

Secure image sharing method over unsecured channels

Hazem Al-Najjar · Saeed Alharthi · Pradeep K. Atrey

Received: date / Accepted: date

Abstract Image sharing has become a prominent field in large and small networks. However, in order to deliver the shares to their respective shareholders, we need a secure channel to protect the shares from potential attacks by other users on the network. These users can locate the shares and retrieve the image using the Lagrange interpolation method if they are able to locate the minimum required number of shares. This paper proposes a method to protect the privacy and security of the shares when sending them through unsecured channels. The proposed method is divided into three phases. In the first phase, image pixels are changed using the column and row indices, then the shares are created using Shamir's secret sharing (SSS) method. Then a relationship is created between the shares using a linear independence function. In the second phase, the first part of the image is encrypted to hide the solver data. Finally, in the third phase, the data is shared using the SSS method. Participants can retrieve the whole image by applying the reverse order of the proposed method using only the information from the sender through the same channel. The simulation results show that the proposed method is efficient and robust against different types of attacks and can be used to send the shares over unsecured channels.

Keywords Image · secret sharing · encryption · chaotic theory · linear independence

1 Introduction

Secret image sharing (SIS) has been widely studied in the past decade [1, 2, 3, 4, 5]. In SIS, the secret image is divided into a number of arbitrary image shares (say k), which are distributed to different users. A minimum number of shares (say $t \leq k$) is needed to retrieve the secret image. Although image shares are usually completely arbitrary and individual shares do not reveal any information about the secret image, their secure distribution over an unsecured network is quite challenging. Examples of unsecured networks include a wireless camera network, where the camera nodes usually send captured images to the base station. In this scenario, to protect the images from an adversary one can adopt a (t, k) -SSS based solution in which the n shares of an image captured at the camera are sent to the base station via different routes. However, this solution does not guarantee security since an adversary near the base station can easily capture t or more shares and reconstruct the secret image. The share images can be encrypted using any strong encryption algorithm such as Advanced Encryption Standard (AES), but it

Hazem Al-Najjar and Saeed Alharthi
Department of Computer and Information Sciences, Taibah University, Al-Madinah, Saudi Arabia
E-mail: hazem_najjar@yahoo.com, E-mail: sharhi@taibahu.edu.sa

Pradeep K. Atrey
Department of Computer Science, University at Albany - State University of New York, Albany, NY, USA
E-mail: patrey@albany.edu

is computationally expensive, especially in a wireless network scenario where the computation power of camera nodes is limited.

Many researchers [?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?] discussed methods for sharing an image or data among different users. These methods assume that an attacker cannot gather t or more shares. This assumption may not hold true especially in unsecured networks such as a wireless camera network. We found that there is little research focusing on sending shares over such unsecured networks in a trusted and safe way, such that even if an attacker finds t or more shares, s/he should not be able to obtain any information about the secret data. The main focus of this paper is to deal with security issues in such scenarios.

In this paper, we propose a method for distributing a secret image over unsecured channels. The core idea behind our method is to use a linear independence relationship and chaotic map in order to encrypt the shares of the image. In our method, we encrypt only the part of the share image instead of whole share image, which helps in achieving efficiency yet it does not affect the security.

Our contributions in this paper are summarized as follows:

- Designing a method to send all shares over unsecured channels.
- Improving the efficiency of the secure sharing method by encrypting part of the image.
- Using an efficient priority method by allowing privileged users to hold small amounts of information.

The rest of this paper is organized as follows. In section 2, we first present an overview of the techniques used in the proposed method and then describe the past works and highlight the novelty of the proposed work against them. We describe the proposed method in detail in section 3. Experimental results are discussed in section 4. The security analysis is presented in section 5. Finally, our conclusions are outlined in section 6.

2 Background and Related Work

2.1 Background

2.1.1 Linear independence

The linear independence relationship is a combination of two or more variables, using any mathematical function such as the XOR operation or addition with modulus to describe the relationship between them without describing the original variables [?].

2.1.2 Logistic map

A logistic map is a chaotic function that generates a chaotic code and quantifies the sensitivity to initial conditions. Tiny changes in the initial conditions result in extremely different behaviors. The logistic map can be described as follows [?]:

$$X_{q+1} = \lambda X_q (1 - X_q) \quad (1)$$

$\lambda \in (0, 4]$, $X \in (0, 1)$ as shown in Figure.1, where the chaotic behavior is achieved, when $\lambda \in [3.57, 4]$. The proposed method uses a logistic map to generate a chaotic code to shuffle the first part of the image.

2.1.3 (t, k) -SSS method

In 1979, Shamir [?] came up with a data sharing scheme, which allows to share a secret data among many people in such a way that the individual shares did not reveal any information the secret, but the secret can be reconstructed if a certain number of people combine their shares. In this scheme, a (t, k) -SSS method as shown in Eq.2 is used. In this method, s is the secret to be shared, m is a prime number chosen by the dealer and a_1, a_2, \dots, a_{t-1} are integer coefficients chosen randomly within $[0, m - 1]$ to generate a $(t - 1)$ -degree polynomial $F(x)$.

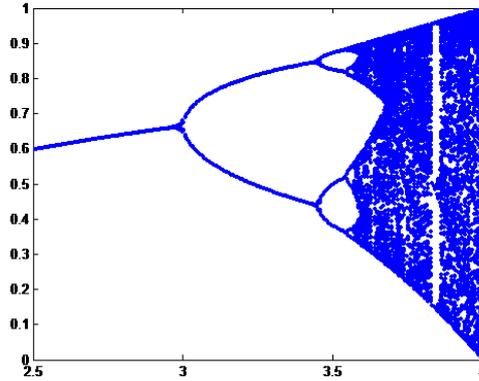


Fig. 1 Logistic map bifurcation

$$F(x) = s + a_1 x + \dots + a_{t-1} x^{t-1} \pmod{m} \quad (2)$$

The above polynomial is evaluated at different x values and the k shares are created and distributed to k people, while any t shares (where $2 \leq t \leq k$) are needed to reconstruct the secret s .

2.1.4 Application of SSS

From the perspective of application of SSS scheme, Zhu et al. [?] developed a novel cloud storage protocol based on SSS scheme called Entangled Authenticated Cloud Storage (EACS), to provide the same quality service for each client in the cloud system and to ensure authentication and integrity in the storage system. The entangled method was used to design a scheme which has four key policies to describe the way that the clients worked with the storage protocol and how the storage cloud treated the clients in the system when one client deleted, removed or modified the files in the system. In other work, Korzhik et al. [?] analyzed the capacity of a stegosystem for the noisy attack channel by considering the scenario where an attacker may know the cover message of the transmitted stegosignal which was received over a noisy channel. Moreover, after considering a strong upper bound for the SG and a lower union bound for the probability of block error, the author proved that the capacity of a stegosystem for the noisy attack channel is equal to zero. Also, Chen and Huang [?] proposed a co-evolutionary genetic watermarking scheme based on wavelet packet transform to improve the capability to resist specific image processing methods while keeping the quality of the watermarked image acceptable. After testing and evaluating the method, the authors noticed that this characteristic would make genetic watermarking schemes more applicable in real-world applications.

2.2 Past Works on SIS

Thien and Lin [?] used Shamir's (t, k) scheme for secure image sharing, in which a dealer creates k shares of an image and any t image shares are combined to reconstruct the secret image. Later, Alharthi and Atrey [?] proposed an algorithm to minimize the size of shares sent by using the adjacent pixels within the same row. So, if the dealer shares an image with a size of 300×300 pixels between three participants, each share's size is 100×300 pixels for each participant. In order to retrieve the image all the participants need to collaborate to retrieve the adjacent pixels from each participant using the Lagrange interpolation method. Hu et al. [?] designed a concept to share a secret image based on chaotic map and Chinese

Remainder Theorem (CRT) where the logistic map was used in phase 1, and then CRT was used in phase 2 to generate the shares. Poonkuntran and Rajesh [?] also demonstrated the use of chaotic model for image authentication. In the other work, Lang [?] designed a novel secure image sharing scheme based on Shamir's three-pass cryptography protocol and the multiple-parameter fractional Fourier transform. This method achieved a high level of validity and reliability in the network. However, in this protocol exchange of the keys between participants compromises the privacy.

In other works, Chang et al. [?] proposed a novel image sharing scheme using a 16×16 -sized Sudoku puzzle as a share grid generator. The experiments showed that the shares can be successfully camouflaged in the host image with satisfactory quality. Furthermore, to enhance the security of distributed file systems over unsecured channels, Zanin, et. al [?] proposed an application for the secret sharing method to secure distributed file systems from multiple attacks if less than n nodes are attacked. The main weakness of his method, which is solved in the proposed method, is that if all the nodes are attacked then the attacker can retrieve the shared file. Recently, Atrey et al. [?] demonstrated the use of SSS method for sharing of sensitive video evidences, however the focus of their work was not to address the security of shares over unsecured channels.

Yuan and Xu [?,?] proposed to encrypt the image using a symmetric encryption scheme such as the chaos-based technique or one-time pad and distribute the encrypted image with the key to different participants. However, the shares should be the same size as the original image. Therefore, this approach requires a great amount of time to achieve effective encryption. Shi [?] proposed to encrypt the secret image by generating public and private images, then distributing the public image to the participants with the private image which used SSS. However, the problem of increasing the share size between the participants will arise, which will decrease the speed and require a great amount of time to encrypt and decrypt the images. Lang and Malik's [?,?] method was to encrypt the image using any encryption scheme then distribute the encrypted image to all the participants without exchanging the key by using a three-pass protocol or with a keyless approach. This minimizes the information shared between the dealer and the participants. However, the information can be leaked as decryption is possible for the chain of users. Feldman [?] proposed to share the image using SSS then encrypt each share before distributing them among the participants using a public key. Still, all the data must be encrypted before sending the shares over unsecure channels, which will be time-consuming in encrypting and decrypting. Huang et al. [?] and Pan et al. [?] proposed algorithms for vector quantization (VQ) based image watermarking with multiple description coding (MDC), which can be used in error-resilient transmission over noisy channels.

In contrary to the above works, the proposed method encrypts only a part of the image depending on the number of shares and then shares the images with the encrypted keys. This minimizes the size of the shares and therefore reduces the execution time for sharing and retrieving the image, while achieving security on unsecured channels.

3 Proposed Method

As shown in Figure 2, the proposed method is divided into three phases. In the first phase, the intermediate shares are generated after applying Eq. 2 on the image so that these intermediate shares can be used to create a linear independence relationship. Additionally, rather than encrypting the whole image (which requires a large amount of time), the linear independence relationship and the logistic map are used. Establishing the linear independence relationship helps in creating a relation between the first intermediate share, which is called a relationship solver, and the rest of intermediate shares in phase 1 to hide the information of other parts. Moreover, using the independence relationship connects the intermediate shares with the solver; so without it, the other shares cannot be retrieved. Furthermore, to create a linear independence relationship, the minimum number of intermediate shares should not be less than two: one solver (should be encrypted) and the remaining intermediate shares connected to the solver with a mathematical equation required to retrieve said intermediate shares using the solver. In phase 2, the encryption algorithm is only applied on the solver, which can be

Algorithm.1: Secure Image Sharing

1. **Input** Image, t_1 , t_2 , key
2. **Output** Shares, key
3. **Pad** the image if necessary
4. **Modify** the image range [0,250]
5. **Replacing** image pixels using Eq.3
6. **Create** t_1 Parts using Eq.5
7. **Create** Linear Independence Relationship using Eq.6
8. **Shuffle** ($Part_1$) // using chaotic logistic Map, key_1 and key_2
9. **Create** t_2 Shares using Eq.8
10. **Send** Shares and RSA (key) to participants

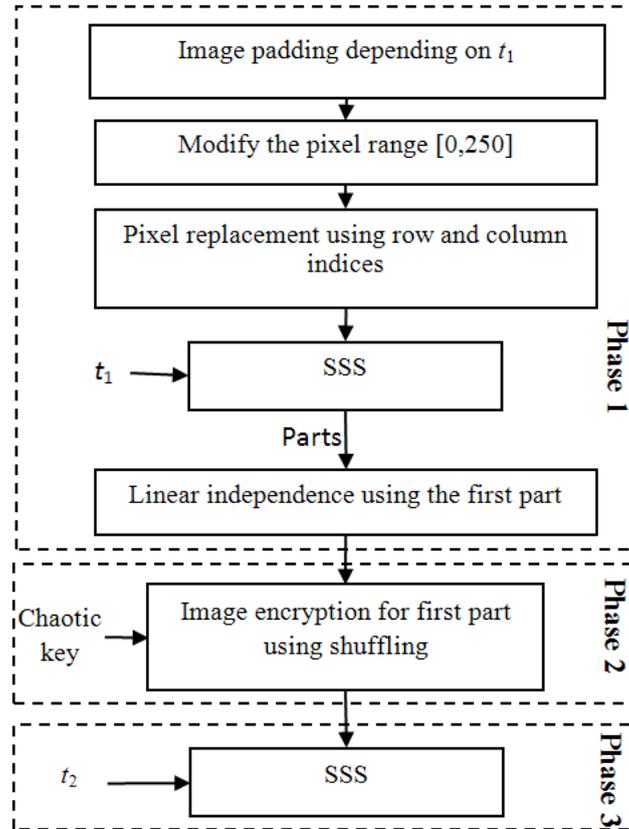


Fig. 2 Flowchart of the three phases in the proposed method

any intermediate share of the image. Without the solver, participants are unable to retrieve the remaining intermediate shares of the image. Finally, in phase 3, the final image shares are created using Eq. 2 and provided to the participants in the network. Note that the intermediate shares and the final shares are different. The intermediate share is used to describe the share of the image (using SSS) in phase 1, whereas the final shares are the shares of the image generated in the final step. Also, the number of intermediate shares and the number of shares could be different as only the latter is determined based on the number of participants. In the rest of the paper, we will call intermediate shares as ‘parts’ and final shares as ‘shares’.

In the following subsections, first these three phases are described in detail (in section 3.1) and then proposed method is outlined in algorithmic steps (in section 3.2).

3.1 Details of Three Phases

3.1.1 Phase 1 - creating parts

Let I be an input image of size $N \times M$. Before replacing the pixels, the pixels need to be modified and ranged between 0 and 250. The SSS method (Eq. 2) works in a finite field of size m , therefore m must be a prime such as 251. Furthermore, the number of pixels that are greater than 250 in standard images is small, so truncating the pixels that are greater than 250 creates no noticeable effect. This causes a small data loss; however, the goal of the proposed method is not to provide a lossless recovery of the secret image (as such recovery is prevented by the pixel value truncation to 250).

The first phase in the proposed method is the replacement phase, in which the image pixels are replaced by new values using the row and column indices in the input image as shown in Eq. 3. In this equation, the $I(i, j)$ denotes the pixel value in the input image I at (i, j) coordinates, and $+$ is the addition operation between the terms.

$$I(i, j) = (I(i, j) + ((i + j) \bmod 251)) \bmod 251 \quad (3)$$

The resulting image is divided into t_1 parts, depending on the number of columns and the number of t_1 in Eq. 2, using Eq. 4 to validate the parts' equality. If the parts are not equal, the padding approach is used.

$$\begin{cases} padding, & \text{if } \bmod(N, t_1) \neq 0 \\ no\ padding, & \text{if } \bmod(N, t_1) = 0 \end{cases} \quad (4)$$

where t_1 is the number of parts

The image is used to generate t parts depending on the number of t in Eq. 2, which is shown in Eq. 5. The resulting output of Eq. 2 generates a part, which is used to differentiate between the shares that are transmitted to the participants in phase 3, and the resulting shares to be generated, before creating a linear independence relationship in phase 2.

In this equation, the adjacent pixels in the image are considered as equation coefficients, t_1 is the number of parts, $i = r$ and $j = (no - 1 \times M/t_1 + c)$.

$$Part_{no}(r, c) = (I(i, j) + I(i, j + 1)x + I(i, j + 2)x^2 \dots + I(i, j + (t_1 - 1))x^{t_1 - 1}) \bmod 251 \quad (5)$$

$$\forall i, r \in 1, 2, \dots, N, \quad j \in 1, 2, \dots, M, \quad no \in 1, 2, \dots, t_1, \quad c \in 1, 2, \dots, M/t_1$$

After division, t_1 parts are used to create a linear independence relationship between each of the parts in order to increase the randomness in the shared image by creating a chain of dependency between current and previous parts. In Eq. 6, each part creates a relationship with the first part to increase the number of relationships between parts and to increase randomness. For simplicity, addition with modulus operations are used to create a relationship.

$$Part_{no}(r, c) = (Part_1(r, c) + Part_{no}(r, c)) \bmod 251 \quad (6)$$

$$\forall no \in 2, 3 \dots t_1$$

The generated parts will then be used in the following steps to create shares to be sent to the participants. Before creating shares, $Part_1$ is shuffled using the chaotic logistic map in order to hide the original share. As a result, if an attacker detects all the shares, the shuffled part will prohibit the attacker from retrieving the original image.

3.1.2 Phase 2 - applying chaotic logistic map

$Part_1$ is shuffled using a chaotic logistic map in order to increase the randomization in the image and to hide the solver of the linear independence relationship. After considering the initial conditions (chaotic key) of the chaotic model, the chaotic values are generated and then sent to the users in the network after encrypting the initial conditions in any strong encryption algorithm such as RSA. The dealer, after shuffling $Part_1$, sends the encrypted chaotic key to the users in the network.

A logistic map equation is used to generate chaotic indices to shuffle $Part_1$. In this equation, chaotic key_1 and chaotic key_2 generate the row and column sequences, respectively, and, by swapping between the chaotic values and the sequence values of a part, the resulting part is shuffled. To generate key_1 and key_2 , Eq. 7 combine the chaotic keys in a mathematical equation where the factor is defined as a public integer number.

$$key_1 = \frac{1}{key}, \quad key_2 = \frac{factor}{key} \quad (7)$$

3.1.3 Phase 3 - data sharing

The generated parts are used in equation Eq. 2 to create shares. In this equation, all parts are used to create shares using the adjacent pixels in the same row. For simplicity, we assume that the number of parts equals three. The pixels in each part are considered the equation's coefficients as shown in Eq. 8, where, $Part_1$, $Part_2$ and $Part_3$ are the first, second and third parts generated after completing the second phase. In this case, three shares and the encrypted key are needed to retrieve the original image.

$$Share_s(r, c) = (Part_1(r, c) + Part_2(r, c)x + \dots + Part_{t_2}(r, c)x^{t_2-1}) \bmod 251 \quad (8)$$

$$\forall s \in 1, 2, 3 \dots t_2$$

3.2 Algorithmic Steps

The algorithm used in the proposed method is shown in algorithm 1, in which the image is divided into a number of parts after applying the replacement approach in the image using the row and column data. If the size of the column's image is less than the required number, a padding approach is applied on the image to create t_2 shares with the same size. The linear relationship between parts is created to increase randomization in the shared parts. Then, in order to increase the sharing complexity, the first part is shuffled and then the shares are generated using the parts' pixels as coefficients in Eq. 2. The size of the parts and the shares in this process is less than or equal to the size of the input image. While the size of the parts can be determined by dividing the size of the input image by the number of parts, the size of the shares can be calculated by dividing the size of the input image by the minimum number of shares required to reconstruct the image. Finally, the dealer encrypts the chaotic key using RSA and then sends the shares with the encrypted key to the participants in the network. Moreover, after generating key_1 and key_2 using Eq.7, the two keys will be used in Eq.1 separately to generate x-values and y-values, i.e. to generate x-values $x_0 = key_1$ and to generate Y-values $x_0 = key_2$.

3.3 Reconstruction Phase

The reconstruction algorithm is shown in algorithm.2, in which the Lagrange interpolation will be used with shares to retrieve all parts, where the first part will be decrypted to find the linear independence solver for the rest of parts. The first part after decryption will be used to solve the linear independence between other parts but instead of using Eq.6, the reverse equation which will change addition to subtraction is used. After that, the parts will apply

Algorithm.2: Image Reconstruction

-
-
1. **Input** $shares = s_1, s_2, \dots, s_n, key$
 2. **Output** Image
 $Parts = \text{Lagrange Interpolation} (shares)$
 $Parts = \text{Decrypt} (Parts, key)$
 $Parts = \text{Solve Linear Independence Relationship between Parts using reverse Eq. 6}$
 $Image = \text{Lagrange Interpolation} (Parts)$
 3. **Replacing** Image pixels using reverse Eq. 3
 4. **Remove** Padding if any
 5. **Retrieve** Image to the participants
-

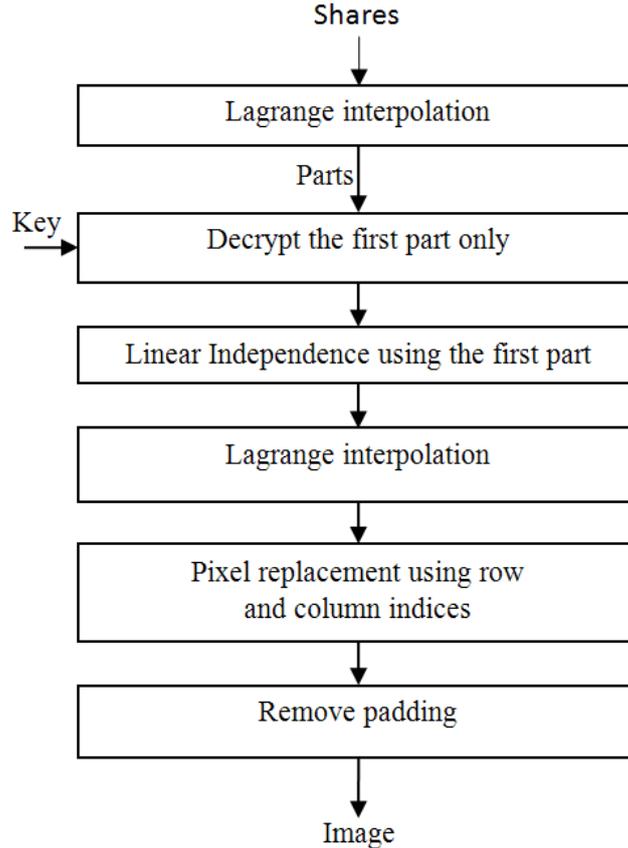


Fig. 3 Flowchart of the reconstruction phase in the proposed method

the Lagrange interpolation to retrieve the replaced image. The replaced image will consider a reverse equation of Eq.3 with a row and column indices to retrieve the Padded image. Finally, the padding on the image will be removed to retrieve the original image. To clarify, the flowchart of three phases in the proposed method is shown in Figure. 3.

4 Experimental Results

To demonstrate the utility of the proposed method we performed experiments with three standard images: Lena, Cameraman, Peppers and Airport, as shown in Figure 4. The key and factor are equal to 1234 and 10, respectively, to evaluate the performance and the applicability of the method. For the color images the process will be the same but instead of using only one matrix (for the gray channel), three metrics will be used (one for each color channel). Therefore, in all of the below processes the gray images will be used. Note that although in experiments we have used 1234 and 10 as keys, in the real scenarios the key and the factor will



Fig. 4 Tested images: Lena, Cameraman, Peppers and Airport, respectively

be greater than these values. The key used in our method consists of a floating-point number. If we use the first 15 digits after decimal, the possible number of keys will be 10^{15} .

To generate the shares using Eq. 2, any x values less than 250 can be used. For example, for $share_1$ we can use $x = 40$, and for $share_2$, $x = 50$. For simplicity, assume that $share_1$ will use $x = 1$ and $share_2$ will use $x = 2$, etc.

4.1 Creating Parts and Applying Encryption

Using Eq. 2, where three adjacent pixels are taken as coefficients of the equation, parts are created from the image. . These parts create a linear independence relationship using the first part (the solver of the relationship), which are shuffled to hide its information. The results after applying the above steps are shown in Figure 5 where the images show all the parts together. Moreover, to increase the randomness in the shared image, we can use the proposed method in [?], so that shares are completely randomized in an efficient way. Another method to increase randomness is to use a higher share number such as 10, 11, 12, etc., as shown in Figure 5.

4.2 Sharing Phase

The image is now ready to send to the participants in the network. Eq. 2 is used to share the image using the adjacent pixels in the previous images (parts of the images), where the test images are as shown in Figure 6.

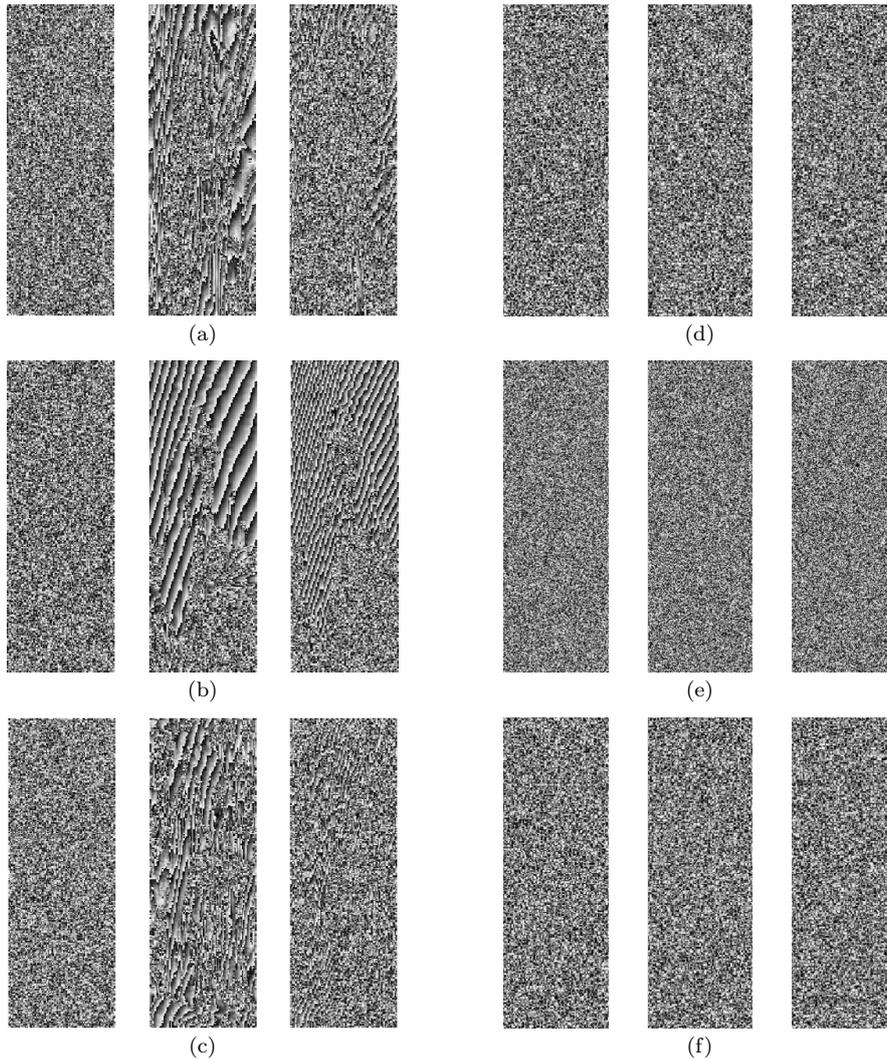


Fig. 5 The resulting shares after phase 2: Lena, Cameraman and Peppers, respectively, with a size of 256×256 pixels, $t_1 = t_2 = 3$ and using different share numbers. For (a)-(c), shares 1, 2, and 3 are used, and for (d)-(f), shares 13, 14 and 15 are used in two phases

4.3 Replacement Phase

After using the replacement phase, distortion in the replacement image is visually noticeable, which increases the randomization between the adjacent pixels. If the replacement phase is not used, the result will not be completely randomized. Two images and two cases are used in the proposed method: a black image and a white image, and with and without replacement mode, as shown in the Figures 7-8. Note that, in Figure 8, even though the original image is uniformly white, part 1 in phase 2 (Figure 8a) is not uniform. This is due to the chaotic encryption which is applied only to the first part.

4.4 Reconstruction Phase

To reconstruct the image, the minimum number of participants should collaborate together, using Lagrange interpolation to retrieve the parts of the image. The first part is then decrypted using the reverse order of the encryption algorithm after generating the chaotic values. The

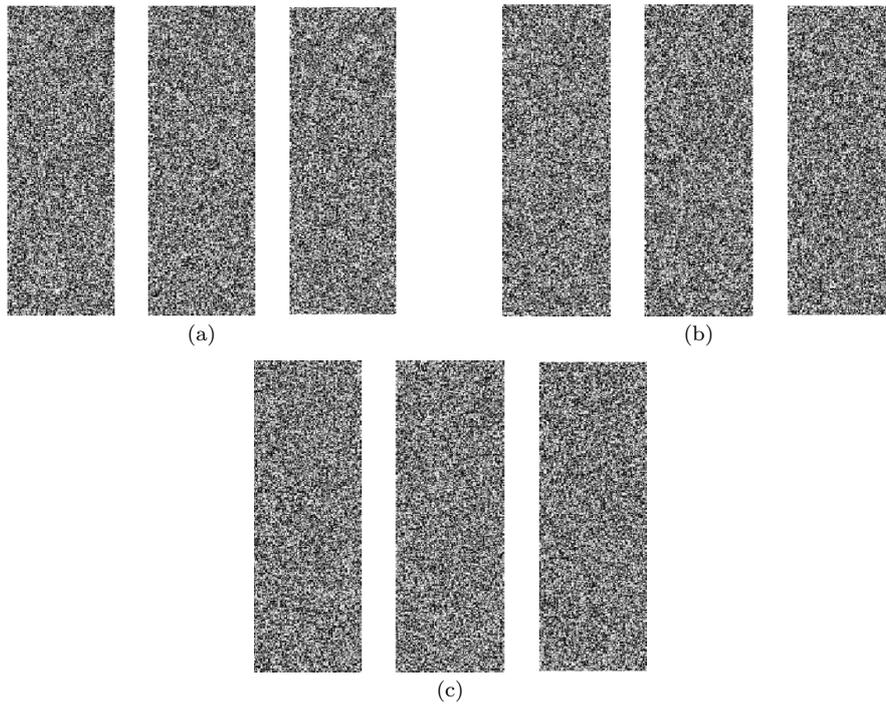


Fig. 6 The resulting shares after phase 3: Lena, Cameraman and Peppers, respectively, with a size of 256×256 pixels, $t_1 = t_2 = 3$. In two sharing methods, in phases 2 and 3, shares 1, 2, and 3 are used

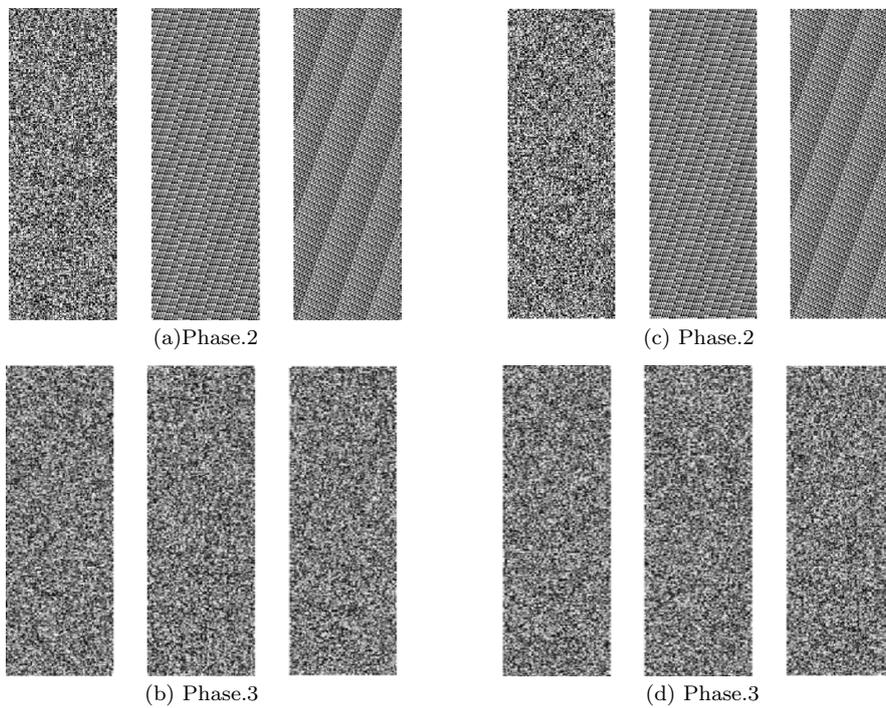


Fig. 7 With replacement phase after phase 2 and phase 3, for white and black images, respectively, $t_1 = t_2 = 3$. In two sharing methods, in phases 2 and 3, shares 1, 2, and 3 are used

decrypted part solves the linear independence relationship with the other parts using the equation below.

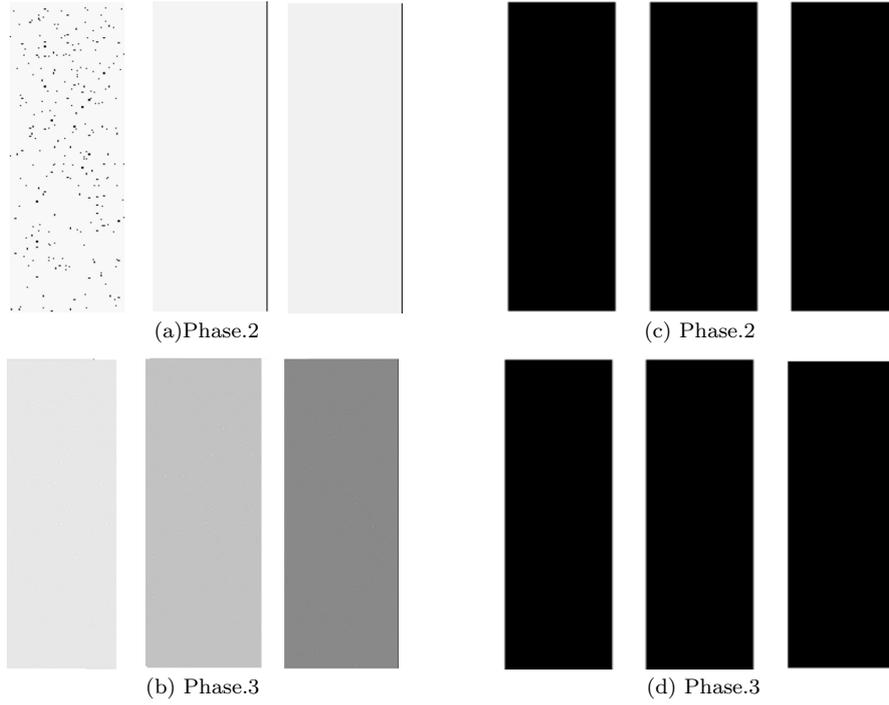


Fig. 8 Without replacement phase after phase 2 and phase 3, for white and black images, respectively. $t_1 = t_2 = 3$, where in the two sharing methods, in phases 2 and 3, shares 1, 2, and 3 are used

$$Part_{no}(r, c) = (Part_{no}(r, c) - Part_1(r, c)) \bmod 251 \quad \forall no \in 2, 3, \dots, t_1 \quad (9)$$

After that, the resulting parts are used to create the original image by using Lagrange interpolation and row and column replacement using the equation below. Finally, the padded pixel is removed.

$$I(i, j) = ((I(i, j)((i + j) \bmod 251)) \bmod 251) \quad (10)$$

For brevity, we used the cameraman image with $t = 3$ and $t = 6$, respectively, where the number of parts is equal to the number of shares as shown in Figure 9.

4.5 Number of Shares

The part is the image share after applying SSS in phase 1, where the share is the image share after applying SSS in phase 3 as shown in Figure 2. Also, t_1 and t_2 are parameters used to denote the number of parts and shares, respectively. The parts are used to create a linear independence relationship with other parts. In order to increase the randomness of the proposed method, the parts will then be combined to create a new image. Phase 3 uses the new image as input to create shares using SSS. Therefore, the relationship between two parameters is that increasing the number of parts increases the randomization and enhances the shares in phase 3.

4.5.1 $t_1 = t_2$ test

In this experiment, the number of shares are changed. The test will consider five cases: when the number of shares is three, four, five, six and seven. The experiment results show that

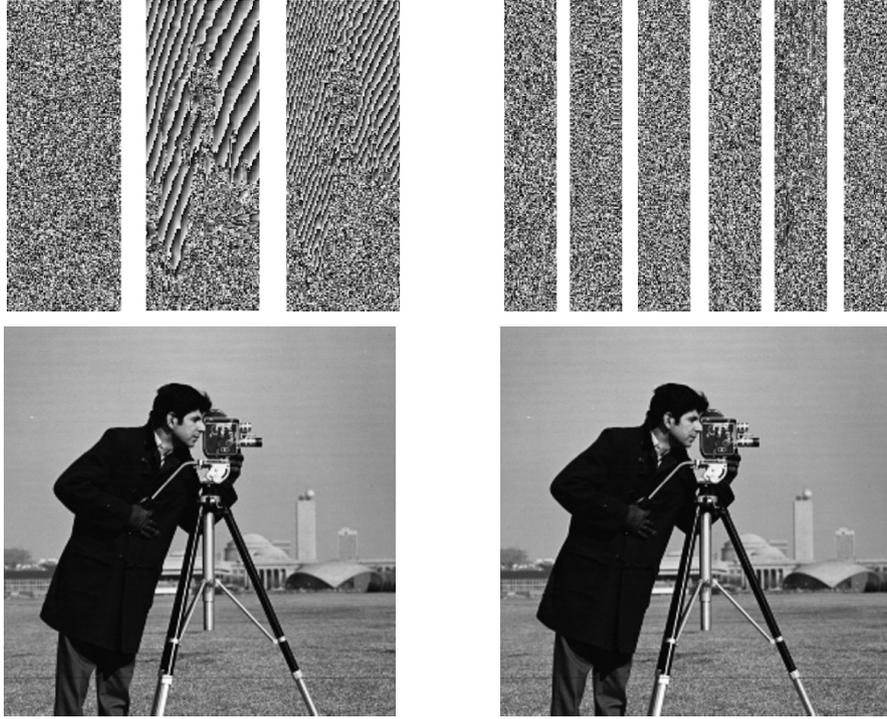


Fig. 9 Reconstruction phase for the cameraman image using two different shares and part numbers

Table 1 The simulation results in seconds for three phases, where Cameraman is used with a size of 256×256 , where $t_1 = t_2 = t$

Number of shares t	Phase.1			Phase.2		Phase.3	
	Replacement	Secret sharing $t_1 = t$	Linear independence	Encryption-Shuffling	Secret Sharing $t_2 = t$		
3	0.0053	0.0174	0.0005	0.0533	0.0172		
5	0.0053	0.0369	0.0007	0.0316	0.0364		
7	0.0053	0.0583	0.0008	0.0225	0.0581		
10	0.0053	0.0904	0.0008	0.0158	0.0891		

when the number of shares increases, the resulting images in phases 2 and 3 will completely randomize. For brevity, only the Cameraman image is used in this test. Figure 10 shows the resulting images after phase 2 using different numbers of shares.

The simulation results after each step are calculated using different numbers of shares, (3, 5, 7, and 10 as shown in Table 1), where all time values are in seconds. The results revealed that the replacement and linear independence phases are more time efficient than the secret sharing phases. When the number of shares increases, the sharing time increases and the encryption time decreases. Hence, the image part size is decreased to $\frac{M \times N}{t}$ depending on t .

4.5.2 $t_1 \neq t_2$ test

Table 2 shows the phases speed if $t_1 \neq t_2$ and t_1 is always equal to 3. This system has better performance than the one shown in Table 1, since the images in phases 1 and 2 are the same in all tested cases. In order to show the system behavior after applying phase 3 if $t_1 \neq t_2$, the Cameraman image is used, as shown in Figure 11. In this method, $t_1 = 3$ and $t_2 = 4, 5, 6$ and 7. The numbers of shares are 11, 12, 13 for the first level in phase 2, and 1, 2 and 3 for the second sharing level in phase 3. Rather than using Eq.8, since it depends on the number of parts in phases 1 and 2, Eq.11 is used after constructing S in phase 2, where the adjacent pixels after padding are used as equation coefficients. The results show that the images are randomized very well.

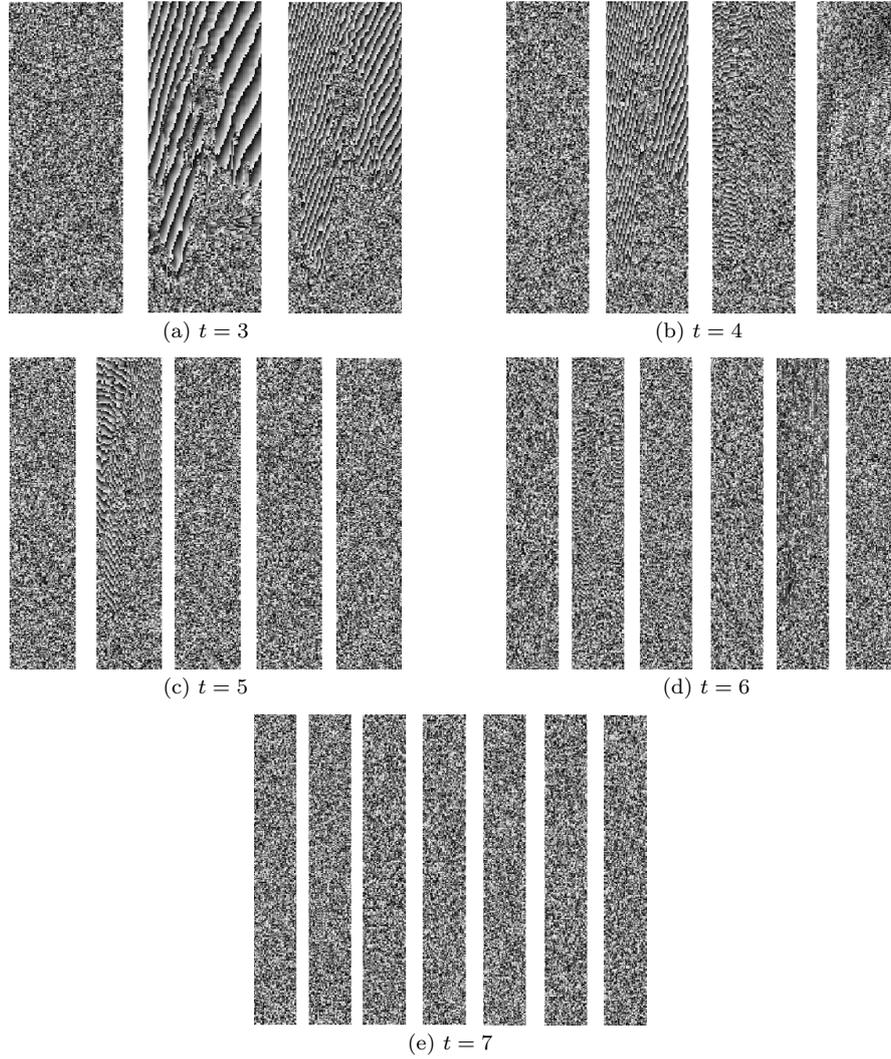


Fig. 10 Cameraman image shares after applying phase 2, where $t = 3, 4, 5, 6, 7$, respectively. $t_1 = t_2 = t$, where, in two sharing methods, in phases 2 and 3, shares 1, 2, and 3 are used

Table 2 The simulation results in seconds for three phases, where Cameraman is used with a size of 256×256 , where $t_1 = 3, t_2 = t$

Number of shares t	Phase.1			Phase.2		Phase.3	
	Replacement	Secret $t_1 = 3$	sharing	Linear independence	Encryption-Shuffling	Secret $t_2 = t$	Sharing
3	0.0053	0.0174		0.0005	0.0533	0.0172	
5	0.0053	0.0174		0.0005	0.0533	0.0364	
7	0.0053	0.0174		0.0005	0.0533	0.0581	
10	0.0053	0.0174		0.0005	0.0533	0.0891	

$$Part_{no}(r, c) = (S(i, j) + S(i, j + 1)x + S(i, j + 2)x^2 \cdots + S(i, j + n)x^n) \pmod{251} \quad (11)$$

$$\forall i, r \in 1, 2, 3 \dots n, j \in 1, 2, 3 \dots m, no \in 1, 2, \dots t_2 \text{ and } c \in 1, 2, 3 \dots m/t_2$$

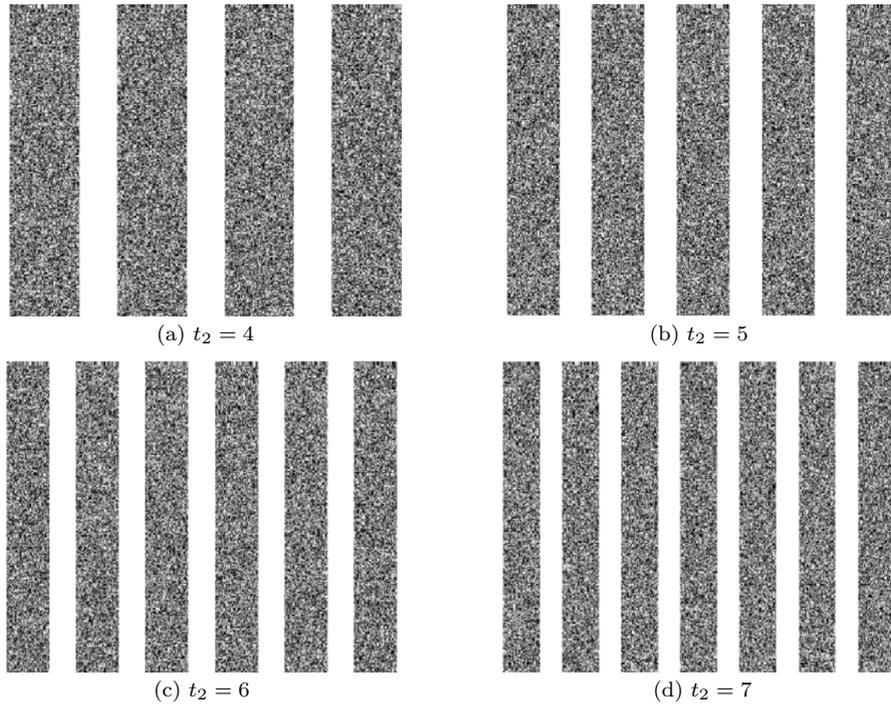


Fig. 11 Cameraman image shares after applying phase 3, where $t_1 = 3, t_2 = t, t = 4, 5, 6, 7$, respectively. For the first level, shares 11, 12, and 13 and in the second level shares 1, 2, and 3 are used

4.5.3 Optimal t_1 test

In this test, different images of different sizes and different numbers of shares in phase one are proposed, in order to find the optimal value of t_1 depending on the image size. The Lena image with sizes of 256×256 and 512×512 and the Airport image with a size of 1024×1024 are used. For these images, t_1 is between 2 and 20 (overall behavior after 20 gradually increases) and t_2 is equal to 3. Figure 12 show the execution times using different sizes and numbers of shares, where the optimal value when $t_2 = 3$ in all cases is four for a 256×256 image and five for 512×512 and 1024×1024 images.

4.5.4 Size test

In this experiment, three images of different sizes are used to compare the proposed method with four encryption algorithms. Any encryption method is divided into two main steps $[\text{?}, \text{?}, \text{?}, \text{?}, \text{?}]$, shuffling and replacement, to change the positions of pixels and to change the values of pixels, respectively. Therefore, the use of two methods achieves high system performance and yields better image results. In this section, four encryption algorithms are proposed: one time pad using two chaotic values, shuffling using two chaotic values, the concatenating method using three chaotic values and advanced encryption standard (AES). The proposed method used three cases, by changing the t_2 value to 3, 4 and 5. In Table 3, the results revealed that in all cases, the proposed method is better than the standard encryption algorithms except for the case where $t_2 = 7$, where the execution time is slower, but one time pad performance is inferior to the proposed method.

Further, to analyze the performance of reconstruction step in the proposed method, we performed various tests using different image sizes and by changing the number of shares (t_2 , number of participants). Table 4 shows the results. As shown in the table, the reconstruction time increases with the number of image shares.

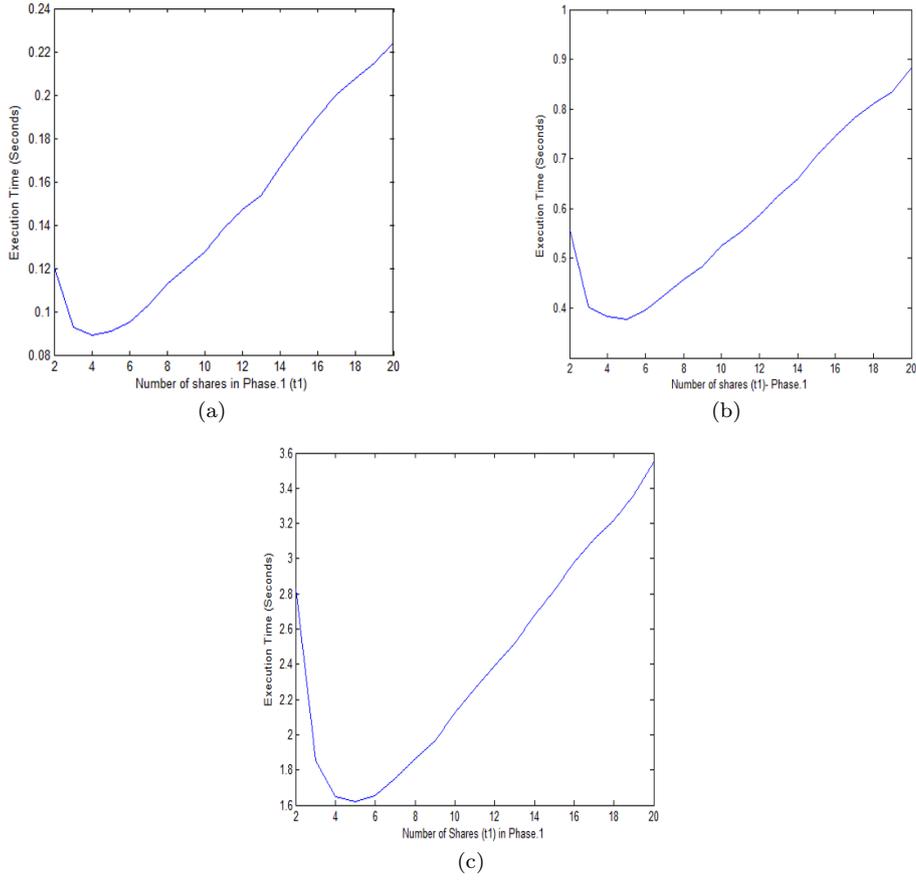


Fig. 12 The t_1 optimal value using $t_2 = 3$ where (a) is Lena 256×256 , (b) Lena 512×512 is and (c) is Airport 1024×1024 , $t_2 = 3$

Table 3 The share creation simulation results in seconds for different methods using different image sizes, where $t_1 = 3$, $t_2 = t$.

Tested method	Image size		
	256×256 Lena	512×512 Lena	1024×1024 Airport
Proposed method, $t_2 = 3$	0.0913	0.3725	1.6240
Proposed method, $t_2 = 5$	0.1109	0.4486	1.9021
Proposed method, $t_2 = 7$	0.1327	0.5351	2.2615
One time pad-two chaotic values	0.1152	0.4052	1.634
Shuffling -two chaotic values	0.1301	0.5277	2.0662
Shuffling and one time pad- Three chaotic values	0.1877	0.7303	2.8832
AES with 1 round	3.179	12.678	50.952
AES with 9 rounds	26.525	106.544	427.257

4.6 Encryption Analysis

4.6.1 Keys space analysis

The strength of any encryption algorithm depends on the key space. If the key space is very large, then a brute force attack is not possible. Therefore, the key space is calculated depending on the sensitivity of the key (small changes in the decrypted key should generate the wrong decrypted image). In our method, the key space is 10^{15} ($\approx 2^{45}$), which is sufficiently large for any brute force attack.

Table 4 The reconstruction simulation results in seconds for different methods using different image sizes, where $t_1 = 3$, $t_2 = t$.

Tested method	Image size		
	256 × 256 Lena	512 × 512 Lena	1024 × 1024 Airport
Proposed method, $t_2 = 3$	0.1681	0.7641	2.8213
Proposed method, $t_2 = 5$	0.2641	0.8662	3.3264
Proposed method, $t_2 = 7$	0.3483	0.9762	4.5123
One time pad-two chaotic values	0.2914	0.8135	2.8151

4.6.2 Information entropy analysis

The entropy is used to calculate the randomness of specific data, where, the true random variable should generate 2^8 symbols with equal probability and the entropy value should equal 8. To check the randomness of our random cipher image, the following equation is used [?]:

$$H(s) = \sum_s P(S_i) \log \frac{1}{(P(S_i))} \quad (12)$$

Where $P(S_i)$ represents the probability of symbol S_i . In our tests, the average entropy of the Lena, Cameraman and Peppers cipher images are 7.9624, 7.9628 and 7.9625, which are very close to the optimal value. Hence, an entropy attack is not feasible.

4.6.3 Histogram analysis

An ideal encryption algorithm should provide a uniformly distributed image histogram, rendering the histogram useless to attackers. In our tests, Figure 13 describes the histogram of the original images the Lena, Cameraman and Peppers images (on left) and the histogram of their shares after applying the proposed method (on right). Since the histogram after applying the proposed method (on right) is uniformly distributed, it does not provide any information about the secret image. Therefore, we deduce the algorithm is secure against the histogram attack.

4.6.4 Correlation analysis

In the past, some algorithms have succumbed to attacks due to the attacker using correlations between two adjacent pixels. As such, we have performed the correlation analysis on the proposed method while taking into consideration all possible adjacent cases (vertical, horizontal and diagonal). In order to find a correlation between adjacent pixels the correlation coefficient is calculated using the following formula [?]:

$$r = \frac{Cov(x, y)}{\sqrt{D(x)}\sqrt{D(y)}} \quad (13)$$

$$D(x) = \frac{1}{R} \sum_{i=1}^R (x - \bar{x})^2 \quad (14)$$

$$Con(x, y) = \frac{1}{R} \sum_{i=1}^R (x - \bar{x})(y - \bar{y}) \quad (15)$$

where, R is the total number of randomized pairs, and x and y are the two vectors that contain the x values and y values of the pair in the tested image, respectively. Table 5 shows the correlation coefficients between two adjacent pixels in all possible cases (vertically, horizontally and diagonally) of the plain-text images and cipher images. The results revealed that the proposed method effectively randomized the pixels.

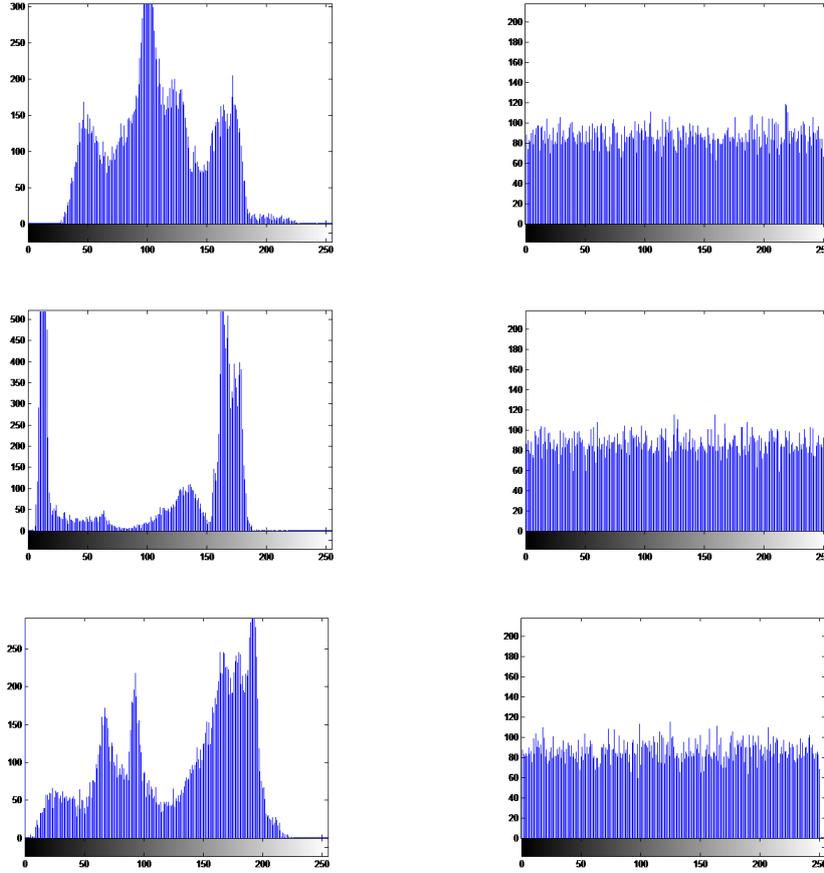


Fig. 13 Histogram of the first part only for Lena, Cameraman and Peppers and their cipher images, respectively

Table 5 Correlation coefficients of adjacent pixels Image

Coefficient	lena		Cameraman		Peppers	
	Plain	Cipher	Plain	Cipher	Plain	Cipher
Vertical	0.9304	0.0152	0.9634	-0.0084	0.9617	-0.0229
Horizontal	0.8786	0.0101	0.9409	0.0097	0.9525	0.0072
Diagonal	0.8558	-0.0113	0.9026	-0.0032	0.9123	-0.0143

4.6.5 Plaintext sensitivity analysis

If the cipher image is not sensitive to changes to the plaintext then a cryptanalyst can obtain useful information from the encrypted image to check the two criteria used: NPCR (Number of Pixel Change Rate) and UACI (Unified Average Changing Intensity). NPCR is defined as a percentage of different numbers of pixels between two cipher images and UACI is defined as the average intensity of differences between two cipher images, as defined in the following equations [?]:

$$NPCR = \frac{\sum_{i,j} D(i,j)}{M \times N} \times 100 \quad (16)$$

$$UACI = \frac{1}{M \times N} \sum_{i,j} \frac{|C_1(i,j) - C_2(i,j)|}{255} \times 100 \quad (17)$$

where $M \times N$ is the size of the cipher images and C_1 and C_2 are two different cipher images encrypted using different keys, and where $D(i, j)$ is defined as follows:

$$D(i, j) = \begin{cases} 0 & C_1(i, j) = C_2(i, j) \\ 1 & C_1(i, j) \neq C_2(i, j) \end{cases} \quad (18)$$

After calculations, the average NPCR and UACI of the Lena, Cameraman and Peppers cipher images are: NPCR = 99.5231 and UACI = 32.5835, NPCR = 99.5094 and UACI = 32.5662 and NPCR = 99.5322 and UACI = 32.9155, respectively. This shows that our algorithm is robust to known-plaintext attacks.

5 Security Analysis and Discussion

In this section, we analyze the security of the application of the proposed method on images. While there remain a number of questions regarding the applicability of the method in real networks, the security analysis focuses mainly on the efficiency of the method. The security analysis is formulated as answers to the following questions:

- Does the approach using encryption, linear independence and replacement negatively impact the speed of the proposed method?
- Can the proposed method send data over unsecured channels?
- Can the priority concept apply in the proposed method?
- What is the impact of using encryption in the proposed method?
- What is the probability of the method succumbing to an attack?

While using the linear independence and replacement approach may compromise the speed of the method, the replacement and linear independence approach uses column and row indices, therefore no data is generated and the overhead of the operation is minimized. Moreover, encryption will use the shuffling approach only on part of the image. The shares will then be sent to the participants on the network using one channel and the dealer assumes responsibility for ensuring the security over the network by sending shares and keys to the participants.

However, many applications require a priority concept in order to share data between different administrative participants such as a president, vice-president and users with high privilege (or without any privilege, depending on particular situation). Therefore, the proposed method can apply this concept by using the chaotic key. For example, the president can take a chaotic key with t -shares, whereas the vice president can only take the chaotic key with one or two shares, and so on. However, using encryption in the proposed method may have an impact, which is discussed in the following subsections.

5.1 The impact of using encryption

The encryption is used for two main reasons: to generate an encrypted share and to encrypt the initial conditions of the chaotic model. The encryption in the two cases does not have any significant impact on the computation time of the algorithm since only part of the image is encrypted. Furthermore, this impact can be minimized and controlled depending on the size of the encrypted share, which depends on the size of the original image and the number of shares. While the encryption step does not add much to the computational complexity, it is very useful in providing higher security, which is analyzed as follows.

To calculate the possibility of proposed method succumbing to an attack, vulnerabilities in the method should be defined first. The size of the encryption key and the possibility of retrieving the shuffled part using the guessing approach (attempting to test all possible cases of the encrypted image) are considered first. Therefore, the size of the key is approximately 10^{15} ($\approx 2^{45}$), which is sufficiently large to resist any brute force attack. If the attacker uses a guessing approach to retrieve the shuffled image, the probability of retrieving the original part is equal to $\frac{1}{(Column \times Row)!}$. In our experiments, the size of the shares is equal to 256×86 and

the probability of retrieving the shuffled part is approximately $\frac{1}{2^{144}}$, which is strong enough to counter any guessing. To simplify the analysis, the degree of weakness \mathcal{W} of the algorithm depends on the values of the three factors as shown in Eq. 19. In this equation, RSA_key is the size of the RSA key used to encrypt the chaotic key, $Part_size!$ is the factorial of the part size and finally $chaotic_key$ is the size of the encryption algorithm for the solver part. The probability \mathcal{P} of the proposed method being successfully attacked is inverse of the degree of weakness, as shown in Eq. 20.

$$\mathcal{W} = \min(2^{RSA_key}, 2^{(8 \times Part_size!)}, 2^{chaotic_key}) \quad (19)$$

$$\mathcal{P} = \frac{1}{\mathcal{W}} \quad (20)$$

5.2 The benefits of using the proposed method

- The proposed method can send all the shares together over unsecured channels to the users with the chaotic key, which solves a critical issue in the image sharing system.
- The proposed method increases the randomness of the shares by creating the relations between the shares and by using the replacement phase.
- The proposed method eliminates the impact of encrypting the whole image and still utilizes the main concept of image sharing in a large network.
- The proposed method can be applied in the wireless camera network scenarios to secure data shared among different nodes.

5.3 Sharing over unsecure channels

The proposed method achieves the following goals which were not met by the previous methods:

- Sending the image shares in a secure way.
- Enhancing and improving the speed of the system by encrypting part of the image.
- Keeping the main concept of the image sharing without modifying it.

The security of shared images is analyzed using the following two propositions. The assumed parameters are:

- S: Secret data
- R: Random values
- O: Original Data
- S-space: The attacker can sniff no more than $n - 1$ shares over the channels.
- U-space: The attacker can sniff more than or equal to n shares over the channels.

Proposition 1 *The SSS which uses one secret value $s \in S$ and random numbers $a_i \in R$ is secured over S-space and unsecured (can be attacked quickly) over U-space.*

Proof The SSS is secured over the S-space since the attacker cannot retrieve information if less than $n - 1$ is used, whereas in U-Space the attacker can retrieve the information because all the shares are used in the retrieving process.

Proposition 2 *The proposed method is secured over S-space and U-space.*

Proof The proposed method contains three phases. In the first and second phases, the image is divided into parts, which are used to create a linear independence between parts. Then, the first part of the image will be encrypted to hide the solver information. In the last phase, the data is shared. In addition, after the final sharing in phase 3, the pixel values are changed completely because the replacement, sharing and the linear independence relationship (in phase.2) are used, which randomize the pixels in an effective way. Mathematically, this phenomenon can be explained as follows.

Table 6 Computational complexity of different algorithm parts in the proposed method

Algorithm part	Description / Number of nested loops
Replacement	The pixels image is replaced and changed to a new value. Two nested loops depend on the image size.
Sharing	The image uses SSS to generate the shares. Two nested loops depend on the image size.
Linear independence	The first share is used to create a linear independence relationship with the other parts. Two nested loops depend on the image size.
Shuffling	The first part is shuffled using the chaotic keys. Two nested loops depend on the $M/t_1 \times N$.
Sharing	The image is combined and then SSS is used. Two nested loops depend on the image size.

$$\begin{aligned}
\text{Replacement } (O) &\rightarrow RO && \forall O \in S, RO \in R \\
\text{Sharing } (RO) &\rightarrow SO && \forall O \in S, SO \in R \\
\text{Linear independence } (SO) &\rightarrow LO && \forall O \in S, LO \in R \\
\text{Shuffling } (\text{First Part in } LO) &\rightarrow SLO && \forall O \in S, SLO \in R
\end{aligned}$$

Therefore, Linear Independence (*First Part*, *Parts_i*) in $SLO \cap$ Linear Independence (*FirstPart*, *Parts_i*) in $LO = \emptyset$ where i is the number of parts

Finally, phase 3 shares the data with the participant using the following equation.

$$slo_0 + slo_1x^1 + slo_2x^2 + \dots + slo_nx^n \quad \forall o \in S, slo_i \in R \quad (21)$$

The proposed method used three steps to hide the information and to increase the randomness in the output data, so the original data (secret data) is changed to random space (R) rather than the original space (S).

5.4 Computational complexity

The computational complexity of the proposed method is $O(n^2)$, where n^2 is the size of the image. Various steps to analyze the computational complexity are provided in the Table 6.

6 Conclusion

In this paper, the problem of sending shares over unsecured channels was discussed. The method used a linear independence relationship between the shares to create a security concept to hide the shares' information from attackers. Furthermore, to protect the transmitted shares, the encryption algorithm is used with the chaotic key to shuffle and to replace the pixel values. Then, all the shares use a modified version of the SSS method to add a level of randomness between the shares. Finally, the shares with the chaotic key are sent together over an unsecured channel to the participants. The participants then assume the responsibility of retrieving the original image by using Lagrange interpolation and the chaotic key. To validate the proposed method, different tests are used: visual tests, number sharing tests and encrypted part tests. The simulation results indicate that the proposed method is efficient and strong against different types of attacks and achieves the main objective of image sharing transmitted over unsecured channels.

References

1. Chih-Ching Thien and Ja-Chen Lin. Secret image sharing. *Computers & Graphics*, 26(5):765–770, 2002.
2. Saeed Alharthi and Pradeep K Atrey. Further improvements on secret image sharing scheme. In *Proceedings of the 2nd ACM workshop on Multimedia in forensics, security and intelligence*, pages 53–58. ACM, 2010.
3. Chunqiang Hu, Xiaofeng Liao, and Di Xiao. Secret image sharing based on chaotic map and chinese remainder theorem. *International Journal of Wavelets, Multiresolution and Information Processing*, 10(03), 2012.
4. Jun Lang. A no-key-exchange secure image sharing scheme based on shamir’s three-pass cryptography protocol and the multiple-parameter fractional fourier transform. *Opt. Express*, 20(3):2386–2398, Jan 2012.
5. Chin-Chen Chang, Pei-Yu Lin, Zhi Hui Wang, and Ming Chu Li. A sudoku-based secret image sharing scheme with reversibility. *Journal of Communications*, 5(1):5–12, 2010.
6. Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
7. George Robert Blakley. Safeguarding cryptographic keys. In *Managing Requirements Knowledge, International Workshop on*, page 313. IEEE Computer Society, 1899.
8. Ran-Zan Wang and Shyong-Jian Shyu. Scalable secret image sharing. *Signal Processing: Image Communication*, 22(4):363–373, 2007.
9. C-HT Yang, Yuan-Hui Huang, and Jhih-Hao Syue. Reversible secret image sharing based on shamir’s scheme with discrete haar wavelet transform. In *Electrical and Control Engineering (ICECE), 2011 International Conference on*, pages 1250–1253. IEEE, 2011.
10. Lee Shu-Teng Chen, Wei-Kai Su, and Ja-Chen Lin. Secret image sharing based on vector quantization. *International Journal of Circuits, Systems and Signal Processing*, (3):137–144, 2009.
11. Yung-Yi Lin and Ran-Zan Wang. Scalable secret image sharing with smaller shadow images. *Signal Processing Letters, IEEE*, 17(3):316–319, 2010.
12. Lin Dong, Daoshun Wang, Min Ku, and Yiqi Dai. (2, n) secret image sharing scheme with ideal contrast. In *Computational Intelligence and Security (CIS), 2010 International Conference on*, pages 421–424. IEEE, 2010.
13. P Devaki and G Raghavendra Rao. Lossless reconstruction of secret image using threshold secret sharing and transformation. *International Journal of Network Security & Its Applications*, 4(3), 2012.
14. Wen-Pinn Fang. Multi-layer secret image sharing. In *Intelligent Information Hiding and Multimedia Signal Processing (IHH-MSP), 2010 Sixth International Conference on*, pages 59–61. IEEE, 2010.
15. Shang-Lin Hsieh, I-Ju Tsai, Chung-Ping Yeh, and Chia-Ming Chang. An image authentication scheme based on digital watermarking and image secret sharing. *Multimedia Tools and Applications*, 52(2-3):597–619, 2011.
16. *Encyclopedia of Mathematics*, chapter Linear independence. Springer, 2001.
17. Peter Grassberger and Itamar Procaccia. Measuring the strangeness of strange attractors. *Physica D: Nonlinear Phenomena*, 9(1):189–208, 1983.
18. Hongfeng Zhu, Tianhua Liu, Dan Zhu, and Haiyang Li. Robust and simple n-party entangled authentication cloud storage protocol based on secret sharing scheme. *Journal of Information Hiding and Multimedia Signal Processing*, 4(2):110–117, 2013.
19. Valery Korzhik, Guillermo Morales-Luna, and Ksenia Nebaeva. The capacity of a stegosystem for the noisy attack channel. *Journal of Information Hiding and Multimedia Signal Processing, Ubiquitous International*, 3(2):205–211, 2012.
20. Yueh-Hong Chen and Hsiang-Cheh Huang. Coevolutionary genetic watermarking for owner identification. *Neural Computing and Applications*, pages 1–8.
21. S. Poonkuntran and R. S. Rajesh. Chaotic model based semi fragile watermarking using integer transforms for digital fundus image authentication. *Multimedia Tools and Applications*, 68(1):79–93, 2014.
22. G. Zanin, A. Mei, and L.V. Mancini. A secure and efficient large scale distributed system for data sharing. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, pages 27–27, 2006.
23. Pradeep K. Atrey, Saeed Alharthi, M. Anwar Hossain, Abdullah AlGhamdi, and Abdulmotaleb El Saadik. Collective control over sensitive video data using secret sharing. *Multimedia Tools and Applications*, 2013. 10.1007/s11042-013-1644-0.
24. Zhang Yuan-Biao and Wang De. An image encryption and sharing algorithm based on chaos and indeterminate equation. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, pages 1–4, 2009.
25. E Xu, Liangshan Shao, Guanghui Cao, Yongchang Ren, and Tao Qu. A new method of information encryption. In *Computing, Communication, Control, and Management, 2009. CCCM 2009. ISECS International Colloquium on*, volume 4, pages 583–586. IEEE, 2009.
26. Runhua Shi, Hong Zhong, Liusheng Huang, and Yonglong Luo. A (t, n) secret sharing scheme for image encryption. In *Image and Signal Processing, 2008. CISP’08. Congress on*, volume 3, pages 3–6. IEEE, 2008.
27. Siddharth Malik, Anjali Sardana, and J Jaya. A keyless approach to image encryption. In *Communication Systems and Network Technologies (CSNT), 2012 International Conference on*, pages 879–883. IEEE, 2012.
28. Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Foundations of Computer Science, 1987., 28th Annual Symposium on*, pages 427–438. IEEE, 1987.

29. Hsiang-Cheh Huang, Shu-Chuan Chu, Jeng-Shyang Pan, Chun-Yen Huang, and Bin-Yih Liao. Tabu search based multi-watermarks embedding algorithm with multiple description coding. *Information Sciences*, 181(16):3379–3396, 2011.
30. J-S Pan, Y-C Hsin, H-C Huang, and K-C Huang. Robust image watermarking based on multiple description vector quantisation. *Electronics letters*, 40(22):1409–1410, 2004.
31. Ying-yu Cao and Chong Fu. An image encryption scheme based on high dimension chaos system. In *Intelligent Computation Technology and Automation (ICICTA), 2008 International Conference on*, volume 2, pages 104–108. IEEE, 2008.
32. Jiang Delei, Bai Sen, and Dong Wenming. An image encryption algorithm based on knight’s tour and slip encryption-filter. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 1, pages 251–255. IEEE, 2008.
33. Weihua Zhu and Ying Shen. Encryption algorithms using chaos and cat methodology. In *Anti-Counterfeiting Security and Identification in Communication (ASID), 2010 International Conference on*, pages 20–23. IEEE, 2010.
34. Tang Liangrui, Li Jing, and Sun Yi. Image encryption based on a four-dimensional chaotic system. In *Natural Computation, 2009. ICNC’09. Fifth International Conference on*, volume 5, pages 487–491. IEEE, 2009.
35. Xiaogang Jia. Image encryption using the ikeda map. In *Intelligent Computing and Cognitive Informatics (ICICCI), 2010 International Conference on*, pages 455–458. IEEE, 2010.