Bit-Level Multi-Image Encryption Algorithm Based on Composite Chaotic System

Yilin Han, Ye Tao, Shanshan Wang, Wenhua Cui, and Yuting Wang

Abstract—**To enhance image encryption efficiency, this paper proposes a bit-level multi-image encryption algorithm based on composite chaotic systems. It necessitates merely a single encryption operation to encrypt an arbitrary number of images, with the computational expense being on par with that of encrypting just one image. Firstly, utilizing Sine and Tent mappings, we introduce an innovative composite chaotic system termed 1D-SATM. The new composite chaotic system has a larger chaotic space and better randomness, than the original two one-dimensional chaotic maps. When encrypting, the first step is to overlay multiple color images to form a single overlay image. Secondly, the color overlay image is segmented into a three-channel matrix. Then the Binary Bit Plane Decomposition (BBD) technique is used to decompose each channel into 8-bit planes, which are subsequently divided into two groups. Again, perform the right cyclic shift and XOR operations on the two groups of bit planes for diffusion. Perform XOR operations on the chaotic sequence generated by 1D-SATM with the diffusion result again, to further obfuscate the data. Finally, the optimized Josephus permutation algorithm is applied, to produce an encrypted color composite image. The experimental results and algorithm analysis indicate that this algorithm has good encryption effectiveness, high security, and fast running speed.**

Index Terms—**Composite chaotic system, Image encryption, Bit plane, Scrambling, diffusion**

I. Introduction

IMAGE encryption is one of the most important methods for providing security and ensuring the confidentiality of image information [1]. Image encryption is a technique that uses mathematical transformations to obfuscate the pixel positions of ordinary images, making the content of the image difficult to identify. In the 1970s, chaos theory was established and applied to various research fields [2]. At first, Matthews [3] proposed that chaotic systems could be used in cryptography. Chaotic systems are very suitable for image encryption due to their sensitivity to initial values, pseudo-randomness, ergodicity, and unpredictability [4]-[5]. The fundamental idea of chaotic cryptography is based on scrambling and diffusion [6]. Scrambling refers to the redistribution of pixel positions in a planar image, eliminating the high correlation between adjacent pixels, to obtain an image that is visually disruptive and unidentifiable [7]-[8]. Diffusion involves using chaotic mappings to disperse the redundancy in a planar image across a diffused image, thereby altering the pixel values [9]-[10]. People often integrate scrambling and diffusion to improve security and the effectiveness of encryption [11].

In the early days, people only used one-dimensional or two-dimensional chaotic systems for image encryption. Although low-dimensional chaotic systems have a simple structure, their chaotic behavior is limited, the key space is small, and the security is weak [12]-[13]. To improve the security of image encryption, a high-dimensional ($\geq$ 3D) chaotic system is used for image encryption [14]. However, due to the complex structure and multiple control parameters of high-dimensional chaotic systems, their computational complexity is higher. To address the shortcomings of low-dimensional and high-dimensional chaotic systems, researchers have proposed a new type of hybrid chaotic system. It consists of multiple interconnected chaotic subsystems, which can make the system more complex and less predictable [15]. This improvement enhances the system's security, increasing its ability to resist attacks or breaches [16]. By adjusting the coupling mode and strength between subsystems, it is possible to customize the behavior of the system, making it more suitable for specific encryption or chaotic applications [17]. This type of chaotic system can compensate for the deficiencies of low-dimensional chaos. Compared to high-dimensional chaos, this system is not only simpler, but also easier to implement.

In practical applications, there are many requirements to encrypt multiple images in one computation to improve computational efficiency. Zhang et al. [18] proposed a multi-image encryption algorithm based on DNA encoding. Zhang's solution is to arrange multiple images into a large image. Then, a general image encryption algorithm is applied to encrypt this large image. Using this method, it is possible to encrypt multiple images through a single encryption computation, the computational cost is almost equivalent to encrypting each image separately; K. A. K. Patro et al. [19] proposed a multi-image encryption technique. In this work, a

Manuscript received August 17, 2024; revised January 26, 2025.

This work was supported by the National Natural Science Foundation of China (62272093), the Department of Education of Liaoning Province (LJKFZ20220197) and the China Minmetals Corporation Science and Technology Special Plan Project (2021ZXA06).

Yilin Han is a graduate student of School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, China. (e-mail: 1148935711@qq.com).

Ye Tao is an associate professor of School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, China. (corresponding author to provide phone: +86-133-0422-4928; e-mail: taibeijack@163.com).

Shanshan Wang is a deputy director of the Science and Technology Innovation Center of Jinjie Company, China MCC22 Group Co Ltd, Tangshan, China. (e-mail: 969704626@qq.com).

Wenhua Cui is a professor of School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, China. (e-mail: cwh@systemteq.net).

Yuting Wang is a graduate student of School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, China. (e-mail: 2977354475@qq.com).

set of images is divided into non-overlapping blocks of size 2 × 2 pixels. Then these blocks are arranged in separate arrays. Finally, the block array is permuted and diffused using the PWLCM system. Due to the division of images into non-overlapping blocks, for large-sized images, the encryption efficiency of this method is not as high as that of single-image encryption schemes.

This paper proposes a multi-image encryption algorithm based on the composite chaotic system 1D-SATM and bit planes. Unlike previous image encryption schemes, this method is a batch image encryption algorithm, that requires only one encryption calculation to encrypt any number of images, and the computational cost is equivalent to encrypting a single image. Firstly, by using Sine and Tent maps, a new composite chaotic system called 1D-SATM is constructed. The new composite chaotic system has a larger chaotic space and better randomness, compared to the original two one-dimensional chaotic maps. In multi-image encryption, the stacking method is adopted instead of arranging multiple images into a large image.

II. Correlation Theory

A. Sine Chaotic Map

The Sine map is also a classic one-dimensional chaotic map [20], which has similar chaotic characteristics to the Logistic map. The mathematical formula is shown in (1).

$$x_{i+1} = a\sin(\pi x_i)/4 \tag{1}$$

Among them, the parameter $a \in (0.87,1]$. The bifurcation diagram of the Sine map is shown in Fig 1.

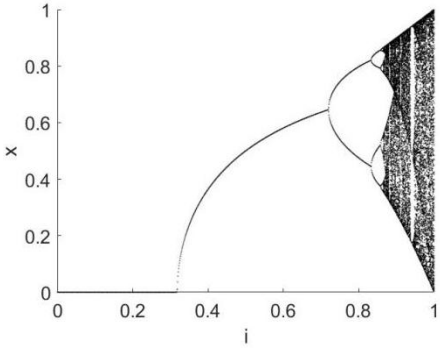


Fig. 1. Bifurcation diagram of Sine map

B. Tent Chaotic Map

The Tent map is named for its function graph resembling a tent and is a one-dimensional piecewise map [21]. Its mathematical expression is shown in (2).

$$x_{i+1} = \begin{cases} ux_i/2 & x_i < 0.5 \\ u(2-x_i)/2 & x_i < 0.5 \end{cases} \tag{2}$$

Parameter $u \in (0,4]$. The bifurcation diagram of the Tent map is shown in Fig 2.

C. 1D-SATM Chaotic Map

This chapter proposes a new composite chaotic system, the fundamental concept of which is to perturb one chaotic map with another, as shown in (3).

$$x_{i+1} = ((f_{main}(f_{seed}(x_i)) + f_{seed}(x_i))r^{\Omega}) \bmod 1 \tag{3}$$

Among them, f_{main} and f_{seed} are both classical one-dimensional chaotic maps and the calculation result of f_{seed} is used as the input of f_{main} . r^{Ω} is used to better distribute state variables in phase space, and the choice of the exponential value r^{Ω} is a trade-off between chaos and computational speed.

Despite iterating only two chaotic maps, the output of the proposed chaotic system is influenced by the dynamic characteristics of three different maps (the main map, the seed map, and the modified seed map). Therefore, compared to the method using only two chaotic maps, the proposed scheme exhibits more complex chaotic characteristics, and its computational complexity is significantly reduced compared to chaotic systems utilizing three maps. Based on the definition of the proposed chaotic system, we combine the Sine map with the Tent map, and its mathematical definition is shown in (4).

$$x\begin{cases} ((\frac{r}{4}\sin(\pi\frac{r}{2}x_i)+\frac{r}{2}x_i)r^{14}) \bmod 1 & x_1 < 0.5 \\ ((\frac{r}{4}\sin(\pi\frac{r}{2}(1-x_i))+\frac{r}{2}(1-x_i))r^{14}) \bmod 1 & x_1 < 0.5 \end{cases} \tag{4}$$

Among them, Ω =14, $a \in (0.87,1]$. i is the number of iterations, and in this article, $i = 23000$. The bifurcation diagram of 1D-SATM is shown in Fig 3. In addition, the output performance of the 1D-SATM was analyzed using the NIST SP800-22 testing standard. The test results are shown in Table I.

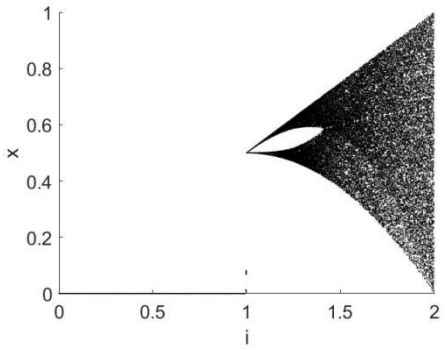


Fig. 2. Bifurcation diagram of Tent map

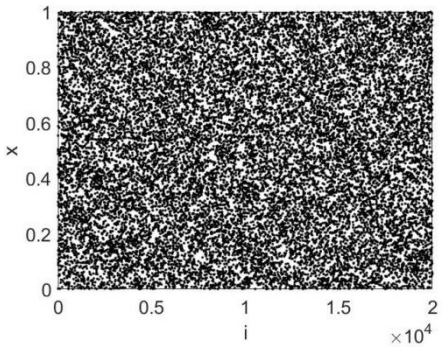


Fig. 3. Bifurcation diagram of 1D-LATM

D. Binary Bit Plane Decomposition (BBD)

In grayscale images, each pixel value is a decimal number between 0-255, which can be represented by an 8-bit binary sequence $(x_{n-1},...,x_1,x_0)$, as shown in (5).

$$N = \sum_{i=0}^{n-1} x_i 2^i = x_0 2^0 + x_1 2^1 + ... + x_{i-1} 2^{i-1} \tag{5}$$

TABLE I
STATISTICAL RANDOMNESS TEST RESULTS

Test	P-value	Result
Single-bit frequency test	0.4728	√
In-block frequency test	0.7074	√
Run test	0.4434	√
Test for longest run of ones in a block	0.1307	√
Binary matrix rank test	0.7712	√
Discrete Fourier (spectral) test	0.3304	√
Non-overlapping template matching test	0.4237	√
Overlapping template matching test	0.3539	√
Maurer's "Universal Statistical" test	0.8618	√
Linear complexity test	0.3543	√
Sequence test	0.4163	√
Approximate entropy test	0.7081	√
Cumulative sums and test	0.8650	√
Random travel test	0.7743	√
Random travel variant test	0.3329	√

BBD can divide grayscale images into 8 binary bit planes, with each bit plane representing the information of a single bit [22]. Usually, the highest bit plane contains the highest weight, representing the most significant image information. The lowest bit plane contains the lowest weight, representing the finest details. Fig 4 shows the grayscale image of the House and its decomposed 8-bit plane images. As seen from the figure, the image information becomes increasingly clearer from the 1-bit plane to the 8-bit plane.

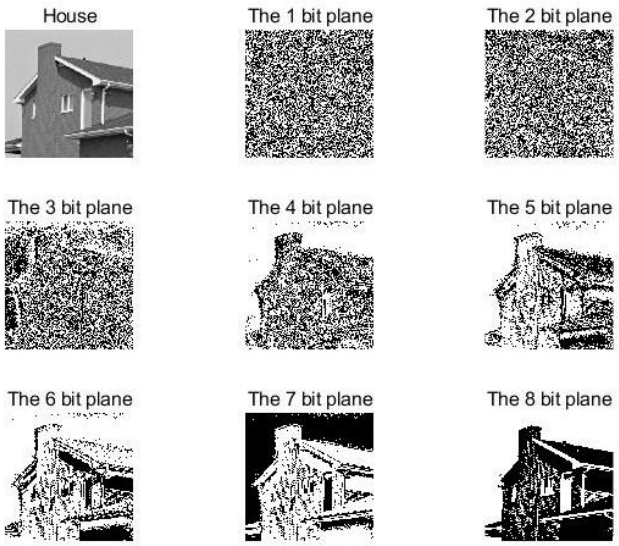


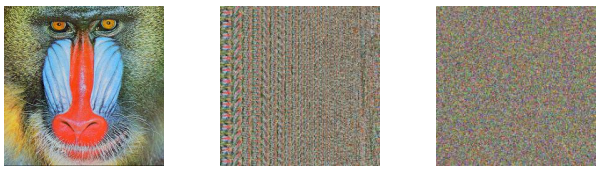
Fig. 4. Image House and its 8-bit planes

E. *Josephus Permutation Algorithm*

The Josephus permutation algorithm is a classic scrambling algorithm, commonly used to scramble an ordered list or sequence randomly. Usually, this algorithm is employed to generate random shuffle orders to enhance randomness or to safeguard data privacy. The traditional Josephus permutation algorithm sets a fixed starting position, start_position, and a fixed jumping distance, skip_distance.

This paper has made improvements by utilizing a dynamic Joseph loop. After each jump, the start_position and skip_distance are randomly generated. Each jump starts only from the start_position and ends at the last position of the sequence. The matrix traversal-info is employed to record the start_position and skip_distance of each traversal. During decryption, the matrix traversal_info is utilized to restore the image.

The traditional method and the improved method were used, to scramble the color Baboon image with a size of 256 × 256 (500 rounds). Fig 5 shows the result of using the traditional Joseph algorithm, and the result after optimizing the algorithm.



(a) Baboon image (b) Traditional method (c) Improved

Fig. 5. Baboon and two types of scrambled images

From the above figure, the traditional Josephus permutation algorithm still reveals certain features in the image, and it also exhibits a specific linear variation characteristic. By utilizing the algorithm optimized from the traditional algorithm for scrambling, the scrambling effect can be significantly enhanced, resulting in each pixel appearing more disordered after scrambling. Simultaneously, the two scrambling methods were timed to encrypt the image. For a color image of size 256×256 encrypted for 500 rounds, the traditional method took 209 seconds, whereas the improved method only required 9 seconds.

III. DESIGN AND IMPLEMENTATION OF ALGORITHMS

A. *Image Encryption*

The flowchart of the proposed image encryption algorithm is shown in Fig 6. Firstly, multiple color images

are overlaid to obtain the overlaid image S_0. Secondly, using BBD to decompose the stacked image into 8 bit planes. These 8 bit planes are divided into two groups, with the four higher-bit planes forming one group, denoted as $A1$; and the four lower-bit planes forming another group, denoted as $A2$. Thirdly, $A1$ and $A2$ are cyclically shifted to the right to obtain $A11$ and $A22$. Fourthly, the chaotic system 1D-SATM is used to generate a chaotic sequence X, which is decomposed into bit planes $x1-x8$. Similarly, plane A is divided into two groups, denoted as $b1(x1,x3,x5,x7)$ and $b2(x2,x4,x6,x8)$. Fifthly, XOR $A2$, $A11$, and $b1$ to obtain $B1$. Simultaneously, using the chaotic system 1D-SATM to generate the chaotic sequence $P1$, XOR $P1$ and $B1$ again to obtain $B1'$. Similarly, XOR $A22$, $B1'$, and $b2$ to obtain $B2$. Simultaneously using the chaotic system 1D-SATM to generate a chaotic sequence $P2$, XOR $P2$ with $B2$ again to obtain $B2'$. Sixthly, the improved Joseph scrambling algorithm for $B1'$ and $B2'$ yields $B11$ and $B22$. Seventhly, reshape the matrices $B11$ and $B22$ into new matrices E_1 and E_2, and then split the matrices E_1 and E_2 into bit planes E_{11} and E_{22}. Subsequently, 8 bit planes are extracted from E_{11} and E_{22}, each stored in variables from $e1$ to $e8$. Next, these 8 bit planes are combined to form a new matrix called ee. Finally, a switch statement is introduced to replace the values of different channels (red channel, green channel, or blue channel), based on the value of the channel. After the loop ends, the three encrypted red, green, and blue channels are merged to obtain the encrypted color image.

Assuming there are k color images of size M × N to be encrypted, we stack these color images into a single stacked image. Subsequently, we will perform an image encryption operation on this stacked image. The following outlines the specific encryption process and method:

(1) Overlay image

For color image S_i, there are:

$$M_{S_i}=\begin{bmatrix} a_{11}^{S_i} & a_{12}^{S_i} & \dots & a_{1N}^{S_i} \\ a_{21}^{S_i} & a_{22}^{S_i} & \dots & a_{2N}^{S_i} \\ \dots & \dots & \dots & \dots \\ a_{M1}^{S_i} & a_{M2}^{S_i} & \dots & a_{MN}^{S_i} \end{bmatrix},\ i=1,2,...,k \tag{6}$$

For the stacked image S_0, there are:

$$M_{S_0}=\begin{bmatrix} \sum_{i=1}^{k} a_{11}^{S_i} & \sum_{i=1}^{k} a_{12}^{S_i} & \dots & \sum_{i=1}^{k} a_{1N}^{S_i} \\ \sum_{i=1}^{k} a_{21}^{S_i} & \sum_{i=1}^{k} a_{22}^{S_i} & \dots & \sum_{i=1}^{k} a_{2N}^{S_i} \\ \dots & \dots & \dots & \dots \\ \sum_{i=1}^{k} a_{M1}^{S_i} & \sum_{i=1}^{k} a_{M2}^{S_i} & \dots & \sum_{i=1}^{k} a_{MN}^{S_i} \end{bmatrix}=\begin{bmatrix} a_{11}^{S_i} & a_{12}^{S_i} & \dots & a_{1N}^{S_i} \\ a_{21}^{S_i} & a_{22}^{S_i} & \dots & a_{2N}^{S_i} \\ \dots & \dots & \dots & \dots \\ a_{M1}^{S_i} & a_{M2}^{S_i} & \dots & a_{MN}^{S_i} \end{bmatrix} \tag{7}$$

M and N are the height and width of the superimposed image, respectively. $a_{p,q}^{S_i}$ represents the pixel value in (p,q) of the i-th color image.

To separate each color image from the stacked image during the decryption process, we use image matrices M_{S_i} and M_{S_0} to calculate the weight matrix W_i $(i=1,2,...,k)$ for each color image $S_i(i=1,2,...,k)$. The weight matrix is defined as shown in (8).

$$W_i=\begin{bmatrix} w_{11}^{S_i} & w_{12}^{S_i} & \dots & w_{1N}^{S_i} \\ w_{21}^{S_i} & w_{22}^{S_i} & \dots & w_{2N}^{S_i} \\ \dots & \dots & \dots & \dots \\ w_{M1}^{S_i} & w_{M2}^{S_i} & \dots & w_{MN}^{S_i} \end{bmatrix},\ i=1,2,...,k \tag{8}$$

Wherein, $w_{p,q}^{S_i}=a_{p,q}^{S_i}/a_{p,q}^{S_0}$, $p=1,2,...,M$, $q=1,2,...,N$.

(2) Separate the three channels of the color image in accordance with (9)

$$\begin{cases} redChannel=S_0(:,:,1) \\ greenChannel=S_0(:,:,2) \\ blueChannel=S_0(:,:,3) \end{cases} \tag{9}$$

(3) Loop through the image data of three channels

Extract the data from each channel, and store it in the variable S_d. Then, extract the pixel values bit by bit according to (10) and store them in S_1-S_8.

$$\begin{cases} S_1=\mathrm{mod}(S_d,2) \\ S_2=\mathrm{mod}(floor(S_d/2),2) \\ S_3=\mathrm{mod}(floor(S_d/4),2) \\ S_4=\mathrm{mod}(floor(S_d/8),2) \\ S_5=\mathrm{mod}(floor(S_d/16),2) \\ S_6=\mathrm{mod}(floor(S_d/32),2) \\ S_7=\mathrm{mod}(floor(S_d/64),2) \\ S_8=\mathrm{mod}(floor(S_d/128),2) \end{cases} \tag{10}$$

(4) Create bit plane matrices $A1$, $A2$

$A1$ contains the four lower-bit planes (S_1-S_4), while $A2$ contains the four higher-bit planes (S_5-S_8).

(5) Generate a key stream sequence

Assuming the size of the image to be encrypted is M × N, we iterate the 1D-SATM chaotic system R+M × N times according to (3), then discard the R-value. This yields a sequence of X, $X=\{x_1,x_2,...,x_{MN}\}$ with a length of M × N. Convert $X(i)$ to an integer sequence $X_1(i)$ using (11).

$$X_1(i)=\mathrm{mod}(floor(X(i)\times 10^{14}),256) \tag{11}$$

The range of elements in $X_1(i)$ is from 0 to 255. This paper uses BBD to decompose X into 8 bit planes, so there are eight binary sequences. Then, this paper flexibly divides the sequence into two groups on average. The four odd-bit planes form one group, and the four even-bit planes form another. This paper converts these two groups into two binary sequences $b1$ and $b2$, respectively, to obtain the binary key stream sequence.

(6) Diffusion

Use the chaotic sequences for cyclic shifts and XOR operations on $A1$, $A2$. Storing the results in $B1$, $B2$.

Step 1: Calculate the sum of all elements in $A2$ using (12).

$$sum_1=\sum_{i=1}^{L} A2(i) \tag{12}$$

Among them, L is the size of $A1$ and $A2$, $L=4MN$.

Step 2: Perform a right cyclic shift of $A1$ by sum_1

positions to obtain $A11$.

Step 3: Encrypt the first element in $A11$ by using the last element in $A11$ and the first element in $A2$ and $b1$, as shown in (13).

$$B1(1) = A11(1) \oplus A11(L) \oplus A2(1) \oplus b1(1) \quad (13)$$

When $i = 2$, there are:

$$B1(i) = A11(i) \oplus A11(i-1) \oplus A2(i) \oplus b1(i) \quad (14)$$

When $i = i+1$, continue to execute formula (14) until $i = L$.

Step 4: Generate chaotic sequence P. Firstly, define a block size $block\_size$, which determines the division of the chaotic sequence into multiple blocks for processing. Use a loop to generate the chaotic sequence to ensure it is long enough. The loop starts from the second position, and the starting and ending positions of each block are determined by $block\_start$ and $block\_end$. Within each block, values in the chaotic sequence are generated by calling the 1D-SATM function.

Step 5: As shown in (15), perform an XOR operation between the chaotic sequence P and $B1$ to further obfuscate the data. This operation is performed in blocks to ensure that the confusion is sufficiently uniform.

$$\begin{aligned} &B1(block\_start + i - 1) \\ &= B1(block\_start + i - 1) \oplus chaos\_block(i) \end{aligned} \quad (15)$$

Among them, $block\_length$ is the length of the block, $i \in [1, block\_length]$. $chaos\_block$ is the corresponding block extracted from the chaotic sequence.

Additionally, if the chaotic sequence is not long enough to handle the entire $B1$ array, a part of the chaotic sequence needs to be regenerated to ensure sufficient confusion.

Step 6: Similarly, use (16) to calculate the sum of all elements in $B1$; Perform a right cyclic shift on $A2$ by sum_2 positions to obtain $A22$; then obtain $B2(i)$ through (17).

$$sum_2 = \sum_{i=1}^{L} B1(i) \quad (16)$$

$$B2(i) = A22(i) \oplus A22(i-1) \oplus B1(i) \oplus b2(i) \quad (17)$$

XOR the value of the chaotic sequence with the value of $B2$ to confuse $B2$. If the chaotic sequence is not long enough to handle the entire $B2$ array, a part of the chaotic sequence will be regenerated.

(7) Scrambling

Use the improved Josephus permutation algorithm for $B1$, $B2$. Simultaneously, use the matrix traversal_info to record the start_position and skip_distance of each traversal. Use the matrix traversal_info to reconstruct the image during decryption. Call the improved Josephus permutation function (Josephus_Traverse) as shown in (18).

$$\begin{cases} B11 = \\ josephus\_traverse(B1, start\_position, skip\_distance) \\ B22 = \\ josephus\_traverse(B2, start\_position, skip\_distance) \end{cases} \quad (18)$$

Among them, start_position is the starting position traversed by the Josephus traversal, and skip_distance is the jumping distance traversed by the Josephus traversal. The values of start_position and skip-distance are obtained from (19).

$$\begin{cases} start\_position = randi(value) \\ skip\_distance = randi([1, value - cnt1]) \end{cases} \quad (19)$$

Among them, the value of $value$ is M × N. $cnt1$ is a variable used to record the index, its purpose is to track the number of elements that have been processed. $value - cnt1$ represents the number of elements remaining unprocessed. This ensures that the skip_distance does not exceed the range of unprocessed elements.

To achieve a better scrambling effect, This paper performs two rounds of the improved Joseph permutation.

(8) Obtain the encrypted image

Reshape the two matrices $B11$ and $B22$ into new matrices E_1 and E_2. Then, split the matrices E_1 and E_2 into bit planes E_{11} and E_{22}. Subsequently, 8 bit planes were extracted from E_{11} and E_{22}, with each bit plane stored in a variable from $e1$ to $e8$. Next, combine these 8 bit planes to form a new matrix called ee. Finally, introduce a switch statement to replace the values of different channels (red, green, or blue) based on the value of the channel. After the loop ends, merge the three encrypted red, green, and blue channels to obtain the encrypted color image.

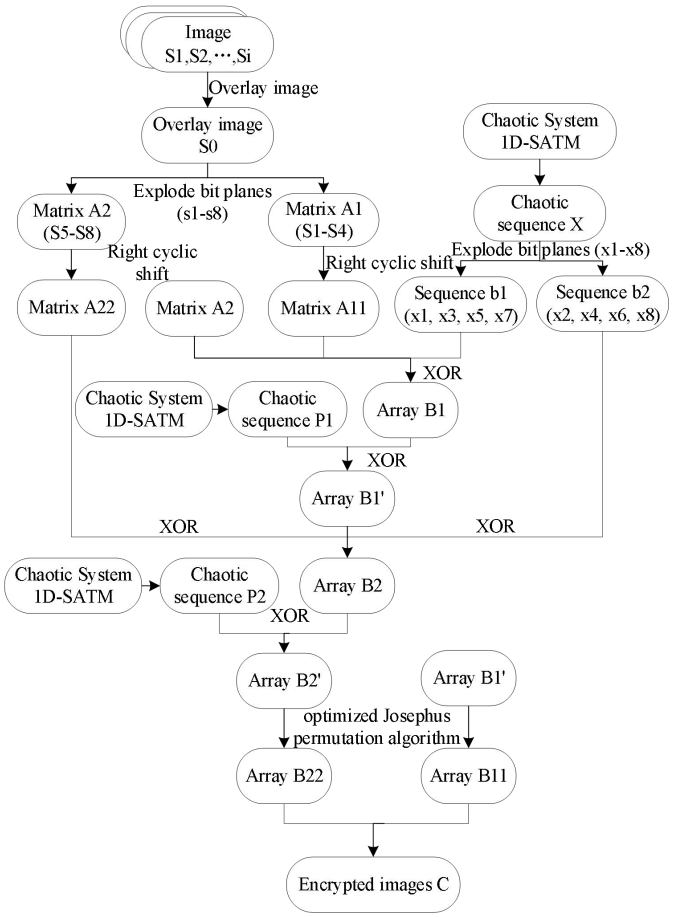


Fig. 6. Image encryption flow chart

B. Image Decryption

Image decryption is the inverse operation of the encryption algorithm. The flowchart of the image decryption is shown in Fig 7.

Firstly, separate the red, green, and blue channels from the encrypted image and decrypt each channel. Secondly,

separate the bit planes from $e1$ to $e8$ separately and combine them into E_1 and E_2, and reshape E_1 and E_2 into $B11$ and $B22$. Thirdly, perform the inverse operation of the improved Josephus permutation algorithm on $B11$ and $B22$ respectively, to obtain $B1'$ and $B2'$. Use the chaotic system 1D-SATM to generate chaotic sequences $P1$, $P2$, perform an XOR operation between $P1$ and $B1'$ to obtain $B1$, and perform an XOR operation between $P2$ and $B2'$ to obtain $B2$. Fourthly, use the chaotic system 1D-SATM to generate a chaotic sequence X, and decompose X into bit planes $x_1 - x_8$. Similarly, divide the bit plane $x_1 - x_8$ into two groups, denoted as $b1(x1, x3, x5, x7)$ and $b2(x2, x4, x6, x8)$. Fifthly, perform the inverse XOR operation on $B1'$, $B2$ and $b2$ to obtain $A22$. Perform the reverse cyclic shift operation on $A22$ to obtain $A2$. Perform the inverse XOR operation on $A2$, $B1$ and $b1$ to obtain $A11$. Perform the reverse cyclic shift operation on $A11$ to obtain $A1$. Sixthly, merge $A1$ and $A2$ into the bit plane to obtain a superimposed image. Finally, separate the stacked images and obtain n decrypted images.

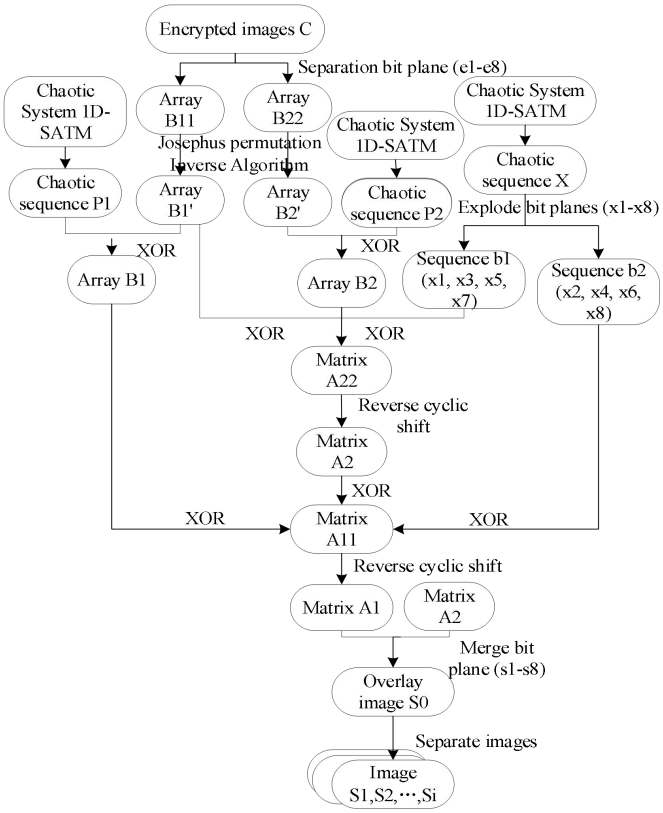


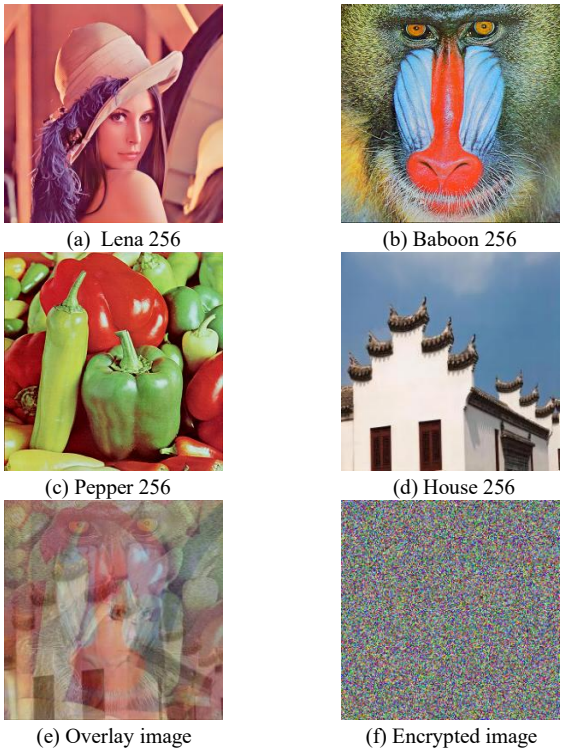
Fig. 7. Image decryption flow chart

IV. Experimental Results and Analysis

All simulation experiments were implemented using MATLAB R2017a. The programs were run on a computer with an Intel (R) Core (TM) i5-7200 CPU @ 2.50GHz. The images used in this section are from the USC-SIPI image library (http://sipi. usc. edu/database/). This section will analyze the security performance of the algorithm through a series of experiments, including runtime, key space analysis, correlation analysis, histogram analysis, sensitivity analysis (key sensitivity and plain sensitivity), information entropy analysis, and robustness analysis (noise addition and data loss).

A. Experimental Result

This paper selects Lena, Pepper, Baboon, and House color images with a size of 256 × 256 for encryption, as shown in Fig 8. This paper utilizes the proposed overlay algorithm to overlay the images, and the resulting overlay image is shown in Fig 8 (e). The stacked image is encrypted to produce the encrypted image as shown in Fig 8 (f).



(a) Lena 256 (b) Baboon 256

(c) Pepper 256 (d) House 256

(e) Overlay image (f) Encrypted image

Fig. 8. Plain, overlay and encrypted images

B. Running Speed

The stacked images were encrypted 100 times. Table Ⅱ compares their average running time with other literature algorithms. The experiment shows that this algorithm has a short running time and good encryption and decryption efficiency.

C. Key Space Analysis

The key space contains all valid keys [56]. As the key space increases, it becomes increasingly difficult to crack it using exhaustive search and other attack methods. The size of the key space should be greater than 2^{200}. In this paper, the initial values and parameters of the chaotic system 1D-SATM are used as the system keys. Assuming a double-precision calculation accuracy of 10^{16}, the key capacity of this algorithm can be obtained as $10^{16\times 8=128} \approx 2^{245}$, which meets the system requirements. If an exhaustive search attack takes seconds to perform, then cracking this encryption system by brute-forcing the keys would require $10^{128/(3.15\times 107)} \approx 3.175 \times 10^{120}$ years. The size of the key space of the algorithm in this paper and its comparison with other literature algorithms are shown in Table III.

D. Correlation Analysis

This paper randomly selects 5500 pairs of adjacent pixels (x_i, y_i) in the vertical, horizontal, and diagonal directions, from the stacked original image and its cipher image with a size of 256 × 256. As shown in Fig 9, the distribution of

adjacent pixels in the original image is relatively concentrated, indicating a strong correlation. In contrast, for the encrypted images, the distribution of pixel values is relatively uniform, which means that there is no correlation between adjacent pixels.

For the stacked images, this paper estimated the correlation coefficients between adjacent pixels of their plain images, and their corresponding cipher images, and the results are shown in Table IV. Table IV compares the results in this paper with some other algorithms. The correlation coefficient between adjacent pixels in the plain image is close to 1, indicating a high degree of correlation between the plain images. The correlation coefficient of the pixels in the cipher image is close to 0, which is approximately uncorrelated. This indicates that the algorithm can resist statistical analysis attacks.

E. Histogram Analysis

The histogram of the cipher image should be uniformly distributed. Figures 10 (a) - (c) show the histograms of the R , G , and B channels of the superimposed image, respectively. Fig 10 (d) - (f) show the histograms of the encrypted overlay image in three channels, respectively. As can be seen from Fig 10, the histograms of the stacked cipher images are very uniform, which means that this algorithm can resist statistical analysis attacks.

TABLE II
RUNNING EFFICIENCY AND COMPARISONS WITH OTHER ALGORITHMS

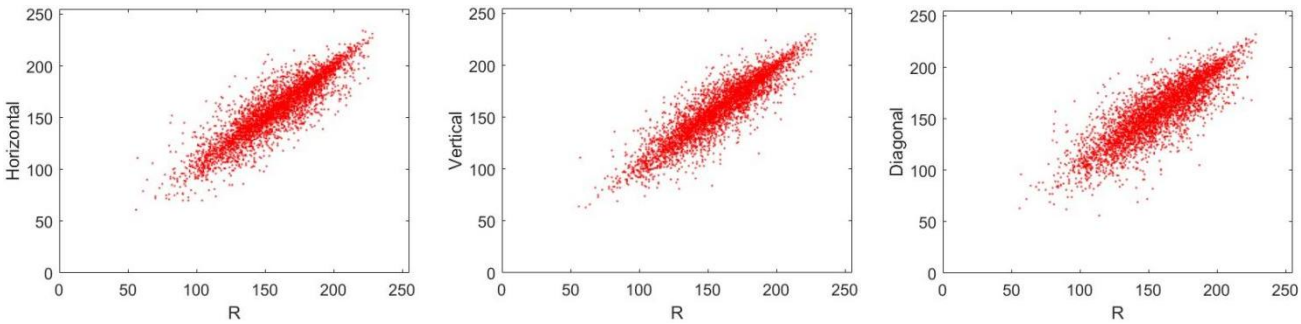
Algorithm	Our	Ref. [23]	Ref. [24]	Ref. [25]	Ref. [26]	Ref. [27]
Run time	1.5346	3.3550	1.6403	3.9810	2.7501	3.5721

TABLE III
KEY SPACE

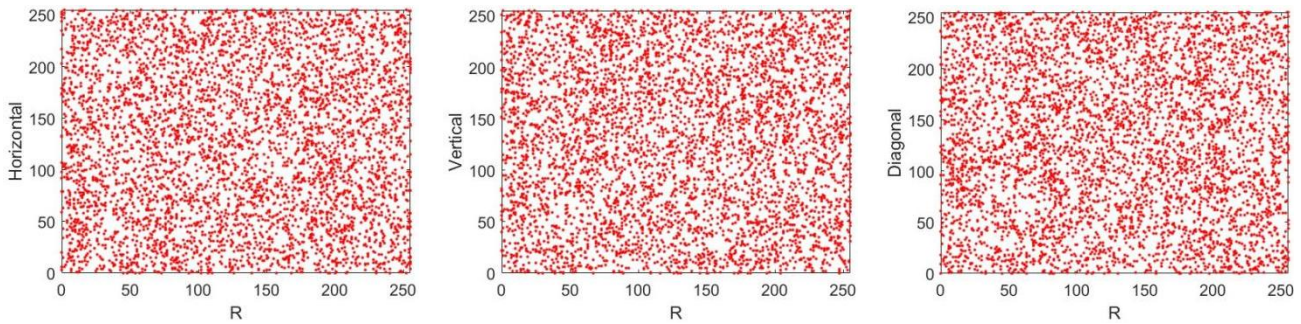
Algorithm	Our	Ref. [23]	Ref. [24]	Ref. [25]	Ref. [26]	Ref. [27]
Run time	10^{128}	10^{111}	10^{168}	1.4×10^{132}	10^{165}	3.31×10^{122}

TABLE IV
CORRELATION COEFFICIENTS

Image	Plain			Cipher		
	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
Overlay image	0.9164	0.9075	0.96556	-0.0095	0.01159	0.01564
Ref. [23]	0.9673	0.9482	0.9827	0.00144	−0.00151	0.00795
Ref. [24]	0.9872	0.9886	0.9792	-0.00386	0.02407	0.011671
Ref. [25]	0.9719	0.9444	0.9163	-0.0002	0.0052	0.0018
Ref. [26]	0.9391	0.9637	0.9638	0.00019	-0.00092	0.0037
Ref. [27]	0.9373	0.9235	0.9426	0.00044	0.00172	0.00141

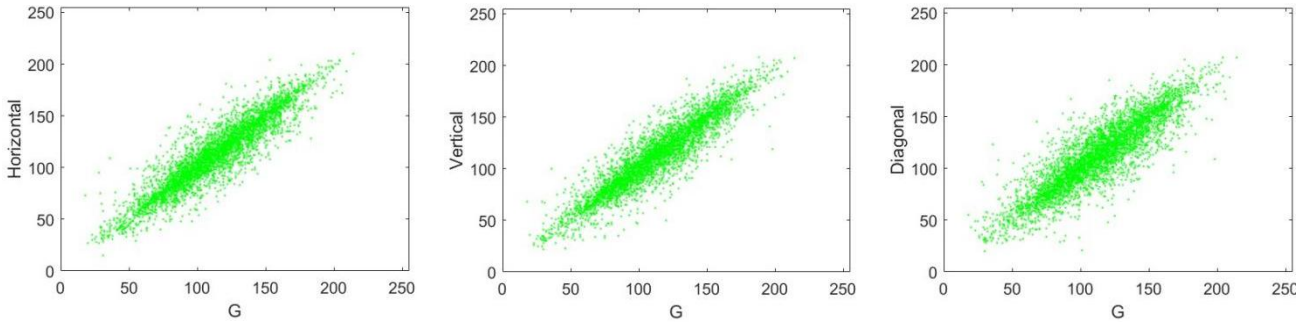


(a) Horizontal direction of the plain R (b) Vertical direction of the plain plain R (c) Diagonal direction of the plain R

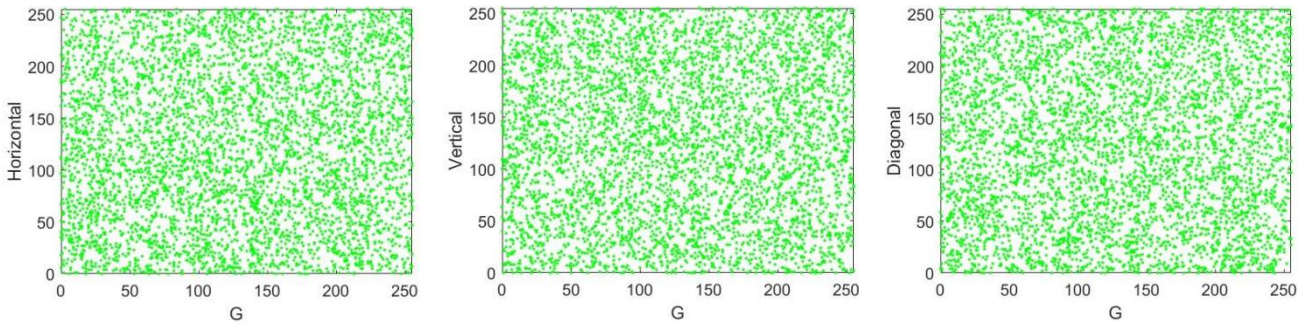


(d) Horizontal direction of the cipher R (e) Vertical direction of the cipher R (f) Diagonal direction of the cipher R

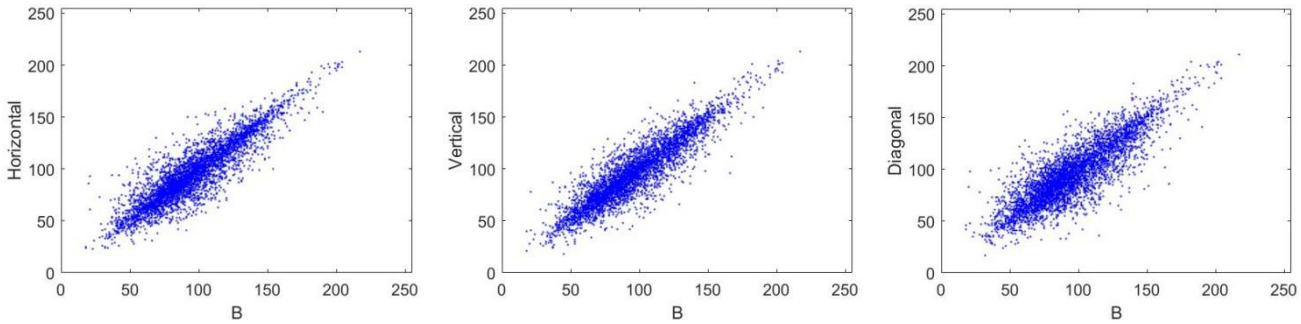
Fig. 9. Correlation analysis of superimposed image



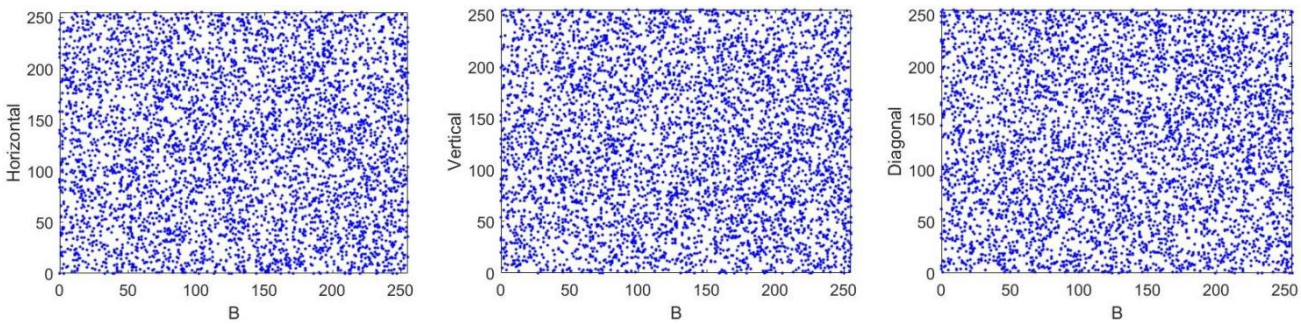
(g) Horizontal direction of the plain G (h) Vertical direction of the plain plain G (i) Diagonal direction of the plain G



(j) Horizontal direction of the cipher G (k) Vertical direction of the cipher G (l) Diagonal direction of the cipher G



(m) Horizontal direction of the plain B (n) Vertical direction of the plain plain B (o) Diagonal direction of the plain B



(p) Horizontal direction of the cipher B (q) Vertical direction of the cipher B (r) Diagonal direction of the cipher B

Fig. 9. Correlation analysis of superimposed image

F. Sensitivity Analysis

High levels of key sensitivity and plain sensitivity contribute to preventing various cryptanalysis analysis attacks and provide stronger data protection and confidentiality. Therefore, both of these two attributes are very important considerations in the design and evaluation of encryption algorithms.

(1) Key sensitivity

Key sensitivity refers to the property in an encryption algorithm, where even a slight change in the encryption key used leads to a significant difference in the encryption result. A good encryption algorithm should have high key sensitivity, which means that even a tiny change in the key

will result in vastly different encryption results, making it difficult for attackers to crack the key by comparing the encryption results under different keys. For the stacked images, we obtain the values of *NPCR* and *UACI* using the following formula. The test results and comparison with different algorithms are shown in Table V.

$$NPCR = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} D(i,j) \times 100\% \tag{20}$$

$$UACI = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{|C_1(i,j) - C_2(i,j)|}{255} \times 100\% \tag{21}$$

(2) Plain sensitivity

Plain sensitivity refers to the significant difference in encryption results, caused by slight changes to the plain (data to be encrypted) in encryption algorithms. Similar to key sensitivity, plain sensitivity is an important attribute of an encryption algorithm. If an encryption algorithm has high plain sensitivity, even a slight change in the plain, the encrypted result should be completely different. Plain sensitivity helps to prevent attackers from using techniques such as known-plain attacks, where attackers attempt to crack the key or gain insights into the encrypted data, by comparing the encryption results under different plain. For the stacked images, we obtain the values of *NPCR* and *UACI* through the equation. The test results and comparison with different algorithms are shown in Table VI.

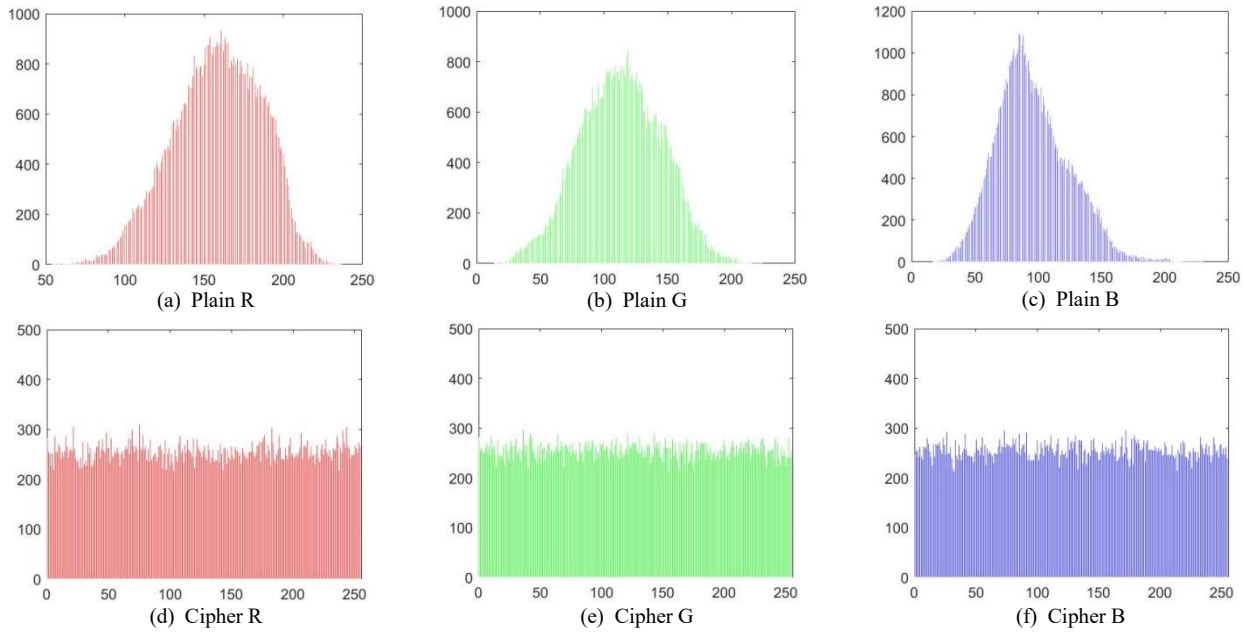
G. Information Entropy

Information entropy reflects the uncertainty of image information. The higher the entropy, the greater the uncertainty, and the less visible information. The formula for calculating information entropy is shown in (21). The theoretical value of information entropy is 8. The information entropy of this algorithm compared to different algorithms is shown in Table VII. From the table, it can be seen that this algorithm is sufficiently secure to resist information entropy attacks.

$$H = -\sum_{i=0}^{R} p(j) \log_2 p(j) \tag{22}$$

H. Robustness Analysis

Robustness refers to the property that an algorithm remains effective, even if the cipher image is subjected to some attacks. This section evaluates the robustness of the algorithm proposed in this paper through noise addition and image cropping.



(a) Plain R (b) Plain G (c) Plain B

(d) Cipher R (e) Cipher G (f) Cipher B

Fig. 10. Histograms of superimposed image

TABLE V
KEY SENSITIVITY ANALYSIS AND COMPARISONS WITH OTHER ALGORITHMS

Algorithm	Our	Ref. [23]	Ref. [24]	Ref. [25]	Ref. [26]	Ref. [27]	Theoretical
NPCR (%)	99.6081	99.6246	99.6401	99.6475	99.6090	99.6900	99.6094
UACI (%)	33.4614	30.5681	33.4775	31.2188	33.4649	33.4600	33.4635

TABLE VI
PLAIN SENSITIVITY ANALYSIS AND COMPARISONS WITH OTHER ALGORITHMS

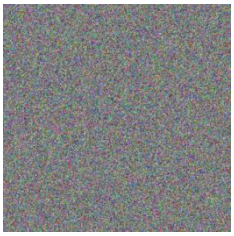
Algorithm	Our	Ref. [23]	Ref. [24]	Ref. [25]	Ref. [26]	Ref. [27]	Theoretical
NPCR (%)	99.6090	99.60501	99.6101	99.6510	99.6133	99.6800	99.6094
UACI (%)	33.4647	32.22662	33.6216	31.2179	33.4233	33.4600	33.4635

TABLE VII
Information entropy and comparisons with other algorithms

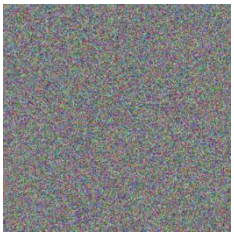
Algorithm	Our	Ref. [23]	Ref. [24]	Ref. [25]	Ref. [26]	Ref. [27]	Theoretical
Information entropy	7.9975	7.9968	7.9970	7.9971	7.9971	7.9977	8

(1) Noise attack

Noise addition is typically a method used to test or evaluate the robustness of data processing algorithms. Adding known or simulated noise to data. To evaluate the robustness of the algorithm in this chapter, 0.1% and 0.3% salt-and-pepper noise were added to the cipher image, respectively, and then the noisy cipher image was decrypted. After decryption, it was still possible to recover the original image information to a large extent, indicating the effectiveness of the algorithm. Fig 11 shows the results of adding noise to the cipher image and decrypting it.



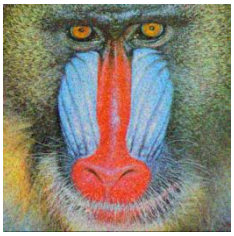
(a) Add 0.1 salt and pepper cipher image



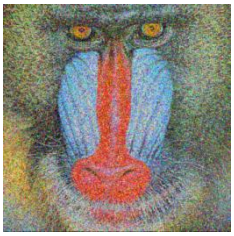
(b) Add 0.3 salt and pepper cipher image

(c) Decrypt Lena image from (a)

(d) Decrypt Lena image from (b)



(e) Decrypt Baboon image from (a)



(f) Decrypt Baboon image from (b)

(g) Decrypt Pepper image from (a)

(h) Decrypt Pepper image from (b)

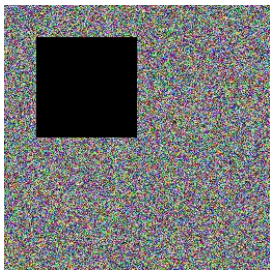
(i) Decrypt House image from (a)

(j) Decrypt House image from (b)

Fig. 11. Noise experiment

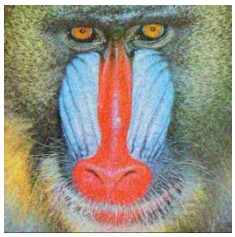
(2) Data loss

Data loss refers to the accidental loss of part or all of the data during data transmission, storage, or processing. The cipher image after data loss is shown in Fig 12 (a), and the decrypted image is shown in Fig 12 (b) - (e).



(a) The cipher image after data loss

(b) Decrypted Lena image



(c) Decrypted Baboon image

(b) Decrypted Pepper image

(c) Decrypted Pepper image

Fig. 12. Clip experiments

V. Conclusion

This paper proposes a multi-image encryption algorithm based on a composite chaotic system (1D-SATM) and bit plane. Unlike previous image encryption schemes, this method is a batch image encryption algorithm that requires only one encryption calculation to encrypt any number of images, with the computational cost equivalent to that of encrypting a single image. A new chaotic system, 1D-SATM, is obtained through coupling improvement, which increases the range of the chaotic map. Firstly, the stacked image is decomposed into 8 bit planes using Binary Bit Plane Decomposition (BBD), and these 8 bit planes are divided into two groups. Secondly, right cyclic shifts and XOR operations are performed on the two sets of bit planes. Among them, the chaotic sequence generated by the 1D-SATM chaotic system XOR with the diffusion result, to confuse the data further. Finally, the optimized Joseph scrambling algorithm is applied. Unlike the traditional Joseph shuffling method, which sets a fixed starting position and a fixed jumping distance, this algorithm has been improved by randomly generating the starting position and jumping distance after each jump. Through experimental analysis of the seven indicators of the algorithm proposed in this paper, it is found that the algorithm performs excellently, effectively resisting exhaustive attacks, differential attacks, and statistical attacks.

References

[1] T. Hu, Y. Liu, L. H. Gong, and C. J. Ouyang, "An Image Encryption Scheme Combining Chaos with Cycle Operation for DNA Sequences," Nonlinear Dynam, vol. 87, no. 1, pp. 51–66, 2017.

[2] Y. Zhou, L. Bao, and C. P. Chen, "A New 1D Chaotic System for Image Encryption," Signal Processing, vol. 97, pp. 172-182, 2014.
[3] R. Matthews, "On the Derivation of a "Chaotic" Encryption Algorithm," Cryptologia, vol. 13, no. 1, pp. 29-42, 1989.
[4] X. Chai, Z. Gan, Y. Lu, and et al, "A Novel Image Encryption Algorithm Based on the Chaotic System and DNA Computing," International Journal of Modern Physics C, vol. 28, no. 5, pp. 1750069, 2017.
[5] S. X. Zhang, S. S. Li, M. D. Bai, and et al, "Digital Color Image Encryption Technology Based on Chaos System," Science Technology and Engineering, vol. 22, no. 13, pp. 5291-5298, 2022.
[6] X. P. Yan, "Research on Key Technologies of Chaotic Image Encryption Based on Chaotic-Diffusion Structure," Dalian Maritime University, pp. 11-32, 2022.
[7] S. K. Du, "Application of Scrambling Technique in Image Encryption," Nanjing University of Posts and Telecommunications, pp. 6-7, 2022.
[8] X. Huang, and G. Ye, "An Efficient Self-adaptive Model for Chaotic Image Encryption Algorithm," Communications in Nonlinear Science and Numerical Simulation, vol. 19, no. 12, pp. 4094-4104, 2014.
[9] S. Q. Yang, "Hopfield Chaotic Neural Network and Symmetric Segmentation Diffusion for Image Encryption," Liaoning University of Engineering and Technology, pp. 26-28, 2022.
[10] Y. P. Li, "Image Encryption Algorithm Based on Cascade Chaos and Nonlinear Diffusion," Dalian Maritime University, pp. 56-57, 2022.
[11] Q. J. Wang, G. D. Li, and B. W. An, "Image Encryption Algorithm Based on Chaotic-Diffusion and New Combination Chaotic System," Journal of Chifeng University (Natural Science Edition), vol. 38, no. 6, pp. 17-21, 2022.
[12] S. Yang, and X. Tong, "A Block Image Encryption Algorithm Based on 2D Chaotic System," IEEE International Conference on Software Engineering and Service Science, 2021.
[13] L. Liu, Y. Lei, and D. Wang, "A Fast Chaotic Image Encryption Scheme with Simultaneous Permutation-Diffusion Operation," IEEE Access, vol. 99, pp. 1-1, 2020.
[14] H. Chen, E. Bai, X. Jiang, and et al, "A Fast Image Encryption Algorithm Based on Improved 6-D Hyper-Chaotic System," IEEE Access, vol. 10, pp. 116031-116044, 2022.
[15] M. T. Elkandoz, and W. Alexan, "Image Encryption Based on a Combination of Multiple Chaotic Maps," Multimedia Tools and Applications, vol. 81, no. 18, pp. 25497-25518, 2022.
[16] M. Es-Sabry, N. El. Akkad, M. Merras, and et al, "A New Color Image Encryption Algorithm Using Multiple Chaotic Maps with the Intersecting Planes Method," Scientific African, vol. 16, pp. e01217, 2022.
[17] Y. Tao, W. H. Cui, and Z. Zhang, "Spatiotemporal Chaos in Multiple Dynamically Coupled Map Lattices and Its Application in a Novel Image Encryption Algorithm," Journal of Information Security and Applications, vol. 55, pp. 102650, 2020.
[18] Y. Chen, S. Xie, and J. Zhang, "A Novel Double Image Encryption Algorithm Based on Coupled Chaotic System," IOP Publishing Ltd, 2022.
[19] K. A. K. Patro, and B. Acharya, "A Novel Multi-Dimensional Multiple Image Encryption Technique," Multimedia Tools and Applications, vol. 79, no. 19-20, pp. 12959-12994, 2020.
[20] S. M. Basha, P. Mathivanan, and A. B. Ganesh, "Bit Level Color Image Encryption Using Logistic-Sine-Tent-Chebyshev (LSTC) map," Optik, PP. 259, 2022.
[21] M. Es-Sabry, N. El. Akkad, M. Merras, and et al, "A New Color Image Encryption Algorithm Using Multiple Chaotic Maps with the Intersecting Planes Method," Scientific African, vol. 16, pp. e01217, 2022.
[22] C. Xu, Y. P. Tao, Y. Zhu, and et al, "A Reversible Information Hiding Method Based on Lossless Compression of High Bit Planes in Image Blocks," Journal of Northeast Normal University (Natural Science Edition), vol. 55, no. 3, pp. 101-108, 2023.
[23] X. Zhang, and X. Wang, "Multiple-Image Encryption Algorithm Based on DNA Encoding and Chaotic System," Multimedia Tools Appl, vol. 78, no. 6, pp. 7841 – 7869, 2018.
[24] Z. Tang, Z. Yin, R. Wang, and et al, "A Double-Layer Image Encryption Scheme Based on Chaotic Maps and DNA Strand Displacement," Journal of Chemistry, pp. 1-10, 2022.
[25] Y. Wang, C. Wu, S. Kang, and et al, "Multi-Channel Chaotic Encryption Algorithm for Color Image Based on DNA Coding," Multimedia Tools and Applications, vol. 79, no. 25, pp. 18317-18342, 2020.
[26] J. Hao, H. Li, H. Yan, and et al, "A New Fractional Chaotic System and Its Application in Image Encryption with DNA Mutation," IEEE Access, vol. 9 pp. 52364-52377, 2021.
[27] W. Feng, Q. Wang, H. Liu, and et al, "Exploiting Newly Designed Fractional-Order 3D Lorenz Chaotic System and 2D Discrete Polynomial Hyper-Chaotic Map for High-Performance Multi-Image Encryption," Fractal and Fractional, vol. 7, no. 12, pp. 887, 2023.