray image and check whether **M0** and **N0** can be divided exactly by 64.

Step 2: if **M0** cannot be divided exactly by 64, but **N0** can, then execute Step 3a. If **N0** cannot be divided exactly by 64, but **M0** can, then execute Step 3b. And, if both **M0** and **N0** cannot be divided exactly by 64, then execute Step 3c (only one of the Steps 3a, 3b, and 3c can be executed).

Step 3a: take an integer number between 0 and 63 and denote it as **nm**, to make $\mathbf{M0} + \mathbf{nm}$ to be divided exactly by 64. Build an $\mathbf{nm} \times \mathbf{N0}$ matrix **NBM** whose each element is equal **nm.** Combine the two matrixes **IM** and **NBM** to be a new $(\mathbf{M0} + \mathbf{nm}) \times \mathbf{N0}$ matrix, as shown in Figure 3.

Step 3b: take an integer number between 0 and 63 and denote it as **nn**, to make $\mathbf{N0} + \mathbf{nn}$ to be divided exactly by 64. Build a $\mathbf{M0} \times \mathbf{nn}$ matrix **NBN** whose each element is equal **nn.** Combine the two matrixes **IM** and **NBN** to be a new $\mathbf{M0} \times (\mathbf{N0} + \mathbf{nn})$ matrix, as shown in Figure 3.

Step 3c: take an integer number between 0 and 63 and denote it as **nm**, to make $\mathbf{M0} + \mathbf{nm}$ to be divided exactly by 64. Take an integer number between 0 and 63 and denote it as **nn**, to make $\mathbf{N0} + \mathbf{nn}$ to be divided exactly by 64. Generate a new $(\mathbf{M0} + \mathbf{nm}) \times (\mathbf{N0} + \mathbf{nn})$ matrix by **IM**, **NBN**, **NBM,** and **NBT** as Figure 3, where elements in matrix **NBM** are equal to **nm**, elements in matrix **NBN** are equal to **nn**, and elements in matrix **NBT** are equal to $\mathbf{nm} + \mathbf{nn}$.

Step 4: finish the data processing with outputting $\mathbf{M} \times \mathbf{N}$ image matrix **IM1**.

After data processing, we put the outputted-image matrix into encryption algorithm. There are three processes in encryption algorithm: preprocessing process, encryption process, and postprocessing process. The encryption algorithm is shown in Algorithm 1.

The detailed description of encryption algorithm is as follows:

Step 1: input the data processed image **IM1** and initial key $\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_9$ into the encryption algorithm.

Step 2: the preprocessing process begins. **IM1** is partitioned as $64 \times 64$ pixels image blocks, and there are $\mathbf{BN1} = (M/64) \times (N/64)$ blocks; they are denoted as **preBlock** (1), **preBlock** (2), . . ., **preBlock** (**BN1**).

Step 3: add all the gray values of pixels of the **IM1**, and the summation is denoted as **S1**. The initial conditions of the hyperchaotic Lorenz system are generated by equations (3)–(6):

$$\mathbf{x}_0 = (\mathrm{mod}(\mathbf{k}_1 \times \mathbf{S}1, 80) - 40) \times \mathbf{k}_2 + \mathbf{k}_6, \quad (3)$$

$$\mathbf{y}_0 = (\mathrm{mod}(\mathbf{k}_2 \times \mathbf{S}1, 80) - 40) \times \mathbf{k}_3 + \mathbf{k}_7, \quad (4)$$

$$\mathbf{z}_0 = (\mathrm{mod}(\mathbf{k}_3 \times \mathbf{S}1, 80) + 1) \times \mathbf{k}_4 + \mathbf{k}_8, \quad (5)$$

$$\mathbf{w}_0 = (\mathrm{mod}(\mathbf{k}_4 \times \mathbf{S}1, 500) - 250) \times \mathbf{k}_5 + \mathbf{k}_9, \quad (6)$$

where mod (a, b) means the remainder of a/b (similarly hereinafter).

Step 4: under initial conditions which are generated in Step 3, equation (1) is iterated by using the fourth-order Runge-Kutta method with 0.002 step size. The revolving control sequence **RS1** and the shifting control sequence **XS1** are generated by equations (7) and (8), respectively:

$$\mathbf{RS}1 = \mathrm{mod}\Big(\mathrm{floor}\Big((y + 100) \times 10^8, 4\Big), \quad (7)$$

$$\mathbf{XS}1 = \mathrm{mod}\Big(\mathrm{floor}\Big((x + 100) \times 10^8, \mathbf{BN}1\Big) + 1, \quad (8)$$

Step 5: image blocks are revolved by using the following equation:

$$\mathbf{preBlock}(i) = \mathrm{imrotate}(\mathbf{preBlock}(i), 90 \times \mathbf{RS}1(i)), \quad (9)$$

where imrotate$(P, m)$ means image $P$ is rotated $m$ degrees in the anticlockwise direction (similarly hereinafter) and$i$is the serial number of elements in the sequences **preBlock** and **RS1.**

Step 6: remove the repeated elements from **XS1** and then put the absent numbers at the end to generate a new sequence **X1**.

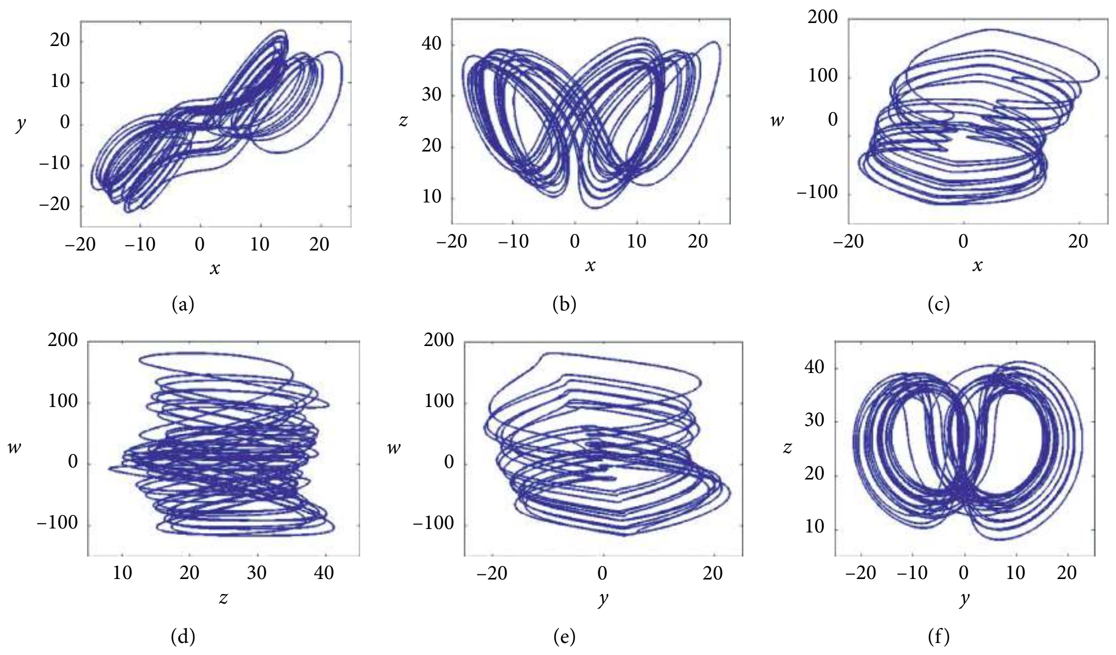


Figure 1: Hyperchaotic attractors in (a) $x$-$y$ plane, (b) $x$-$z$ plane, (c) $x$-$w$ plane, (d) $z$-$w$ plane, (e) $y$-$w$ plane, and (f) $y$-$z$ plane.

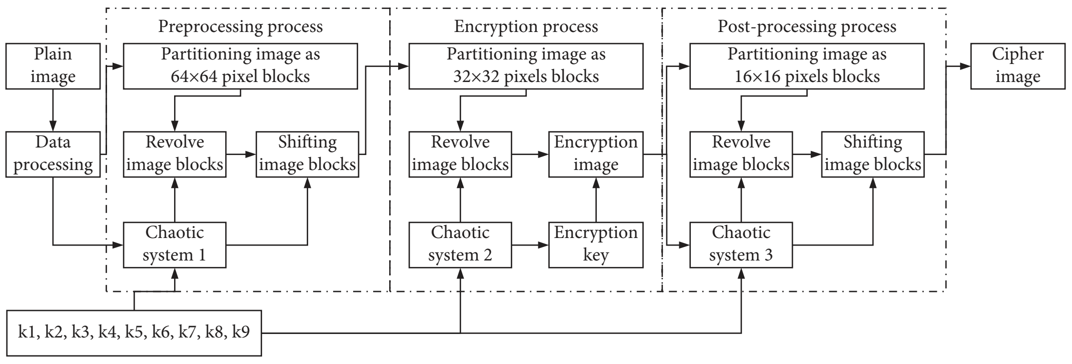


Figure 2: Encryption scheme.

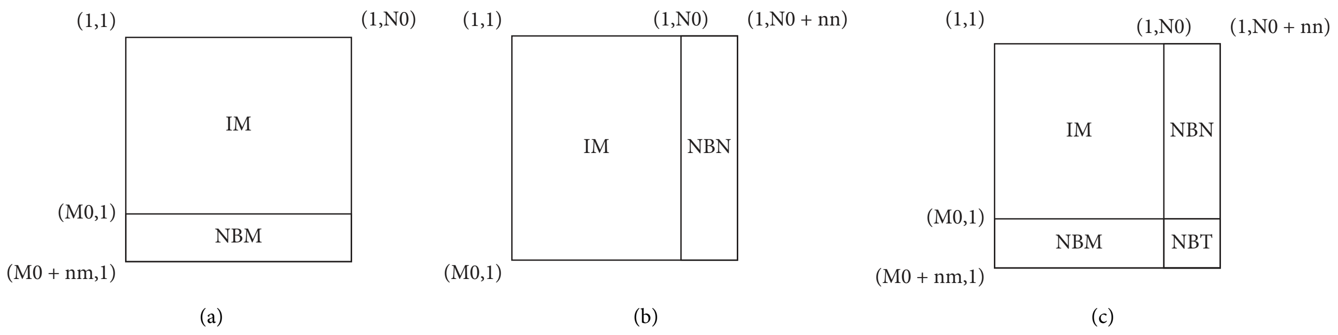


Figure 3: The rules of matrix combination.

```
     Input: image IM1 and initial key $[[\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_9]]$
 (1) [M N] ⟵ size (IM1);
 (2) Partitioning IM1 as $64 \times 64$ pixels image blocks
         [preBlock (1), preBlock (2), ..., preBlock (BN1)], where $\mathbf{BN}1 = (M/64) \times (N/64)$
 (3) $S1 \longleftarrow \mathrm{SUM}(\mathrm{IM1})$
 Generate initial conditions of the hyperchaotic Lorenz system:
     $\mathbf{x}_0 \longleftarrow (\mathrm{mod}(\mathbf{k}_1 \times \mathbf{S}1, 80) - 40) \times \mathbf{k}_2 + \mathbf{k}_6; \mathbf{y}_0 \longleftarrow (\mathrm{mod}(\mathbf{k}_2 \times \mathbf{S}1, 80) - 40) \times \mathbf{k}_3 + \mathbf{k}_7;$
     $\mathbf{z}_0 \longleftarrow (\mathrm{mod}(\mathbf{k}_3 \times \mathbf{S}1, 80) + 1) \times \mathbf{k}_4 + \mathbf{k}_8; \mathbf{w}_0 \longleftarrow (\mathrm{mod}(\mathbf{k}_4 \times \mathbf{S}1, 500) - 250) \times \mathbf{k}_5 + \mathbf{k}_9;$
 (4) Iterate equation (1) with$\mathbf{x}_0, \mathbf{y}_0, \mathbf{z}_0, \mathbf{w}_0$.
 Generate revolving control sequence $\mathbf{RS}1 \longleftarrow \mathrm{mod}(\mathrm{floor}((y + 100) \times 10^8, 4)$
     Generate shifting control sequence $\mathbf{XS}1 \longleftarrow \mathrm{mod}(\mathrm{floor}((x + 100) \times 10^8, \mathrm{BN1}) + 1$
 (5) for$i = 1$ to BN1then
   $\mathbf{preBlock}(i) \longleftarrow \mathrm{imrotate}(\mathbf{preBlock}(i), 90 \times \mathbf{RS}1(i))$
     end for
 (6) Generate a new sequence X1 by removing repeated elements from XS1 and putting the absent numbers at the end.
 (7) for$i = 1$ to BN1/2then
   $\mathbf{preBlock}(\mathbf{X1}_\mathbf{i}) \leftrightarrow \mathbf{preBlock}(\mathbf{X1}_{\mathbf{BN1-i+1}})$
     end for
 (8) Restructure preBlock to an image matrix IM2
 (9) Partitioning IM2 as $32 \times 32$ pixels' image blocks
         [encBlock (1), encBlock (2),..., encBlock (BN2)], where $\mathbf{BN}2 = (M/32) \times (N/32)$
(10) Generate system parameter $\boldsymbol{p}$ and initial condition $\mathbf{t}_0$ of skew tent map:
     $\mathbf{p} \longleftarrow \mathrm{mod}((\mathbf{k}_1 + \mathbf{k}_2 + \mathbf{k}_3) \times \mathbf{k}_4 + \mathbf{k}_5, 1); \mathbf{t}_0 \longleftarrow \mathrm{mod}((\mathbf{k}_5 + \mathbf{k}_6 + \mathbf{k}_7) \times \mathbf{k}_8 + \mathbf{k}_9, 1)$
(11) Iterate equation (2) with$\mathbf{p}, \mathbf{t}_0$.
 Generate revolving control sequence $\mathbf{RS}2 \longleftarrow \mathrm{mod}(\mathrm{floor}((\mathbf{t} + 100) \times 10^8, 4)$
     Generate key sequence $\mathbf{KS} \longleftarrow \mathrm{mod}(\mathrm{floor}(\mathbf{t} \times 10^{12}), 256)$
(12) Reshape key sequence KS to be a $32 \times 32$ blocks
       [keyBlock (1), keyBlock (2),..., keyBlock (BN2)]
(13) for $i = 1$ to BN2then
       $\mathbf{encBlock}(i) \longleftarrow \mathrm{imrotate}(\mathbf{encBlock}(i), 90 \times \mathbf{RS}2(i))$
     end for
(14) $\mathbf{cBlock}(1) \longleftarrow \mathbf{encBlock}(1) \oplus \mathbf{keyBlock}(1)$
 for$i = 2$ to BN2then
       $\mathbf{cBlock}(i) \longleftarrow \mathbf{encBlock}(i) \oplus \mathbf{keyBlock}(i) \oplus \mathbf{cBlock}(i - 1)$
     end for
(15) Restructure cBlock to an image matrix IM3
(16) Partitioning IM3 as $16 \times 16$ pixels' image blocks:
       [postBlock (1), postBlock (2),..., postBlock (BN3)], where $\mathbf{BN}3 = (M/16) \times (N/16)$
(17) $S2 \longleftarrow \mathrm{SUM}(\mathrm{IM3})$
     Generate initial conditions of hyper-chaotic Lorenz system:
 $\mathbf{x}_0 \longleftarrow (\mathrm{mod}(\mathbf{k}_6 \times \mathbf{S}2, 80) - 40) \times \mathbf{k}_4 + \mathbf{k}_1; \mathbf{y}_0 \longleftarrow (\mathrm{mod}(\mathbf{k}_7 \times \mathbf{S}2, 80) - 40) \times \mathbf{k}_3 + \mathbf{k}_2;$
 $\mathbf{z}_0 \longleftarrow (\mathrm{mod}(\mathbf{k}_8 \times \mathbf{S}2, 80) + 1) \times \mathbf{k}_2 + \mathbf{k}_3; \mathbf{w}_0 \longleftarrow (\mathrm{mod}(\mathbf{k}_9 \times \mathbf{S}2, 500) - 250) \times \mathbf{k}_1 + \mathbf{k}_4;$
(18) Iterate equation (1) with$\mathbf{x}_0, \mathbf{y}_0, \mathbf{z}_0, \mathbf{w}_0$.
     Generate revolving control sequence $\mathbf{RS}3 \longleftarrow \mathrm{mod}(\mathrm{floor}((y + 100) \times 10^8, 4)$
     Generate shifting control sequence $\mathbf{XS}2 \longleftarrow \mathrm{mod}(\mathrm{floor}((x + 100) \times 10^8, \mathbf{BN}3) + 1$
(19) for $i = 1$ to BN3 then
         $\mathbf{postBlock}(i) \longleftarrow \mathrm{imrotate}(\mathbf{postBlock}(i), 90 \times \mathbf{RS}3(i))$
     end for
(20) Generate a new sequence X2 by removing repeated elements from XS2, and putting the absent numbers at the end.
(21) for $i = 1$ to BN3/2then
         $\mathbf{postBlock}(\mathrm{X2}_\mathbf{i}) \leftrightarrow \mathbf{postBlock}(\mathbf{X2}_{\mathbf{BN3-i+1}})$
     end for
(22) Restructure postBlock to an image matrix CM
     Output: cipher image CM
```

Algorithm 1: Encryption algorithm.

Step 7: substitute $\textbf{preBlock}(X1_{\mathbf{i}})$ with$\textbf{preBlock}(\mathbf{X1_{BN1-i+1}})$, where$i = 1, 2, \ldots, \text{BN1}/2$.

Step 8: generate a new image **IM2** by restructuring **preBlock**. The preprocessing process is finished.

Step 9: The encryption process begins. **IM2** is partitioned as $32 \times 32$ pixels image blocks, and there are $\textbf{BN}2 = (M/32) \times (N/32)$ blocks; they are denoted as **encBlock** (1), **encBlock** (2),. . ., **encBlock** (**BN2**).

Step 10: generate system parameter ***p*** and initial condition $\mathbf{t}_0$ of skew tent map by equations (10) and (11), respectively:

$$\mathbf{p} = \text{mod}\left(\left(\mathbf{k}_1 + \mathbf{k}_2 + \mathbf{k}_3\right) \times \mathbf{k}_4 + \mathbf{k}_5, 1\right), \tag{10}$$

$$\mathbf{t}_0 = \text{mod}\left(\left(\mathbf{k}_5 + \mathbf{k}_6 + \mathbf{k}_7\right) \times \mathbf{k}_8 + \mathbf{k}_9, 1\right), \tag{11}$$

Step 11: equation (2) was iterated under system parameter $p$ and initial condition $\mathbf{t}_0$. The revolving control sequence **RS2** and the key sequence **KS** are generated by equations (12) and (13), respectively:

$$\textbf{RS}2 = \text{mod}\Big(\text{floor}\Big((\mathbf{t} + 100) \times 10^8, 4\Big), \tag{12}$$

$$\textbf{KS} = \text{mod}\Big(\text{floor}\Big(\mathbf{t} \times 10^{12}\Big), 256\Big). \tag{13}$$

Step 12: reshape key sequence KS to be BN2 key blocks with $32 \times 32$ and denote them as **keyBlock (1)**, **keyBlock (2)**,. . ., **keyBlock (BN2).**

Step 13: Revolve the image blocks by using the following equation:

$$\textbf{encBlock}(i) = \text{imrotate}(\textbf{encBlock}(i), 90 \times \textbf{RS}2(i)). \tag{14}$$

Step 14: encrypt image blocks by the following equation:

$$\begin{cases} \textbf{cBlock}(i) = \textbf{encBlock}(i) \oplus \textbf{keyBlock}(i), & i = 1, \\ \textbf{cBlock}(i) = \textbf{encBlock}(i) \oplus \textbf{keyBlock}(i) \oplus \textbf{cBlock}(i-1), & 2 \leq i \leq \text{BN2}. \end{cases} \tag{15}$$

Step 15: generate a new image **IM3** by restructuring **cBlock.** The encryption process is finished.

Step 16: the postprocessing process begins. **IM3** is partitioned as $16 \times 16$ pixels image blocks, and there are $\textbf{BN}3 = (M/16) \times (N/16)$ blocks; they are denoted as **postBlock** (1), **postBlock** (2), . . ., **postBlock** (**BN3**).

Step 17: add all the gray values of pixels of the **IM3**, and the summation is denoted as **S2.** The initial conditions of the hyperchaotic Lorenz system are generated by the following equations:

$$\mathbf{x}_0 = \left(\text{mod}\left(\mathbf{k}_6 \times \mathbf{S}2, 80\right) - 40\right) \times \mathbf{k}_4 + \mathbf{k}_1, \tag{16}$$

$$\mathbf{y}_0 = \left(\text{mod}\left(\mathbf{k}_7 \times \mathbf{S}2, 80\right) - 40\right) \times \mathbf{k}_3 + \mathbf{k}_2, \tag{17}$$

$$\mathbf{z}_0 = \left(\text{mod}\left(\mathbf{k}_8 \times \mathbf{S}2, 80\right) + 1\right) \times \mathbf{k}_2 + \mathbf{k}_3, \tag{18}$$

$$\mathbf{w}_0 = \left(\text{mod}\left(\mathbf{k}_9 \times \mathbf{S}2, 500\right) - 250\right) \times \mathbf{k}_1 + \mathbf{k}_4. \tag{19}$$

Step 18: under initial conditions which are generated in Step 17, equation (1) is iterated by using the fourth-order RungeKutta method with 0.002 step size. The revolving control sequence **RS3** and the shifting control sequence **XS2** are generated by equations (20) and (21), respectively:

$$\textbf{RS}3 = \text{mod}\Big(\text{floor}\Big((y + 100) \times 10^8, 4\Big), \tag{20}$$

$$\textbf{XS}2 = \text{mod}\Big(\text{floor}\Big((x + 100) \times 10^8, \textbf{BN}3\Big) + 1 \tag{21}$$

Step 19: image blocks are revolved by the following equation:

$$\textbf{postBlock}(i) = \text{imrotate}(\textbf{postBlock}(i), 90 \times \textbf{RS}3(i)). \tag{22}$$

Step 20: remove the repeated elements from **XS2** and then put the absent numbers at the end to generate a new sequence **X2.**

Step 21: substitute $\textbf{postBlock}(\mathbf{X2_i})$ with$\textbf{postBlock}(\mathbf{X2_{BN3-i+1}})$, where$i = 1, 2, \ldots, \text{BN3}/2$.

Step 22: generate cipher image **CM** by restructuring **postBlock.** The postprocessing process is finished.

Step 23: output the cipher image, and the encryption algorithm is finished.

### 2.2. Decryption Algorithm.

*2.2. Decryption Algorithm.* The decryption scheme as shown in Figure 4 has two parts as encryption: the decryption algorithm part and data processing part. The decryption algorithm is the inverse process of the encryption algorithm and also has three processes: preprocessing process, decryption process, and postprocessing process. The decryption algorithm is shown in Algorithm 2.

The detail description of the decryption algorithm as follows:

Step 1: input cipher image **CM1** and initial key $\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_9$ into decryption algorithm, and the cipher image is supposed to be $M \times N$ 8 bit gray image.

Note: inputted security keys are defined as$\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_9 \in (\mathbf{0}, \mathbf{1})$, which are used to generate parameter and initial values of the hyperchaotic Lorenz system and skew tent map. To avoid weak key problem [53], we require each input key to be 15 decimal places.

Step 2: the preprocessing process begins. Partition **CM1** as $16 \times 16$ pixels image blocks, and there are **BN**4 =

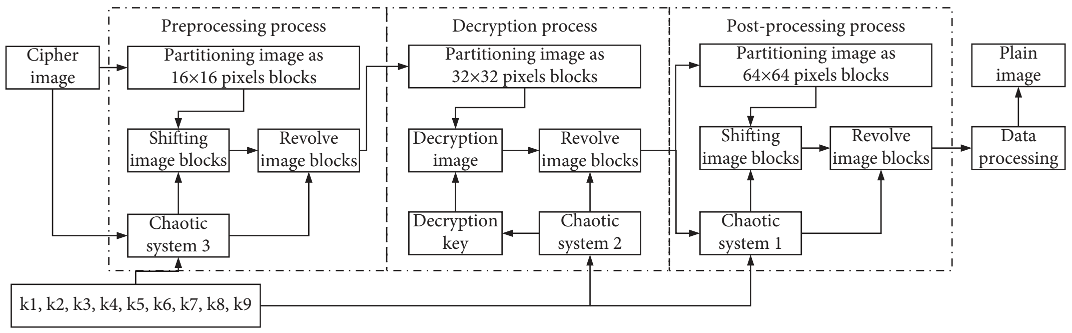


FIGURE 4: Decryption scheme.

```
     Input: cipher image CM1 and initial key $[[\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_9]]$
(1)  [M N]⟵size (CM1);
(2)  Partitioning CM1 as $16 \times 16$ pixels image blocks:
         [preCBlock (1), preCBlock (2), ..., preCBlock (BN4)], where $\mathbf{BN}4 = (M/16) \times (N/16)$
(3)  $S3 \longleftarrow \mathrm{SUM}(\mathrm{CM1})$
     Generate initial conditions of the hyperchaotic Lorenz system:
     $\mathbf{x}_0 \longleftarrow (\mathrm{mod}(\mathbf{k}_6 \times \mathbf{S}3, 80) - 40) \times \mathbf{k}_4 + \mathbf{k}_1; \mathbf{y}_0 \longleftarrow (\mathrm{mod}(\mathbf{k}_7 \times \mathbf{S}3, 80) - 40) \times \mathbf{k}_3 + \mathbf{k}_2$
     $\mathbf{z}_0 \longleftarrow (\mathrm{mod}(\mathbf{k}_8 \times \mathbf{S}3, 80) + 1) \times \mathbf{k}_2 + \mathbf{k}_3; \mathbf{w}_0 \longleftarrow (\mathrm{mod}(\mathbf{k}_9 \times \mathbf{S}3, 500) - 250) \times \mathbf{k}_1 + \mathbf{k}_4$
(4)  Iterate equation (1) with$\mathbf{x}_0, \mathbf{y}_0, \mathbf{z}_0, \mathbf{w}_0$.
     Generate revolving control sequence $\mathbf{RS}4 \longleftarrow \mathrm{mod}(\mathrm{floor}((y + 100) \times 10^8, 4)$
     Generate shifting control sequence $\mathbf{XS}3 \longleftarrow \mathrm{mod}(\mathrm{floor}((x + 100) \times 10^8, \mathbf{BN}4) + 1$
(5)  Generate a new sequence X3 by removing repeated elements from XS3 and putting the absent numbers at the end.
(6)  for$i = 1$ to BN4/2then
         $\mathbf{preCBlock}(\mathrm{X3}_{\mathbf{i}}) \leftrightarrow \mathbf{preCBlock}(\mathbf{X3}_{\mathbf{BN4-i+1}})$
     end for
(7)  for$i = 1$ to BN4then
         $\mathbf{preCBlock}(i) \longleftarrow \mathrm{imrotate}(\mathbf{preCBlock}(i), -90 \times \mathbf{RS}4(i))$
     end for
(8)  Restructure preCBlock to an image matrix CM2
(9)  Partitioning CM2 as $32 \times 32$ pixels' image blocks:
         [decCBlock (1), decCBlock (2),..., decCBlock (BN5)], where $\mathbf{BN}5 = (M/32) \times (N/32)$
(10) Generate system parameter $\boldsymbol{p}$ and initial condition $\mathbf{t}_0$ of skew tent map:
     $\mathbf{p} \longleftarrow \mathrm{mod}((\mathbf{k}_1 + \mathbf{k}_2 + \mathbf{k}_3) \times \mathbf{k}_4 + \mathbf{k}_5, 1); \mathbf{t}_0 \longleftarrow \mathrm{mod}((\mathbf{k}_5 + \mathbf{k}_6 + \mathbf{k}_7) \times \mathbf{k}_8 + \mathbf{k}_9, 1)$
(11) Iterate equation (2) with$\mathbf{p}, \mathbf{t}_0$.
     Generate revolving control sequence $\mathbf{RS}5 \longleftarrow \mathrm{mod}(\mathrm{floor}((\mathbf{t} + 100) \times 10^8, 4)$
     Generate key sequence $\mathbf{KS}1 \longleftarrow \mathrm{mod}(\mathrm{floor}(\mathbf{t} \times 10^{12}), 256)$
(12) Reshape key sequence KS1 to be a $32 \times 32$ blocks
        [keyCBlock (1), keyCBlock (2),..., keyCBlock (BN5)]
(13) for$i = \mathbf{BN}5$ to 2 step $-$ 1then
        $\mathbf{pBlock}(i) \longleftarrow \mathbf{de\,cCBlock}(i) \oplus \mathbf{keyCBlock}(i) \oplus \mathbf{pBlock}(i-1)$
     end for
     $\mathbf{pBlock}(1) \longleftarrow \mathbf{de\,cCBlock}(1) \oplus \mathbf{keyCBlock}(1)$
(14) for$i = 1$ to BN5then
        $\mathbf{de\,cCBlock}(i) \longleftarrow \mathrm{imrotate}(\mathbf{de\,cCBlock}(i), -90 \times \mathbf{RS}5(i))$
     end for
(15) Restructure pBlock to an image matrix CM3
(16) Partitioning CM3 as $64 \times 64$ pixels' image blocks:
     [postCBlock (1), postCBlock (2), ..., postCBlock (BN6)], where $\mathbf{BN}6 = (M/64) \times (N/64)$
(17) $S4 \longleftarrow \mathrm{SUM}(\mathrm{CM3})$
     Generate initial conditions of the hyperchaotic Lorenz system:
     $\mathbf{x}_0 \longleftarrow (\mathrm{mod}(\mathbf{k}_1 \times \mathbf{S}4, 80) - 40) \times \mathbf{k}_2 + \mathbf{k}_6; \mathbf{y}_0 \longleftarrow (\mathrm{mod}(\mathbf{k}_2 \times \mathbf{S}4, 80) - 40) \times \mathbf{k}_3 + \mathbf{k}_7;$
     $\mathbf{z}_0 \longleftarrow (\mathrm{mod}(\mathbf{k}_3 \times \mathbf{S}4, 80) + 1) \times \mathbf{k}_4 + \mathbf{k}_8; \mathbf{w}_0 \longleftarrow (\mathrm{mod}(\mathbf{k}_4 \times \mathbf{S}4, 500) - 250) \times \mathbf{k}_5 + \mathbf{k}_9;$
(18) Iterate equation (1) with $\mathbf{x}_0, \mathbf{y}_0, \mathbf{z}_0, \mathbf{w}_0$.
     Generate revolving control sequence $\mathbf{RS}6 \longleftarrow \mathrm{mod}(\mathrm{floor}((y + 100) \times 10^8, 4)$.
     Generate shifting control sequence $\mathbf{XS}4 \longleftarrow \mathrm{mod}(\mathrm{floor}((x + 100) \times 10^8, \mathbf{BN}6) + 1$
```

ALGORITHM 2: Continued.

```
(19) Generate a new sequence X4 by removing repeated elements from XS4 and putting the absent numbers at the end.
(20) for$i$ = 1 to BN6/2then
         postCBlock$(X4_i)\leftrightarrow$postCBlock$(X4_{BN6-i+1})$
     end for
(21) for$i$ = 1 to BN6then
         postCBlock$(i) \longleftarrow$ imrotate(postCBlock$(i), -90 \times$ RS6$(i)$)
     end for
(22) Restructure postCBlock to an image matrix PM
     Output: plain image PM
```

ALGORITHM 2: Decryption algorithm.

$(M/16)\times(N/16)$ blocks; and they are denoted as **preCBlock** (1), **preCBlock** (2), . . ., **preCBlock** (**BN4**).

Step 3: add all the gray values of pixels of the **CM1**, and the summation is denoted as **S3.** The initial conditions of the hyperchaotic Lorenz system are generated by equations (23)–(26):

$$\mathbf{x}_0 = \left(\mathrm{mod}\left(\mathbf{k}_6 \times \mathbf{S}3, 80\right) - 40\right) \times \mathbf{k}_4 + \mathbf{k}_1, \tag{23}$$

$$\mathbf{y}_0 = \left(\mathrm{mod}\left(\mathbf{k}_7 \times \mathbf{S}3, 80\right) - 40\right) \times \mathbf{k}_3 + \mathbf{k}_2, \tag{24}$$

$$\mathbf{z}_0 = \left(\mathrm{mod}\left(\mathbf{k}_8 \times \mathbf{S}3, 80\right) + 1\right) \times \mathbf{k}_2 + \mathbf{k}_3, \tag{25}$$

$$\mathbf{w}_0 = \left(\mathrm{mod}\left(\mathbf{k}_9 \times \mathbf{S}3, 500\right) - 250\right) \times \mathbf{k}_1 + \mathbf{k}_4. \tag{26}$$

Step 4: under initial conditions which are generated in Step 3, equation (1) is iterated by using the fourth-order Runge-Kutta method with 0.002 step size. The revolving control sequence **RS4** and the shifting control sequence **XS3** are generated by equations (27) and (28), respectively:

$$\mathbf{RS}4 = \mathrm{mod}\Big(\mathrm{floor}\big((y + 100) \times 10^8, 4\big), \tag{27}$$

$$\mathbf{XS}3 = \mathrm{mod}\Big(\mathrm{floor}\big((x + 100) \times 10^8, \mathbf{BN}4\Big) + 1 \tag{28}$$

Step 5: remove the repeated elements from **XS3** and then put the absent numbers at the end to generate a new sequence **X3.**

Step 6: substitute **preCBlock**$(\mathrm{X3_i})$ with**preCBlock** $(\mathbf{X3_{BN4-i+1}})$, where$i = 1, 2, \ldots, \mathrm{BN4}/2$.

Step 7: image blocks are revolved by the following equation:

$$\mathbf{preCBlock}\,(i) = \mathrm{imrotate}\left(\mathbf{preCBlock}\,(i), -90 \times \mathbf{RS}4\,(i)\right). \tag{29}$$

Step 8: generate a new image **CM2** by restructuring **preCBlock**. The preprocessing process is finished.

Step 9: the decryption process begins. Partition **CM2** as $32\times 32$ pixels image blocks, and there are **BN**5 = $(M/32)\times(N/32)$ blocks; denote them as **decCBlock (1)**, **decCBlock (2)**,. . ., **decCBlock (BN5).**

Step 10: generate system parameter $\boldsymbol{p}$ and initial condition $\mathbf{t}_0$ of skew tent map by equations (30) and (31), respectively:

$$\mathbf{p} = \mathrm{mod}\left(\left(\mathbf{k}_1 + \mathbf{k}_2 + \mathbf{k}_3\right) \times \mathbf{k}_4 + \mathbf{k}_5, 1\right), \tag{30}$$

$$\mathbf{t}_0 = \mathrm{mod}\left(\left(\mathbf{k}_5 + \mathbf{k}_6 + \mathbf{k}_7\right) \times \mathbf{k}_8 + \mathbf{k}_9, 1\right). \tag{31}$$

Step 11: iterate equation (2) under system parameter $\boldsymbol{p}$ and initial condition $\mathbf{t}_0$**.** The revolving control sequence **RS5** and the key sequence **KS1** are generated by equations (32) and (33), respectively:

$$\mathbf{RS}5 = \mathrm{mod}\Big(\mathrm{floor}\big((\mathbf{t} + 100) \times 10^8, 4\big), \tag{32}$$

$$\mathbf{KS}1 = \mathrm{mod}\Big(\mathrm{floor}\big(\mathbf{t} \times 10^{12}\big), 256\Big). \tag{33}$$

Step 12: reshape key sequence **KS1** to be **BN5** key blocks with $32\times 32$ and denote them as **keyCBlock** (1), **keyCBlock** (2),. . ., **keyCBlock** (**BN5**).

Step 13: decrypt the image blocks by the following equation:

$$\begin{cases} \mathbf{pBlock}\,(i) = \mathbf{de\,cCBlock}\,(i) \oplus \mathbf{keyCBlock}\,(i), & i = 1, \\ \mathbf{pBlock}\,(i) = \mathbf{de\,cCBlock}\,(i) \oplus \mathbf{keyCBlock}\,(i) \oplus \mathbf{pBlock}\,(i-1), & 2 \le i \le \mathrm{BN}5. \end{cases} \tag{34}$$

Step 14: revolve the image blocks by using the following equation:

$$\mathbf{de\,cCBlock}\,(i) = \mathrm{imrotate}\left(\mathbf{de\,cCBlock}\,(i), -90 \times \mathbf{RS}5\,(i)\right). \tag{35}$$

Step 15: generate a new image **CM3** by restructuring **pBlock**. The decryption process is finished.

Step 16: the preprocessing process begins. Partition **CM3** as $64\times 64$ pixels image blocks, and there are **BN**6 = $(M/64)\times(N/64)$ blocks; denote them as

**postCBlock** (1), **postCBlock** (2), ..., **postCBlock** (**BN6**).

Step 17: add all the gray values of pixels of the **CM3**, and the summation is denoted as **S4.** The initial conditions of the hyperchaotic Lorenz system are generated by the following equations:

$$\mathbf{x}_0 = (\mathrm{mod}(\mathbf{k}_1 \times \mathbf{S}4, 80) - 40) \times \mathbf{k}_2 + \mathbf{k}_6, \tag{36}$$

$$\mathbf{y}_0 = (\mathrm{mod}(\mathbf{k}_2 \times \mathbf{S}4, 80) - 40) \times \mathbf{k}_3 + \mathbf{k}_7, \tag{37}$$

$$\mathbf{z}_0 = (\mathrm{mod}(\mathbf{k}_3 \times \mathbf{S}4, 80) + 1) \times \mathbf{k}_4 + \mathbf{k}_8, \tag{38}$$

$$\mathbf{w}_0 = (\mathrm{mod}(\mathbf{k}_4 \times \mathbf{S}4, 500) - 250) \times \mathbf{k}_5 + \mathbf{k}_9. \tag{39}$$

Step 18: under initial conditions which are generated in Step 18, iterate equation (1) by using the fourth-order RungeKutta method with 0.002 step size. The revolving control sequence **RS6** and the shifting control sequence **XS4** are generated by equations (7) and (8), respectively:

$$\mathbf{RS}6 = \mathrm{mod}\Big(\mathrm{floor}\big((y + 100) \times 10^8, 4\big)\Big), \tag{40}$$

$$\mathbf{XS}4 = \mathrm{mod}\Big(\mathrm{floor}\big((x + 100) \times 10^8, \mathbf{BN6}\big) + 1 \tag{41}$$

Step 19: remove the repeated elements from XS4 and then put the absent numbers at the end to generate a new sequence **X4.**

Step 20: substitute $\mathbf{postCBlock}(\mathbf{X4_i})$ with$\mathbf{postCBlock}(\mathbf{X4_{BN6-i+1}})$, where $i = 1, 2, \ldots, \mathrm{BN}$ 6/2.

Step 21: revolve the image blocks by the following equation:

$$\mathbf{postCBlock}(i) = \mathrm{imrotate}(\mathbf{postCBlock}(i), -90 \times \mathbf{RS}6(i)). \tag{42}$$

Step 22: generate plain image **PM** by restructuring **postCBlock.** The postprocessing process is finished.

Step 23: output image matrix **PM**, and the decryption algorithm is finished.

After decryption algorithm part, we also need to run data processing to get the plain image. The purpose of this section is to remove the pixels when added by data processing before the encryption part; the detailed description is as follows:

Step 1: check the relationship of three pixels' value which are located in (1, N), (M, 1), and (M, N). We denote these three pixel value as **nn**, **nm,** and **nt**, respectively. If **nt** is not equal to **nn**, nm, or **nn + nm**, then execute Step 2a. If **nt** is equal to **nn**, then execute Step 2b. If **nt** is equal to **nm**, then execute Step 2c. If **nt** is equal to **nm + nn**, then execute Step 2d (only one of the Steps 2a, 2b, 2c, and 2d can be executed)

Step 2a: do nothing with **PM**, and the matrix **DPM** will be equal to **PM.**

Step 2b: check some elements' value in matrix **PM** which locate from rows **1** to **M** and columns **N-nn + 1** to **N.** If all those elements' value are equal to **nn**, then matrix **DPM** is equal to a part of matrix **PM** from rows **1** to **M** and columns **1** to **N-nn.** Otherwise, the matrix **DPM** is equal to **PM.**

Step 2c: check some elements' value in matrix **PM** which locate from rows **M-nm + 1** to **M** and columns **1** to **N.** If all those elements' value are equal to **nm**, then matrix **DPM** is equal to a part of matrix **PM** from rows **1** to **M-nm** and columns **1** to **N.** Otherwise, the matrix **DPM** is equal to **PM.**

Step 2d: check some elements' value in matrix **PM** which locate from rows **M-nm + 1** to **M** and columns **N-nn + 1** to **N.** If all those elements' value are equal to **nm + nm**, then matrix **DPM** is equal to a part of matrix **PM** from rows **1** to **M-nm** and columns **1** to **N-nn.** Otherwise, the matrix **DPM** is equal to **PM.**

Step 3: output the plain image **DPM**, and the whole decryption scheme is finished.

## 3. Simulation and Security Analysis

In this section, we evaluated our proposed scheme by software simulation running in MATLAB 2015b. The system parameters which are given in equation (1) are $a = 10$, $b = 8/3$, $c = 28$, and $\gamma = -1$, and the inputted keys are $k1 = 0.132312432152369$, $k2 = 0.723357645512358$, $k3 = 0.234342634695632$, $k4 = 0.432313435498574$, $k5 = 0.832312114474569$, $k6 = 0.642312345436958$, $k7 = 0.532345341689841$, $k8 = 0.932378647316581$, and $k9 = 0.232373784321657$. The test images are $512 \times 512$ pixels with 8 bit gray level, and the simulation results are shown in Figure 5. In following subsections, many commonly used security analyses will be discussed.

### 3.1. Key Space Analysis.

The key space of a cryptosystem is the very important factor on security when brute-force attack is happening. In our scheme, the cryptosystem needs input k1, k2, k3, k4, k5, k6, k7, k8, and k9 $\in\in (0, 1)$ as security key. If we supposed the change step of each inputted key are $10^{-15}$, then the total key space is calculated as $S = (10^5)^9 \approx 2^{455}$, which is larger enough on security [4–6].

### 3.2. Differential Attack.

To resist differential attack, an image cryptosystem is required to have a good plaintext sensitivity. If the inputted image has only one-bit changed, the corresponding encrypted image is totally different, then the cryptosystem is considered to have enough plaintext sensitivity to resist differential attack. To measure the difference between two image, NPCR (number of pixels change rate) and UACI (unified average changing intensity) are put forward [8–11].

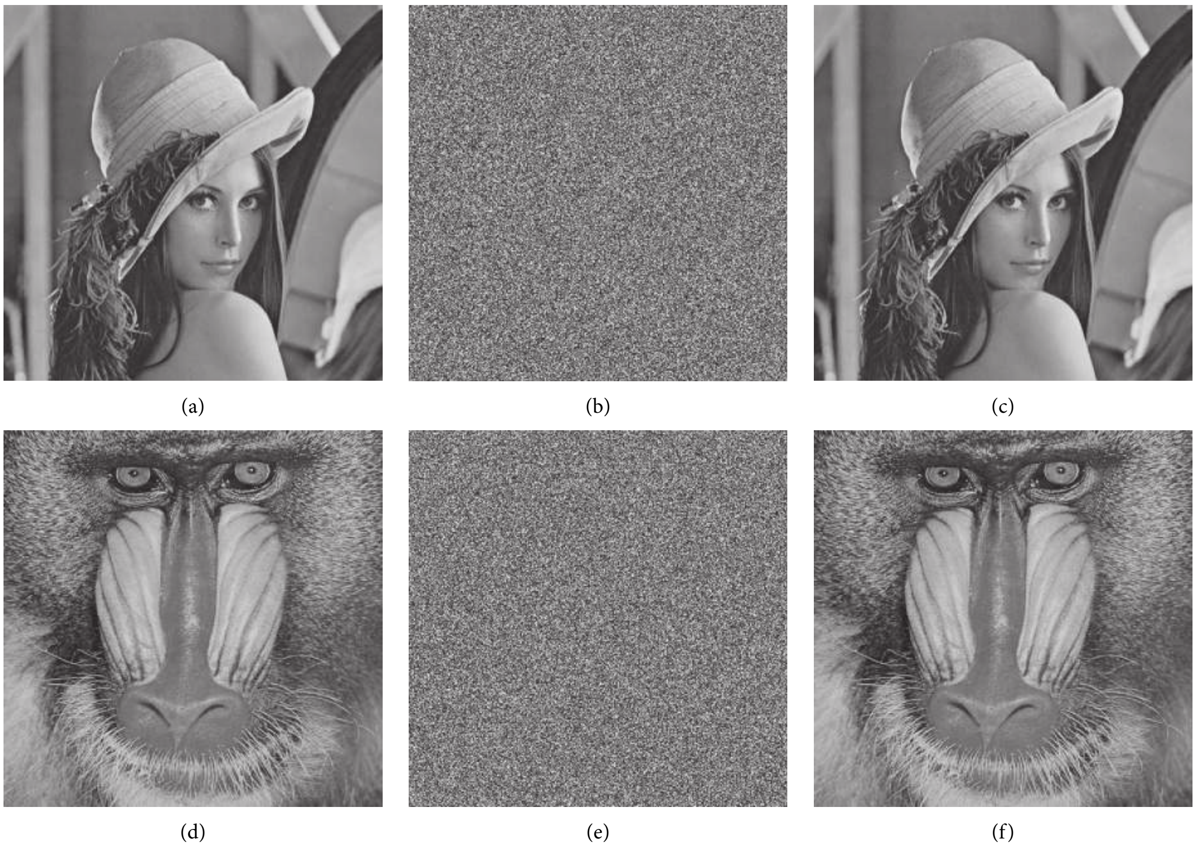


Figure 5: The simulation results of encryption and decryption: (a, d) plain images; (b, e) cipher images; (c, f) decrypted images.

The NPCR and UACI are given by

$$\mathrm{NPCR} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} D(i,j)}{M\times N}\times 100\%,$$
$$D(i,j) = \begin{cases} 0, & C_1(i,j) = C_2(i,j), \\ 1, & C_1(i,j) \neq C_2(i,j), \end{cases}$$
$$\mathrm{UACI} = \frac{1}{M\times N}\left(\sum_{i=1}^{M}\sum_{j=1}^{N}\frac{\left|C_1(i,j) - C_2(i,j)\right|}{255}\right)\times 100\%, \tag{43}$$

where $C_1(i,j)$ and $C_2(i,j)$ represent two cipher-images obtained from encrypting two one-pixel different images. $M$ and N represent the height and width of images, respectively.

The NPCR means the percentage of different pixels at same position between two corresponding encrypted images which are obtained by two images with one-bit difference, and the UACI represents the average intensity of the difference between two same position's pixels which are obtained from two cipher images.

In order to evaluate whether the NPCR and UACI images passed the tests, the critical values of NPCR and UACI are defined [54, 55]. For a significance level $\alpha$, a critical NPCR score $\mathcal{N}_\alpha^*$ is obtained by

$$\mathcal{N}_\alpha^* = \frac{Q - \Phi^{-1}(\alpha)\sqrt{Q/H}}{Q+1}, \tag{44}$$

where $H$ is the total number of pixels in an image and $Q$ represents the largest allowed pixel value in the image. An image encryption scheme can be considered to pass the NPCR if the obtained NPCR is larger than $\mathcal{N}_\alpha^*$.

The critical UACI interval $(\mathcal{U}_\alpha^{*-}, \mathcal{U}_\alpha^{*+})$ can be calculated by

$$\begin{cases} \mathcal{U}_\alpha^{*-} = \mu_{\mathcal{U}} - \Phi^{-1}\left(\dfrac{\alpha}{2}\right)\sigma_{\mathcal{U}}, \\ \mathcal{U}_\alpha^{*+} = \mu_{\mathcal{U}} + \Phi^{-1}\left(\dfrac{\alpha}{2}\right)\sigma_{\mathcal{U}}, \end{cases} \tag{45}$$

where

$$\mu_{\mathcal{U}} = \frac{Q+2}{3Q+3},$$
$$\sigma_{\mathcal{U}} = \sqrt{\frac{(Q+2)\left(Q^2+2Q+3\right)}{18(Q+1)^2QH}}. \tag{46}$$

If the obtained UACI falls into range $(\mathcal{U}_\alpha^{*-}, \mathcal{U}_\alpha^{*+})$, the corresponding encryption algorithm is considered to have a high security level. For images with size $512\times 512$, when we set the significance level $\alpha = 0.05$, the critical value of NPCR is $\mathcal{N}^*_{0.05} = 99.5893\%$ and $(\mathcal{U}^{*-}_{0.05}, \mathcal{U}^{*+}_{0.05}) = (33.3730\%, 33.5541\%)$.

In this section, the NPCR and UACI are calculated by two cipher images obtained by encrypting two images, the one obtained by random changing one pixel of another. We tested NPCR and UACI 100 times, and the results are shown in Table 1. According to the results in Table 1, it is no doubt that our cryptosystem is plaintext sensitive enough on resisting the differential attack.

### 3.3. Statistical Analysis

*3.3.1. Histogram Analysis.* If the histogram of the corresponding cipher image shows uniform distribution, the cryptosystem has good ability on resisting the statistical attack. In other words, there is no obvious statistical distinction of the count of pixels in each gray level. The distribution of plain images' histograms and cipher images' histograms is shown in Figure 6. We use the chi-squared test to evaluate the uniformity of encrypted image's histogram [43, 56]. When we set the significance level $\alpha = 0.05$, the chi-squared test results of cipher images is given in Table 2. The histogram analysis result shows that our proposed cryptosystem has good diffused property on resisting the statistical attack.

*3.3.2. Correlation Coefficient.* As we all know, the adjacent pixels in plain image have a strong correlation. However, there is less correlation among adjacent pixels in cipher image [5–10].

The correlation coefficient is given by

$$r_{ab} = \frac{\operatorname{cov}(a,b)}{\sqrt{D(a)D(b)}}, \tag{47}$$

where a and $b$ are two adjacent pixels gray values, and

$$E(a) = \frac{1}{N}\sum_{i=1}^{N} a_i,$$

$$D(a) = \frac{1}{N}\sum_{i=1}^{N} (a_i - E(a))^2, \tag{48}$$

$$\operatorname{cov}(a,b) = \frac{1}{N}\sum_{i=1}^{N} (a_i - E(a))(b_i - E(b)).$$

In this test, 10000 pairs of adjacent pixels are randomly selected for calculating the correlation coefficient. The correlation coefficient results are shown in Table 2, and the correlation distribution of image “Lena” is shown in Figure 7 and Table 3

*3.4. Key Sensitivity Analysis.* The key sensitivity analysis is considered to be a measurement of the diffusion property of the cryptosystem. In our proposed scheme, there are nine inputted keys: k1, k2, k3, k4, k5, k6, k7, k8, and k9 $\in\in (0,1)$. In this test, the plain image ‘Lena’ is first time encrypted with initial keys which are assigned at the beginning of Section 3, the second time encrypted with modifying $k1' = k1 + 10^{-15}$, and the third time encrypted with modifying $k2' = k2 + 10^{-15}$. And then, the differences between the ciphers encrypted by modified keys and initial keys are figured. The test result is shown in Figure 8, and the NPCR and UACI between ciphers generated by different keys are shown in Table 4. The result of test clearly shows that our proposed cryptosystem has good key sensitivity on resisting the exhaustive attack.

### 3.5. Information Entropy Analysis

*3.5.1. Global Information Entropy.* The global information entropy is used to indicate the uncertainty degree of image information [9, 57]. The global information entropy can be calculated by

$$H(s) = \sum_{i=0}^{2^K-1} P(s_i)\log_2\frac{1}{Ps_i}, \tag{49}$$

where $K$ is the image bit depth, e.g., $K=8$, for an 8 bit gray image, and $P(s_i)$ means the probability of $s_i$. The ideal case of 8 bit gray image information entropy is $H(s) = 8$ bits. The entropy test result is shown in Table 5. According to the test results, our entropies are very close to the ideal value 8. Thus, our cryptosystem shows good performance on resisting the entropy attack.

*3.5.2. Local Shannon Entropy.* Global Shannon entropy reflects the total randomness of image, and it has certain limitations in some occasions. Therefore, local Shannon entropy (LSE) was proposed by Wu et al. [58, 59]. To measure local entropy, we first randomly select $k$ non-overlapping image blocks $B_1, B_2, \ldots, B_k$ with $T_B$ pixels from image $\boldsymbol{I}$; then, the LSE is given by

$$\overline{H_{k,T_B}}(I) = \sum_{i=1}^{k}\frac{H(B_i)}{k}, \tag{50}$$

where $H(B_i)$ is the Shannon entropy of image block$B_i$ and can be calculated by equation (49).

For our tests, we select parameters $(k, T_B) = (30, 1936)$, and then the ideal value of LSE is 7.902469317. The significance $\alpha = 0.05$, and we consider the tests passed when the test LSE values fall into (7.901901305, 7.903037329). The LSE test results are shown in Table 6.

*3.6. NIST SP800-22 Tests.* In our scheme, control sequences are generating by the hyperchaotic Lorenz system (HCLS) and skew tent map (STM). Therefore, the randomness of iterative sequences which generated by these two chaotic system is directly affecting the performance of the image cryptosystem. Here, we use SP800-22 [60] to test the randomness of binary sequences generated by the hyperchaotic Lorenz system and skew tent map. The test results are shown in Table 7.

Table 1: The results of NPCR and UACI.

| Image | NPCR (%) | | | | | UACI (%) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Min | Max | Average | Std. | $\mathcal{N}^*_{0.05}$ 99.5893 | Min | Max | Average | $\mathcal{U}^{*+}_{0.05}$ = 33.5541 |
| Lena | 99.5935 | 99.6326 | 99.6161 | 0.0123 | Pass | 33.3918 | 33.5717 | 33.4709 | Pass |
| Baboon | 99.5943 | 99.6342 | 99.6109 | 0.0129 | Pass | 33.3932 | 33.5913 | 33.4792 | Pass |
| Pepper | 99.5912 | 99.6391 | 99.6090 | 0.0121 | Pass | 33.3985 | 33.5574 | 33.4664 | Pass |
| Airplane | 99.5962 | 99.6323 | 99.6095 | 0.0131 | Pass | 33.3805 | 33.5353 | 33.4593 | Pass |
| Boat | 99.5931 | 99.6399 | 99.6101 | 0.0147 | Pass | 33.3746 | 33.5546 | 33.4654 | Pass |

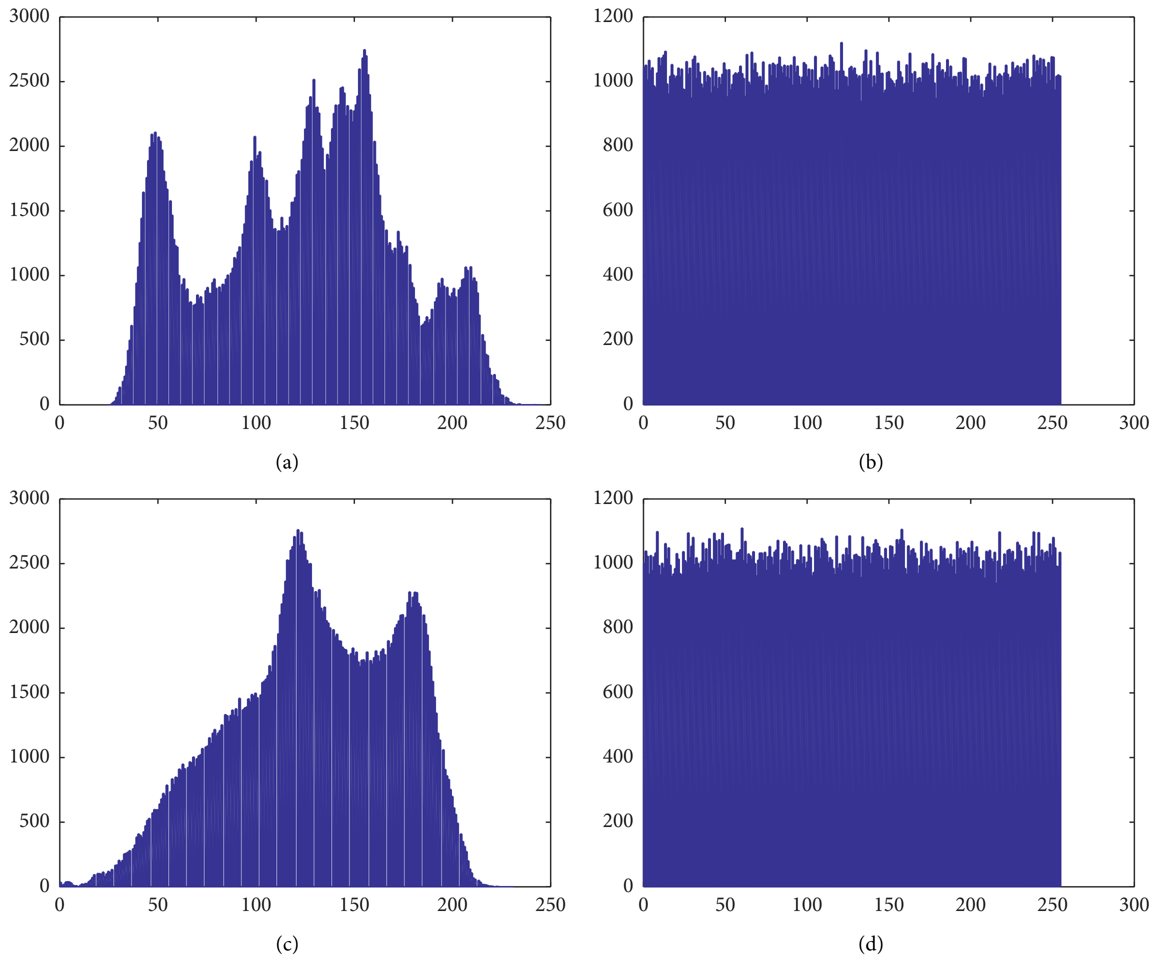
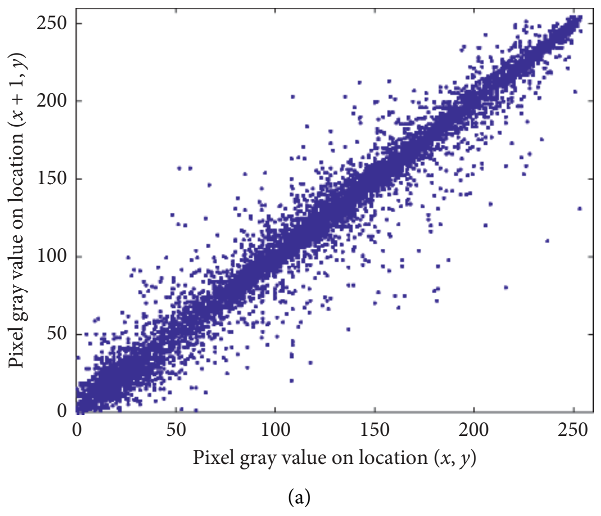


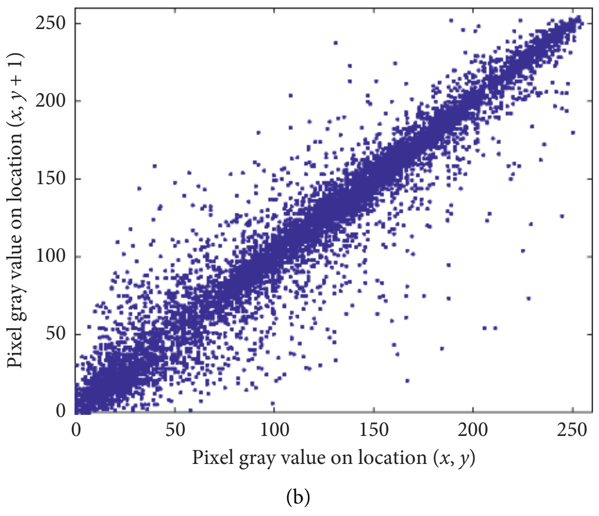
Figure 6: Histograms of (a, c) plain images ”Lena” and “Baboon” and (b, d) their corresponding cipher images.

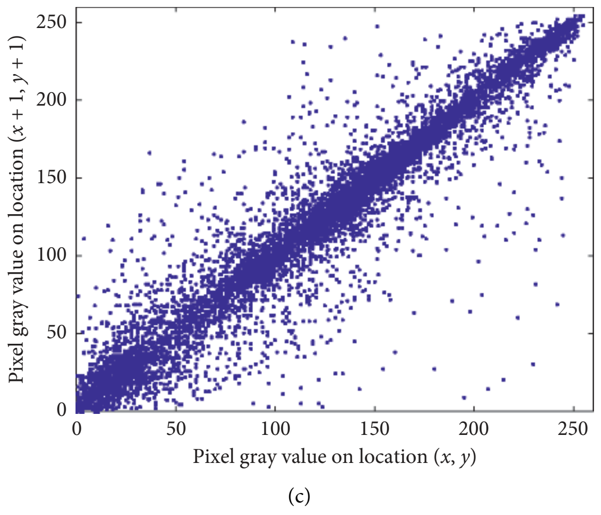
Table 2: Histogram uniformity evaluation by the chi-squared test.

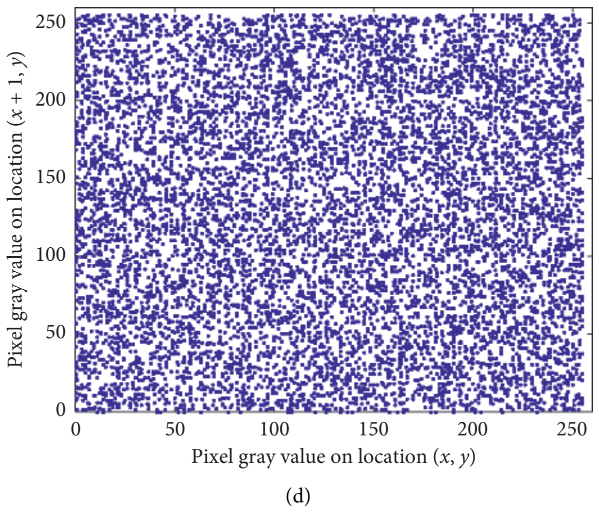
| Images | $P$ value | Decision ($H = 0$ or 1) $\alpha = 0.05$ |
|---|---|---|
| Lena | 0.1888141 | 0; Accept |
| Baboon | 0.1117246 | 0; Accept |

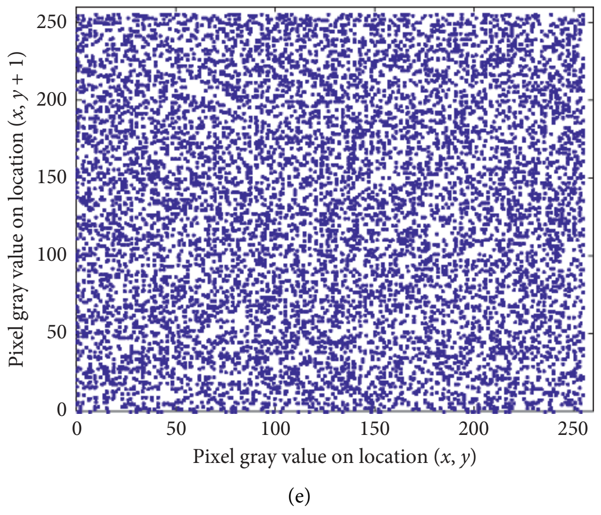
*3.7. Performance Analysis and Comparisons.* The simulation and tests are implemented in MATLAB R2015b, which is worked on a MacBook with Inter (R) Core i7, CPU 1.4 GHz, and 16 GB memory, and the software running system is macOS (High Sierra 10.13.1). As we all known, there are many factors affecting the speed of encryption and decryption except algorithm itself, such as operating system, hardware environment, programming language, and code optimization. Thus, in this performance comparison, we not only give the comparison of time cost but also give the comparison of other correlation factors. Furthermore, we also give the comparison of encryption throughput and











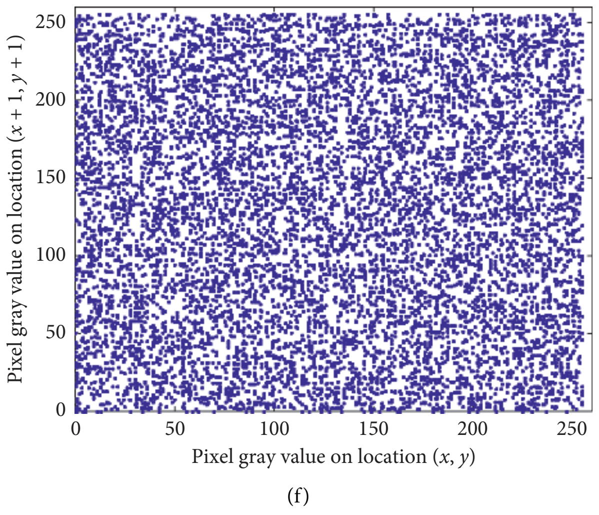


FIGURE 7: Correlation distributions of the (ac) plain image and (df) cipher image. (a) Figure 4(d) in the horizontal direction, (b) Figure 4(e) in the vertical direction, (c) Figure 4(f) in the diagonal direction.

Table 3: Correlation coefficients.

| Images | Horizontal | | Vertical | | Diagonal | |
|---|---|---|---|---|---|---|
| | Plain | Cipher | Plain | Cipher | Plain | Cipher |
| Lena | 0.9845 | −0.0013 | 0.9699 | 0.0005 | 0.9585 | 0.0016 |
| Baboon | 0.7608 | 0.0032 | 0.8618 | 0.0012 | 0.7238 | −0.0004 |

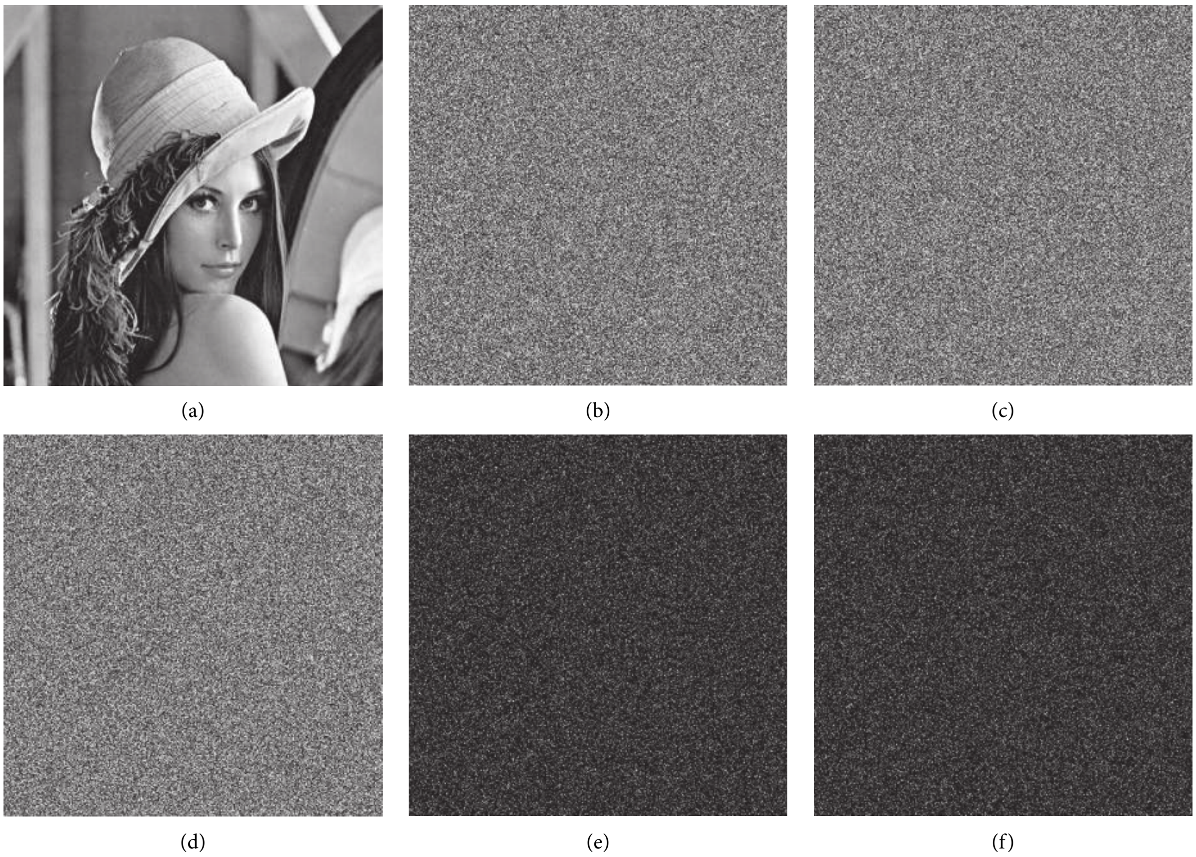


Figure 8: Key sensitivity analysis in encryption. (a) Original image. (b) Encrypted image with initial keys. (c) Encrypted image with modifying $k1' = k1 + 10^{-15}$. (d) Cipher image with modifying $k2' = k2 + 10^{-15}$. (e) Image of |b-c|. (f) image of |b-d|.

Table 4: Results of NPCR and UACI between ciphers.

| Ciphers | NPCR (%) | $\mathcal{N}^{*}_{0.05}$ 99.5893 | UACI (%) | $\mathcal{U}^{*-}_{0.05} = 33.3730$ $\mathcal{U}^{*+}_{0.05} = 33.5541$ |
|---|---|---|---|---|
| Figures 8(b) and 8(c) | 99.6063 | Pass | 33.4369 | Pass |
| Figures 8(b) and 8(d) | 99.6120 | Pass | 33.4494 | Pass |

Table 5: The global entropy test results.

| Images | Lena | Baboon | Pepper | Airplane | Boat |
|---|---|---|---|---|---|
| Entropy | 7.9993420 | 7.9993005 | 7.9993824 | 7.9993269 | 7.9994635 |

Table 6: The local Shannon entropy test results.

| Images | Lena | Baboon | Pepper | Airplane | Boat |
|---|---|---|---|---|---|
| LSE | 7.90278317 | 7.90294534 | 7.90219342 | 7.90265212 | 7.90301576 |
| (7.901901305, 7.903037329) | Pass | Pass | Pass | Pass | Pass |

Table 7: NIST SP800-22 test results.

| Test items | HCLS $P$ value | Result ≥0.01 | STM $P$ value | Result ≥0.01 |
|---|---|---|---|---|
| Frequency | 0.43304 | Pass | 0.32631 | Pass |
| Block frequency | 0.93008 | Pass | 0.88263 | Pass |
| Runs | 0.86392 | Pass | 0.75258 | Pass |
| Longest runs | 0.97571 | Pass | 0.89365 | Pass |
| Binary matrix rank | 0.50984 | Pass | 0.71245 | Pass |
| Discrete Fourier transform | 0.16365 | Pass | 0.32123 | Pass |
| Nonoverlapping template | 0.28953 | Pass | 0.12545 | Pass |
| Overlapping template | 0.57548 | Pass | 0.36585 | Pass |
| Maurer's universal statistical | 0.38361 | Pass | 0.45852 | Pass |
| Linear complexity | 0.63612 | Pass | 0.23562 | Pass |
| Serial* | 0.13252 | Pass | 0.31142 | Pass |
| Approximate entropy | 0.83427 | Pass | 0.63253 | Pass |
| Cumulative sums* | 0.99253 | Pass | 0.85652 | Pass |
| Random excursions* | 0.71533 | Pass | 0.35262 | Pass |
| Random excursions variant* | 0.56579 | Pass | 0.45253 | Pass |

*The average values of multiple tests.

Table 8: Speed comparison.

| Algorithms | CPU speed (GHz) | Memory size (GB) | Program language | Encryption time (ms) (512 × 512) | Encryption throughput (MBps) | Number of cycles |
|---|---|---|---|---|---|---|
| Our work | 1.4 | 16 | Matlab | 138 | 1.811 | 1682 |
| Ref [61] | 2.5 | 4 | Matlab | 613 | 0.408 | 5846 |
| Ref [42] | 3.0 | 8 | C | 139 | 1.799 | 1591 |
| Ref [62] | 2.7 | 16 | C | 166 | 1.506 | 1710 |
| Ref [28] | 3.0 | 8 | C | 390 | 0.641 | 4463 |
| Ref [6] | 2.6 | 8 | Matlab | 484 | 0.517 | 4800 |
| Ref [63] | 2.4 | 8 | C | 203 | 1.232 | 1859 |
| Ref [41] | 2.4 | 8 | C | 85 | 2.941 | 778 |

Table 9: Security comparison.

| Algorithms | Cipher correlation coefficients | | | Global entropy | Local entropy | Key space | Plaintext sensitivity | |
|---|---|---|---|---|---|---|---|---|
| | Horizontal | Vertical | Diagonal | | | | NPCR (%) | UACI (%) |
| Our work | −0.0013 | 0.0005 | 0.0016 | 7.9993 | 7.902783 | $2^{455}$ | 99.61 | 33.47 |
| Ref. [28] | 0.0010 | 0.0054 | 0.0056 | 7.9992 | - | $2^{520}$ | 99.60 | 33.50 |
| Ref. [15] | −0.0331 | 0.0057 | 0.0169 | 7.9972 | - | $2^{90}$ | 99.61 | 33.40 |
| Ref. [10] | 0.0064 | 0.0078 | 0.0029 | 7.9993 | - | $10^{15} \times 2^{256}$ | 99.59 | 33.46 |
| Ref. [18] | 0.0026 | 0.0019 | 0.0028 | 7.9992 | 7.902552 | $2^{640}$ | 99.60 | 33.48 |
| Ref. [40] | −0.02887 | 0.01459 | 0.03658 | 7.9993 | - | $2^{582}$ | 99.62 | 33.41 |
| Ref. [41] | 0.001987 | 0.00449 | −0.0087 | 7.9993 | 7.901558 | $2^{572}$ | 99.63 | 33.48 |
| Ref. [42] | 0.0119 | 0.0092 | 0.0013 | 7.9993 | - | $10^{30} \times 2^{384}$ | 99.64 | 33.61 |
| Ref. [44] | 0.003 | 0.0018 | 0.0006 | 7.997 | 7.9024 | $10^{112}$ | 99.62 | 33.44 |
| Ref. [45] | 0.00193 | 0.0018 | 0.00342 | 7.998 | - | $10^{238}$ | 99.68 | 33.47 |
| Ref. [46] | −0.0017 | −0.0084 | −0.0019 | 7.9975 | 7.9051 | $2^{312}$ | 99.62 | 33.50 |
| Ref. [47] | −0.000026 | −0.00031 | −0.021 | - | 7.9028 | - | 99.96 | 33.69 |
| Ref. [49] | 0.004851 | −0.0020 | −0.00270 | 7.9993 | 7.902481 | $2^{250}$ | 99.565 | 33.450 |
| Ref. [50] | 0.00058 | 0.00061 | −0.0003 | 7.9974 | - | $2^{532}$ | 99.60 | 33.49 |

number of cycles to make a more intuitive performance description. The speed analysis result is shown in Table 8, and the security comparisons with other works are shown in Table 9. The speed test and the comparisons show that our proposed cryptosystem not only has good speed on encryption but also has good security performance.

## 4. Conclusions

We proposed an efficient chaos-based image encryption scheme using the jigsaw method which contains two operations of image blocks: revolving and shifting. In the proposed encryption scheme, the preprocessing makes sure our scheme has enough plain image sensitivity on resisting differential attack. The encryption process guarantees our cryptosystem has full diffusion on resisting the statistical analysis. The postprocessing process further increases the diffusion characteristic. In addition, some common security analyses and speed comparisons show that our image cryptosystem has both security and speed performance.

## Data Availability

The data used to support the findings of this study are included within the article.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

Our research was supported by the National Natural Science Foundation of China (U1836205, 61662009, and 61772008), the open Foundation of Guizhou Provincial Key Laboratory of Public Big Data (2017BDKFJJ023 and 2017BDKFJJ026), the Science and Technology Foundation of Guizhou (Guizhou Science-Contract-Major-Program [2018]3001, Guizhou-Science-Contract-Major-Program [2018]3007, Guizhou-Science-Contract-Major-Program [2017]3002, Guizhou-Science-Contract-Support [2019]2004, Guizhou-Science-Contract-Support [2018]2162, Guizhou-Science-Contract-Support [2018]2159, Guizhou-Science-Contract [2017]1045, Guizhou-Science-Contract 1049, and Guizhou-Science-Contract [2019]1249), and the Scientific Research Foundation of Guizhou province, China (QKHPTRC[2017] 5788). The work was also supported by the cultivation project of Guizhou University ([2019] 56), the Project of Innovative Group in Guizhou Education Department ([2013]09), and the Youth Science and Technology Talents Growth Project of the Guizhou Provincial Department of Education (Guizhou-Education-Contract-KY-Word [2018] 260).

## References

[1] Y. Zhang, "Test and verification of AES used for image encryption," *3D Research*, vol. 9, no. 1, 2018.

[2] E. Lorenz, "Deterministic non-period flow," *Journal of the Atmospheric Sciences*, vol. 20, no. 3, pp. 130–141, 1963.

[3] R. Matthews, "On the derivation of a "chaotic" encryption algorithm," *Cryptologia*, vol. 13, no. 1, pp. 29–42, 1989.

[4] G. Hu, D. Xiao, Y. Zhang, and T. Xiang, "An efficient chaotic image cipher with dynamic lookup table driven bit-level permutation strategy," *Nonlinear Dynamics*, vol. 87, no. 2, pp. 1359–1375, 2016.

[5] T. Hu, Y. Liu, L.-H. Gong, and C.-J. Ouyang, "An image encryption scheme combining chaos with cycle operation for DNA sequences," *Nonlinear Dynamics*, vol. 87, no. 1, pp. 51–66, 2016.

[6] Z. Hua, F. Jin, B. Xu, and H. Huang, "2D Logistic-Sine-coupling map for image encryption," *Signal Processing*, vol. 149, pp. 148–161, 2018.

[7] C. Cao, K. Sun, and W. Liu, "A novel bit-level image encryption algorithm based on 2D-LICM hyperchaotic map," *Signal Processing*, vol. 143, pp. 122–133, 2018.

[8] X. Chai, X. Zheng, Z. Gan, D. Han, and Y. Chen, "An image encryption algorithm based on chaotic system and compressive sensing," *Signal Processing*, vol. 148, pp. 124–144, 2018.

[9] G. Ye, H. Zhao, and H. Chai, "Chaotic image encryption algorithm using wave-line permutation and block diffusion," *Nonlinear Dynamics*, vol. 83, no. 4, pp. 2067–2077, 2015.

[10] Y. Luo, L. Cao, S. Qiu, H. Lin, J. Harkin, and J. Liu, "A chaotic map-control-based and the plain image-related cryptosystem," *Nonlinear Dynamics*, vol. 83, no. 4, pp. 2293–2310, 2015.

[11] Z. Hua, S. Yi, and Y. Zhou, "Medical image encryption using high-speed scrambling and pixel adaptive diffusion," *Signal Processing*, vol. 144, pp. 134–144, 2018.

[12] F. Khelifi, T. Brahimi, J. Han, and X. Li, "Secure and privacy-preserving data sharing in the cloud based on lossless image coding," *Signal Processing*, vol. 148, pp. 91–101, 2018.

[13] X. Wang and H. L. Zhang, "A novel image encryption algorithm based on genetic recombination and hyper-chaotic systems," *Nonlinear Dynamics*, vol. 83, no. 1-2, pp. 333–346, 2015.

[14] X. Wang, C. Liu, and H. Zhang, "An effective and fast image encryption algorithm based on Chaos and interweaving of ranks," *Nonlinear Dynamics*, vol. 84, no. 3, pp. 1595–1607, 2016.

[15] X. Wang, Q. Wang, and Y. Zhang, "A fast image algorithm based on rows and columns switch," *Nonlinear Dynamics*, vol. 79, no. 2, pp. 1141–1149, 2014.

[16] D.-d. Liu, W. Zhang, H. Yu, and Z.-l. Zhu, "An image encryption scheme using self-adaptive selective permutation and inter-intra-block feedback diffusion," *Signal Processing*, vol. 151, pp. 130–143, 2018.

[17] X. Wu, K. Wang, X. Wang, H. Kan, and J. Kurths, "Color image DNA encryption using NCA map-based CML and one-time keys," *Signal Processing*, vol. 148, pp. 272–287, 2018.

[18] X. Wang, C. Liu, D. Xu, and C. Liu, "Image encryption scheme using chaos and simulated annealing algorithm," *Nonlinear Dynamics*, vol. 84, no. 3, pp. 1417–1429, 2016.

[19] A. Akhshani, A. Akhavan, S.-C. Lim, and Z. Hassan, "An image encryption scheme based on quantum logistic map," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 12, pp. 4653–4661, 2012.

[20] A. Kanso, "Self-shrinking chaotic stream ciphers," *Communications in Nonlinear Science and Numerical Simulation*, vol. 16, no. 2, pp. 822–836, 2011.

[21] A. Akhavan, A. Samsudin, and A. Akhshani, "A symmetric image encryption scheme based on combination of nonlinear chaotic maps," *Journal of the Franklin Institute*, vol. 348, no. 8, pp. 1797–1813, 2011.

[22] L. Zhang, X. Liao, and X. Wang, "An image encryption approach based on chaotic maps," *Chaos, Solitons & Fractals*, vol. 24, no. 3, pp. 759–765, 2005.

[23] W. Zhang, K.-W. Wong, H. Yu, and Z.-L. Zhu, "An image encryption scheme using reverse 2-dimensional chaotic map and dependent diffusion," *Communications in Nonlinear Science and Numerical Simulation*, vol. 18, no. 8, pp. 2066–2080, 2013.

[24] G. Alvarez and S. Li, "Some basic cryptographic requirements for chaos-based cryptosystems," *International Journal of Bifurcation and Chaos*, vol. 16, no. 08, pp. 2129–2151, 2006.

[25] A. A. A. El-Latif, L. Li, N. Wang, Q. Han, and X. Niu, "A new approach to chaotic image encryption based on quantum

chaotic system, exploiting color spaces," *Signal Process*, vol. 93, no. 11, pp. 2986–3000, 2013.

[26] J. Fridrich, "Symmetric ciphers based on two-dimensional chaotic maps," *International Journal of Bifurcation and Chaos*, vol. 08, no. 06, pp. 1259–1284, 1998.

[27] S. Lian, J. Sun, and Z. Wang, "A block cipher based on a suitable use of the chaotic standard map," *Chaos, Solitons & Fractals*, vol. 26, no. 1, pp. 117–129, 2005.

[28] A. Souyah and K. M. Faraoun, "An image encryption scheme combining chaos-memory cellular automata and weighted histogram," *Nonlinear Dynamics*, vol. 86, no. 1, pp. 639–653, 2016.

[29] Z. Li, C. Peng, L. Li, and X. Zhu, "A novel plaintext-related image encryption scheme using hyper-chaotic system," *Nonlinear Dynamics*, vol. 94, no. 2, pp. 1319–1333, 2018.

[30] J. Wu, X. Liao, and B. Yang, "Image encryption using 2D Hénon-Sine map and DNA approach," *Signal Processing*, vol. 153, pp. 11–23, 2018.

[31] H. S. Ye, N. R. Zhou, and L. H. Gong, "Multi-image compression-encryption scheme based on quaternion discrete fractional Hartley transform and improved pixel adaptive diffusion," *Signal Processing*, vol. 175, 2020.

[32] L. Y. Zhu, H. S. Song, X. Zhang et al., "A robust meaningful image encryption scheme based on block compressive sensing and SVD embedding," *Signal Processing*, vol. 175, 2020.

[33] A. B. Joshi, D. Kumar, A. Gaffar, and D. C. Mishra, "Triple color image encryption based on 2D multiple parameter fractional discrete Fourier transform and 3D Arnold transform," *Optics and Lasers in Engineering*, vol. 133, 2020.

[34] Y. Lin, E. Yang, and G. Y. Wang, "A novel parallel image encryption algorithm based on hybrid chaotic maps with OpenCL implementation," *Soft Computing*, vol. 24, pp. 12413–12427, 2020.

[35] D. Ouyang, J. Shao, H. Jiang, S. K. Nguang, and H. T. Shen, "Impulsive synchronization of coupled delayed neural networks with actuator saturation and its application to image encryption," *Neural Networks*, vol. 128, pp. 158–171, 2020.

[36] W. Y. Wen, Y. K. Hong, Y. M. Fang, M. Li, and M. Li, "A visually secure image encryption scheme based on semi-tensor product compressed sensing," *Signal Processing*, vol. 173, 2020.

[37] G. D. Ye, P. Chen, X. D. You, S. Yang, and X. L. Huang, "Image encryption and hiding algorithm based on compressive sensing and random numbers insertion," *Signal Processing*, vol. 172, 2020.

[38] Y. Zhang, A. Chen, Y. Tang, J. Dang, and G. Wang, "Plaintext-related image encryption algorithm based on perceptron-like network," *Information Sciences*, vol. 526, pp. 180–202, 2020.

[39] Z. Li, C. Peng, W. Tan et al., "A novel chaos-based color image encryption scheme using bit-level permutation," *Symmetry*, vol. 121497 pages, 2020.

[40] E. Yavuz, R. Yazıcı, M. C. Kasapbaşı, and E. Yamaç, "A chaos-based image encryption algorithm with simple logical functions," *Computers & Electrical Engineering*, vol. 54, pp. 471–483, 2016.

[41] E. Yavuz, "A novel chaotic image encryption algorithm based on content-sensitive dynamic function switching scheme," *Optics & Laser Technology*, vol. 114, pp. 224–239, 2019.

[42] S. Amina and F. K. Mohamed, "An efficient and secure chaotic cipher algorithm for image content preservation," *Communications in Nonlinear Science and Numerical Simulation*, vol. 60, pp. 12–32, 2018.

[43] D. Ravichandran, P. Praveenkumar, J. B. Balaguru Rayappan, and R. Amirtharajan, "Chaos based crossover and mutation for securing DICOM image," *Computers in Biology and Medicine*, vol. 72, pp. 170–184, 2016.

[44] C. Lakshmi, K. Thenmozhi, J. B. B. Rayappan, and R. Amirtharajan, "Hopfield attractor-trusted neural network: an attack-resistant image encryption," *Neural Computing and Applications*, vol. 32, no. 15, pp. 11477–11489, 2020.

[45] S. A. Banu and R. Amirtharajan, "A robust medical image encryption in dual domain: chaos-dna-iwt combined approach," *Medical & Biological Engineering & Computing*, vol. 58, pp. 1–14, 2020.

[46] M. Alawida, A. Samsudin, J. S. Teh, and R. S. Alkhawaldeh, "A new hybrid digital chaotic system with applications in image encryption," *Signal Processing*, vol. 160, pp. 45–58, 2019.

[47] M. Alawida, J. S. Teh, A. Samsudin, and W. H. Alshoura, "An image encryption scheme based on hybridizing digital chaos and finite state machine," *Signal Processing*, vol. 164, pp. 249–266, 2019.

[48] J. A. P. Artiles, D. P. B. Chaves, and C. Pimentel, "Image encryption using block cipher and chaotic sequences," *Signal Processing: Image Communication*, vol. 79, pp. 24–31, 2019.

[49] X. Wang, H. Zhao, L. Feng, X. Ye, and H. Zhang, "High-sensitivity image encryption algorithm with random diffusion based on dynamic-coupled map lattices," *Optics and Lasers in Engineering*, vol. 122, pp. 225–238, 2019.

[50] Y. Luo, J. Yu, W. Lai, and L. Liu, "A novel chaotic image encryption algorithm based on improved baker map and logistic map," *Multimedia Tools and Applications*, vol. 78, no. 15, pp. 22023–22043, 2019.

[51] X. Wang and M. Wang, "A hyperchaos generated from Lorenz system," *Physica A: Statistical Mechanics and Its Applications*, vol. 387, no. 14, pp. 3751–3758, 2008.

[52] T. Xiang, X. Liao, and K.-W. Wong, "An improved particle swarm optimization algorithm combined with piecewise linear chaotic map," *Applied Mathematics and Computation*, vol. 190, no. 2, pp. 1637–1645, 2007.

[53] J. S. Teh, M. Alawida, and Y. C. Sii, "Implementation and practical problems of chaos-based cryptography revisited," *Journal of Information Security and Applications*, vol. 50102421 pages, 2020.

[54] R. Sivaraman, R. Sundararaman, J. B. B. Rayappan et al., "Ring Oscillator as Confusion–Diffusion Agent: A Complete TRNG Drove Image Security," *IET Image Processing*, vol. 14, 2020.

[55] Y. Wu, J. P. Noonan, and S. Agaian, "NPCR and UACI randomness tests for image encryption," *Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT)*, vol. 1, no. 2, pp. 31–38, 2011.

[56] Z. Li, C. Peng, W. Tan, and L. Li, "A novel chaos-based image encryption scheme by using randomly DNA encode and plaintext related permutation," *Applied Sciences*, vol. 10, no. 21, p. 7469, 2020.

[57] Z. Li, C. Peng, W. Tan et al., "An efficient plaintext-related chaotic image encryption scheme based on compressive sensing," *Sensors*, vol. 21, no. 758, 2021.

[58] N. Chidambaram, P. Raj, T. Karruppuswamy et al., "An advanced framework for highly secure and cloud-based storage of colour images," *IET Image Processing*, vol. 14, no. 13, pp. 3143–3153, 2020.

[59] Y. Wu, Y. Zhou, G. Saveriades, S. Agaian, J. P. Noonan, and P. Natarajan, "Local Shannon entropy measure with statistical tests for image randomness," *Information Sciences*, vol. 222, pp. 323–342, 2013.

[60] A. Rukhin, J. Soto, J. Nechvatal et al., *A Statistical Test Suite for Random and Pseudorandom Number Generators for*

*Cryptographic Application*, pp. 800–822, NIST Special Publication, Gaithersburg, MD, USA, 2001.

[61] L. Xu, X. Gou, Z. Li, and J. Li, "A novel chaotic image encryption algorithm using block scrambling and dynamic index based diffusion," *Optics and Lasers in Engineering*, vol. 91, pp. 41–52, 2017.

[62] L. L. Palacios, G. G. Delgado, J. A. Díaz et al., "Symmetric cryptosystem based on skew tent map," *Multimedia Tools and Applications*, vol. 77, no. 2, pp. 2739–2770, 2018.

[63] E. Yavuz, R. Yazıcı, M. C. Kasapbaşi et al., "Enhanced chaotic key-based algorithm for low-entropy image encryption," in *Proceedings of the 2014 22nd Signal Processing And Communications Applications Conference (SIU)*, pp. 385–388, IEEE, Trabzon, Turkey, October 2014.