CRYPTANALYSIS OF A SYMMETRIC COLOR IMAGE ENCRYPTION WITH ONE-ROUND ENCRYPTION

Hoang Xuan Thanh[1] and Thang Manh Hoang[2]

School of Electronics and Telecommunications, Hanoi University of Science and Technology, Hanoi, Vietnam, Tel.: +84 43692242, e-mail: thang.hoangmanh@hust.edu.vn

Received Date: December 12, 2015

Abstract

We present the security weakness of encryption algorithm in the form of substitution-permutation network with multiple rounds of permutation and single round of diffusion proposed by W. Zhang et al. The types of chosen-plaintext and chosen-ciphertext attacks are successful against the cryptosystem, and the equivalent versions of keys for encryption and decryption are restored. The security analysis suggests that encryption using substitution-permutation network must be executed more than one encryption round to ensure the security. Our specific examples will demonstrate the cryptanalysis.

Keywords: Chaos-based image encryption, Cryptanalysis

Introduction

For decades, chaotic systems have been employed for security and privacy due to its characteristics of sensitivity on initial conditions, control parameters, psuedo-randomness and ergodicity [1]. Many methods of chaos-based encryption were proposed, including chaos-based image encryption, e.g. [2, 3, 4, 5, 6, 7]. So far, there are various ways in using chaos for designing an encryption (see [8] and therein), e.g. *(i)* in creation of position permutation matrices, *(ii)* in generation of pseudo-random bit sequences for mixing with plaintext, and *(iii)* in production of ciphertext with the use of plaintext as initial condition of chaotic map. However, due to intrinsic security flaws in the design of encryption algorithms, many of cryptosystems have not met basic requirements [9], so those have been broken soon after being proposed, e.g. [10, 8, 11, 12]. The architecture of substitution-permutation network (SPN) is the most prominent in providing high security for data encryption [14, 15]. In fact, chaos-based SPNs are combination of above *(i)* and *(ii)* providing security by means of avalanche characteristics [16]. Specifically, chaos-based permutation is the exchange of pixels in which the location of current pixels are considered as initial vectors of chaotic systems in computation for new locations, e.g. [17,18,2,19,4,20,21]. Chaotic systems can be utilized for the diffusion in some ways, but in most of cryptosystems chaotic systems are used as random sequence generators. Then, random sequences are mixed with plaintext words in various fashions, e.g. [22,23,24,18,25]. So far, there is very limited number of successful attacks on chaos-based substitution permutation networks reported. In the literature, to the best knowledge of the authors, there are only two successful attacks on chaos-based SPNs in the case that one round of encryption is carried out to networks, i.e. in [10, 26]. As presented in [10], the method can be extended to deal with multiple-round encryption, while the work in [10] only performs for one-round cryptosystem.

Intrinsic features of bits distributions of images have been recently investigated and exploited for the purpose of encryption proposed by W. Zhang et al. [24], in which the architecture of SPN was utilized. In this paper, cryptanalysis on a chaos-based cryptosystem is presented. It shows that two types of attacks, chosen-plaintext and chosen-ciphertext, are successful in dealing with the cryptosystem of one-round encryption, and equivalent versions of keys for encryption/decryption are achieved. The specific examples will demonstrate the cryptanalysis.

Description of Image Encryption

A gray level image is a matrix of pixels, in which each pixel is represented by a number of bits. The number of n bits encodes the intensity or gray scale. For example, a 8-bit pixel has 256 gray scales; 0 is black and 255 is white. A 8-bit pixel can be presented by $b_7 b_6 ... b_0$; where b_7 and $b_0$are most significant and least bits, respectively. In the matrix of pixels, location and value of pixels are illustrated by $f(x,y) = b_7 b_6 ... b_0$. A RGB image has three color layers; R (red), G (green), and B (blue). Each layer is considered as a matrix of gray scale. So, the value of pixel at location (x,y) is $f_R(x,y)$, $f_G(x,y)$, and $f_B(x,y)$; corresponding to red, green and blue color layers, respectively. To encrypt a $N \times N$ RGB image as given in [24], the RGB color image is rearranged to exploit intrinsic features of bit distribution. Specifically, 2 most significant bits of every pixel from R, G and B color layers are extracted and merged together to become a $N \times N$ 6-bit gray scale image. Three other $N \times N$ 6-bit images are of 6 least significant bits of pixels. As illustrated in Figure 1, each of four $N \times N$ 6-bit images is a quarter of $2N \times 2N$ square; the square of four quarters is called a matrix in the following text. The resulting $2N \times 2N$ matrix is used for encryption. The encryption algorithm consists of two processes, i.e. confusion and diffusion as shown in Figure 2.

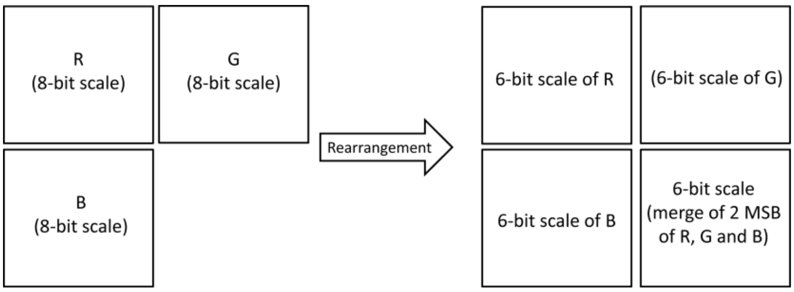


Figure 1. A RGB image is rearranged into a matrix for encryption

At a certain round of encryption, pixel permutation is accomplished by computing new location (x', y') using current (x,y) as an initial vector of chaotic map. In the decryption, inverse permutation is carried out to restore (x,y) using(x', y') as initial vector. In fact, the forward and inverse permutation is successful with the use of bijective two-dimensional chaotic map such as Cat map [29], or Standard map [27, 28] as given in Equation (1), respectively.

$$\begin{cases} x' = (x+y) mod\ N \\ y' = \left(y + k.sin\ \frac{x'.N}{2\pi}\right) mod\ N \end{cases} \quad (1)$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & p \\ q & pq+1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} mod\ N$$

The confusion process consists of a number of permutation rounds. As given in [24], Cat map is used for permutation. The set of system parameters (p,q) of Cat map is considered as part of secret key, which is generated by using the state variable of Logistic map as given in Equation (2).

$$f(x_n) = \alpha x_{n-1}(1 - x_{n-1}) \quad (2)$$

The initial conditions x_0 for Logistic map are $conf\_key_1$ and $conf\_key_2$, respectively, for generation of parameters p and q of Cat map. It is noted that the first 2000 elements of state variable generated by logistic map is unused to ensure randomness in value of p and q. As demonstrated by W. Zhang et al. in [24], the confusion consists of multiple rounds of permutation and that is followed by one-round diffusion process. In addition, different sets of system parameters are used for different rounds of permutation. The steps in the encryption and decryption are illustrated in Figure 2. At the encryption, P is plain image, whereas at the decryption, P is recovered image. C is cipher image. Notations with the prefix of M are for matrix in 2-D, while those with A are for 1-D array. The description for notations and value ranges are written as in Equation (3).

$$Encryption: \begin{cases} P = \{f(x,y); f(x,y) \in [0,255], \forall x,y \in [1,N]\} \\ M_E = \{f(x,y); f(x,y) \in [0,63], \forall x,y \in [1,2N]\} \\ MP_E = \{f(x,y); f(x,y) \in [0,63], \forall x,y \in [1,2N]\} \\ A_E = \{ac(i); ac(i) \in [0,63], i \in [1,4N^2]\} \\ AD_E = \{cipher\_d(i); cipher\_d(i) \in [0,63], i \in [1,4N^2]\} \\ MT_E = \{f(x,y); f(x,y) \in [0,63], \forall x,y \in [1,2N]\} \\ \mathrm{C} = \{f(x,y); f(x,y) \in [0,255], \forall x,y \in [1,2N]\} \end{cases}$$

(3)

$$\mathrm{De}cryption: \begin{cases} C = \{f(x,y); f(x,y) \in [0,255], \forall x,y \in [1,N]\} \\ M_D = \{f(x,y); f(x,y) \in [0,63], \forall x,y \in [1,2N]\} \\ A_D = \{cipher\_d(i); cipher\_d(i) \in [0,63], i \in [1,4N^2]\} \\ AD_D = \{ac(i); ac(i) \in [0,63], i \in [1,4N^2]\} \\ MT_D = \{f(x,y); f(x,y) \in [0,63], \forall x,y \in [1,2N]\} \\ MP_D = \{f(x,y); f(x,y) \in [0,63], \forall x,y \in [1,2N]\} \end{cases}$$

As illustrated in Figure 2(a), the plain image is rearranged into $2N \times 2N$ matrix, M_E, in the form given in Figure 1. The $2N \times 2N$ matrix M_E is permuted to obtain the matrix, M_{PE}, then M_{PE} is transformed into the 1-dimensional array A_E of $4\mathrm{N}^2$ elements. The diffusion process is carried out on A_E in the fashion of domino, and the 1-dimensional array AD_E is achieved. The cipher word for i^{th} element is computed by

$$\begin{cases} temp_1 = cipher\_d(i-1) \\ temp_2 = rand_1(temp_1) \\ cipher\_d(i) = ([ac(i) \oplus rand_2(temp_2)] + rand_3(i)) \; mod \; 64 \end{cases} \quad (4)$$

where $rand_1$ and $rand_2$ are random number arrays of 64 elements generated by Logistic map, whose values of elements are in the range of 0 and 63. The Logistic map as given in Equation (2) is employed, and the initial conditions of Logistic map for generation of $rand_1$ and $rand_2$ are key_d_2 and key_d_3. The $temp_1$ and $temp_2$ are two temporary variables, and used as an indices in calling values of arrays $rand_2$ and $rand_1$. The first element of 1-dimensional array $temp_1(\equiv cipher\_d(0))$ takes the initial value of $temp_1 = [\alpha \times key\_d_1 \times (1 - key\_d_1)] \times 1000]$ mod 64. Similarly, $rand_3$ is an array of $2N \times 2N$ elements generated by Logistic map using initial condition of $\_d_4$. Then, 1-dimensional array AD_E is transformed into $2N \times 2N$ matrix. The $2N \times 2N$ matrix MT_E is rearranged back into the format of RGB image which is the cipher image C. As illustrated in Figure

2(b), the process for decryption is carried out in the reverse way in compared with that for encryption. The cipher image C is rearranged to the $2N \times 2N$ matrix M_D, and then the matrix M_D is transformed into the 1-D array A_D before being inversely diffused to obtain the 1-D array AD_D. It is explicit that the equation for the inverse diffusion process at the decryptor is as

$$\begin{cases} temp_1 = cipher_{d(i-1)} \\ temp_2 = rand_1(temp_1) \\ ac(i) = \begin{cases} [64 + cipher\_d(i) - rand_3(i)] \oplus rand_2(temp_2), cipher\_d(i) < rand_3(i) \\ [cipher\_d(i) - rand_3(i)] \oplus rand_2(temp_2), cipher\_d(i) \geq rand_3(i) \end{cases} \end{cases} \quad (5)$$

It is clear that the secret key consists of $conf\_key_1$, $conf\_key_2$, key_d_1, key_d_2, key_d_3 and key_d_4; those are fraction numbers less than unity. Note that, this secret key is used for generating the encryption and decryption keys, i.e. $rand_1$, $rand_2$, and $rand_3$.

Next, the 1-D array AD_D is transformed back into the $2N \times 2N$ matrix MT_D. Inverse permutation is applied on the $2N \times 2N$ matrix MT_D to have MP_D. The recovered plain image P is achieved by rearranging the $2N \times 2N$ matrix MP_D into the format of RGB image as given in Figure 1.

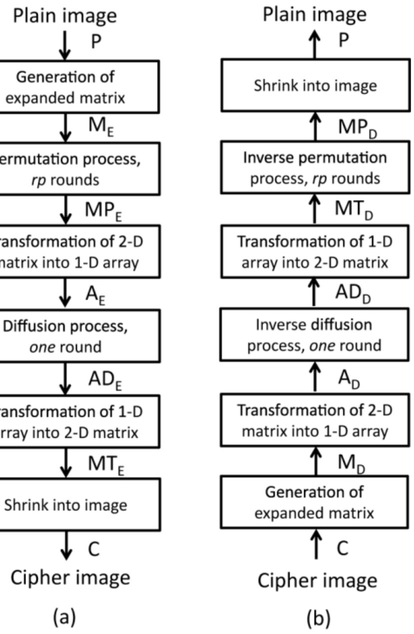


Figure 2. Encryption and decryption. (a) Steps in the encryption, (b) Steps in the decryption

Cryptanalysis of Image Encryption

According to the Kerchoff's principle [14], all the details about a cryptosystem are transparent to all, except for the secret key. Moreover, there are four main classical types of attacks in the order of hardest to easiest as

- Ciphertext-only: The opponent possesses one or more ciphertexts.
- Known-plaintext: The opponent possesses one or more plaintexts, and its corresponding ciphertexts.

- Chosen-plaintext: The opponent can access to the encryption machinery. Some known plaintexts can be chosen for encryption and corresponding ciphertexts are obtained.
- Chosen-ciphertext: The opponent can access to the decryption machinery. Some known ciphertexts can be chosen for decryption and corresponding plaintexts are obtained.

These types of attacks are mainly to recover the plaintext or encryption/decryption keys. The cryptosystem does not provide sufficient security if at least one of the above types of attacks is successful. Let's look closely into the principles in each process of encryption algorithm for the cryptanalysis. Firstly, the confusion process exchanges every pair of pixel values in the plaintext image. In fact, regardless to the number of permutation rounds and progress of permutation, the exchange is carried out using lookup tables for row and column. In this algorithm, the lookup tables are generated by a two-dimensional chaotic map using a certain value set for the secret key; each dimension of the chaotic map is used for a dimension of image. In other words, the goal of confusion attack in the encryptor and/or decryptor is to recover the lookup tables. Secondly, the diffusion process carries out a series of computation to make the ciphertext dependent on both plaintext and encryption keys under an avalanche effect. In this encryption/decryption algorithm as in Equation (4) and (5), the encryption/decryption keys are initial value of $temp_1$, random sequences $rand_1$, $rand_2$, and $rand_3$. It is noted that we do not expect to recover the secret key, but any successful recovery of either partially or fully encryption/decryption keys in any equivalent form, by what the plaintext is fully recovered, is enough to say that the cryptosystem is successfully attacked. This section presents the cryptanalysis using two easiest types of attacks, i.e. chosen-plaintext and chosen-ciphertext. With the chosen-plaintext attack, it is assumed that the attacker can access the encryptor and he can choose suitable plaintexts for encryption and obtains its corresponding ciphertexts for the breaking process. Similar to the chosen-ciphertext attack, the attacker can access the decryptor and suitable ciphertexts are chosen for decryption and its corresponding recovered plaintexts are obtained for the attacking process. In these cases, both encryptor and decryptor are seen as black boxes. It is noted that the cryptosystem is in the form of SPN which consists of multiple rounds of permutation followed by one round of diffusion. Throughout examples in the following text, the number of permutation rounds is of $rp = 5$. In order to visualize the cryptanalysis process, a small RGB image with the size of 5×5 pixels is employed as an example, along with the description for the general case of the RGB image with the size of $N \times N$. In addition, the 2D matrix is used for representing the 1D sequence.

Chosen-plaintext Attack

Attack on Confusion

As mentioned above, the confusion of encryption algorithm performs a number of permutation rounds, thus the goal of confusion attack is to recover the lookup tables, which governs the overall pixel permutations. By taking a close look on the diffusion equation with the forward affect in Equation (4), it is clear that if the value of the i^{th} element in the 1-dimensional array is modified, as a result, it makes changed to values of elements from i to the end of sequence. The affect in value of elements of 1-dimensional array can be tracked in its cipher image and vice versa in the process of diffusion as given in Equation (5). This is considered as the basis for the confusion attack. The attack is illustrated in Figure 3 that an arbitrary image P_{arb} is chosen for encryption and the cipher image C_{arb} is obtained at the output of encryptor, the expanded matrix M_{E_arb} and MT_{E_arb} respectively from the plain image P_{arb} and the cipher image C_{arb} are obtained by rearrangement as shown in Figure 1.

The expanded matrix M_{E_arb} and MT_{E_arb} are used as referential masks to detect locations at what its values ar changed after confusion. To attack for permutation of location (x_0, y_0), another plain image $P_{(x_0,y_0)}$ (called a sample plain image) is chosen so that its extended matrix $M_{E\_(x_0,y_0)}$ is with the value of all elements correspondingly equal to that of ME_{arb}, except for that of element at location (x_0, y_0). After encryption of $P(x_0, y_0)$, the cipher image $C_{(x_0,y_0)}$ with its extended matrix $MT_{E\_(x_0,y_0)}$ is obtained for analysis. By comparing $MT_{E\_(x_0,y_0)}$ and MT_{E_arb}, the location (x_1, y_1) with the beginning of value tolerances is detected. It is understood that the pixel at location (x_0, y_0), after rp rounds of permutation, is finally exchanged with that at location (x_1, y_1) after permutation. If other sample plain images are chosen for other locations or (x_0, y_0) is run over all matrices, the full set of affected locations is achieved. In representing the overall confusion rule, two matrices with the same size of $2N \times 2N$, ROW and COL, are used as lookup tables, and store row and column destinations of permutation, respectively. Assume that (x_0, y_0) is the current location, and $(x_1, y_1)$is the destination location in the permutation. Element at location (x_0, y_0) of ROW takes the value x_1 as the lookup table for row and that of COL takes to the value $y_1$with that for column. The confusion attack to find the permutation rule for a pair of pixels is illustrated in Figure 3 and the step-by-step procedure is described as follows to recover the confusion information of a current location (x_0, y_0) and the destination (x_1, y_1)

Step 1: Choose arbitrary values for elements of extended matrix $\mathrm{M_{E\_arb}}$, e.g. equal to zeros.
Step 2: Shrink to become $\mathrm{P_{arb}}$ for encryption
Step 3: Encrypt $\mathrm{P_{arb}}$ to obtain $\mathrm{C_{arb}}$ at the output of encryptor
Step 4: Generate the extended matrix $\mathrm{MT_{E_{arb}}}$ using the ciphertext $\mathrm{C_{arb}}$
Step 5: Select a current location for the confusion attack, $\mathrm{x_0}$ and $\mathrm{y_0}$
Step 6: Assign$\mathrm{M_{E_(x_0,y_0)}} = \mathrm{M_{arb}}$, and modify the element's value of $\mathrm{M_{E\_(x_0,y_0)}}$ at location (x_0, y_0) into a new value.
Step 7: Shrink $\mathrm{M_{E\_(x_0,y_0)}}$ to become $\mathrm{P_{(x_0,y_0)}}$ for encryption
Step 8: Encrypt $\mathrm{P_{(x_0,y_0)}}$ and obtain $\mathrm{C_{(x_0,y_0)}}$ at the output of encryptor
Step 9: Generate the extended matrix $\mathrm{MT_{E\_(x_0,y_0)}}$ using the ciphertext $\mathrm{C_{(x_0,y_0)}}$
Step 10: Compare two matrices $\mathrm{MT_{E\_arb}}$ and $\mathrm{MT_{E\_(x_0,y_0)}}$ to find location (x_1, y_1), at which the value tolerance starts
Step 11: Store the value of x_1 into location (x_0, y_0) of matrix ROW, and store the value of y_1 into location (x_0, y_0) of matrix COL
Step 12: Repeat Step 5 to Step 11 to scan all current locations and to find all destinations

In order to illustrate the confusion attack, an example is illustrated in Figure 4, where Standard map is employed and all system parameters are adopted as given in [24], i.e. system parameter $\alpha = 3.99999$, and initial conditions for generating coefficients of Cat map $conf\_key_1 = 0.12345678912340$ and $conf\_key_2 = 0.88795676859464$, and parameters to generate random number arrays for the diffusion process $key\_d_1 = 0.33798657654353$, $key\_d_2 = 0.72345678912345$, $key\_d_3 = 0.29837465123439$, $key\_d_4 = 0.52341254685124$, and the initial $temp_1 = [\alpha \times key\_d_1 \times (1 - key\_d_1) \times 1000]$. The number of permutation rounds is $rp = 5$, the size of plain images for attack is $N = 5$ (all matrices with the size of 10×10). Here, the extended matrix M_{E_arb} of arbitrary plain image P_{arb} is chosen of all zeros for simplicity as seen on the left panel of Figure 4(a). After encryption, the resulted matrix MT_{E_arb} is obtained from the ciphertext C_{arb} as in the right panel of Figure 4(a). It is easy to observe that the sample image $P(x_0, y_0)$ is chosen so that its extended matrix in the left panel of Figure 4(b) is with only the element at location $(x_0, y_0) = (8, 9)$different from that in M_{E_arb}. After encryption for $P(x_0, y_0)$, the extended matrix, $MT_{E\_(x_0,y_0)}$, generated using $C(x_0, y_0)$ as in the right panel of Figure 4(b) is different from MT_{E_arb} in the right panel of Figure 4(a), starting at location $(x_1, y_1) = (6, 3)$ and beyond in shaded. It means that the input pixel at location $(x_0, y_0) = (8, 9)$ exchanges with that at location $(x_1, y_1) = (6, 3)$ in the permutation, regardless of number of permutation rounds, rp.

The result of confusion attack for the plain image with the size of 5×5 using the above secret key is depicted in Figure 5. There, the overall confusion rule is presented in two lookup tables; Figure. 5(a) and 5(b) are for row and column, respectively. The indices of rows and columns of lookup tables represent for the original locations as $(x_0, y_0)$of elements in M_E, and the destination rows x_1 and columns $y_1$are stored in the elements in the lookup tables. For example, the element of M_E, at (1,2), is exchanged with that at location (8,9); 8 and 9 are values at (1,2) lookup tables for confusion of row and column, respectively. By applying this procedure, the confusion attack is successful regardless of the number of permutation rounds, type of chaotic systems, and without knowledge of secret key as well. The successful attack on confusion process will support the diffusion attack.

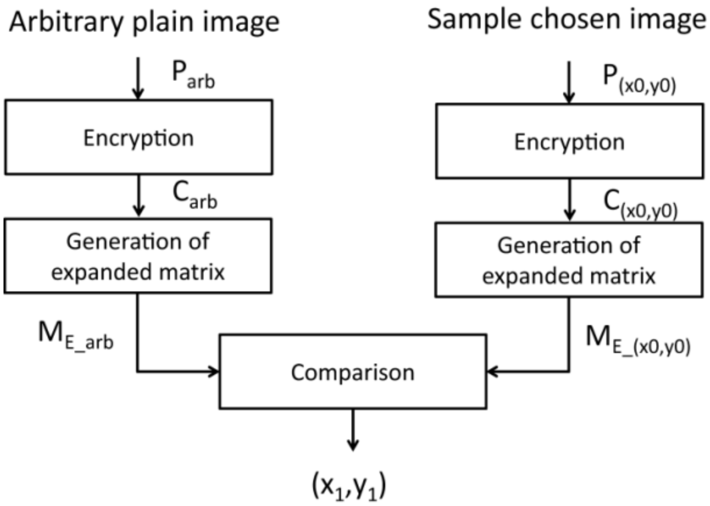
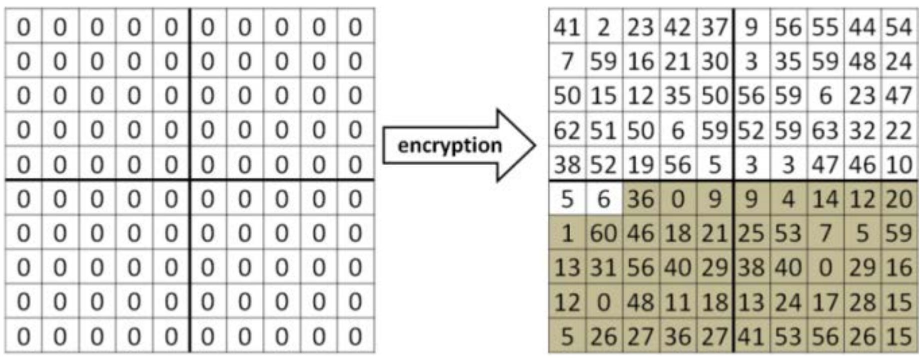
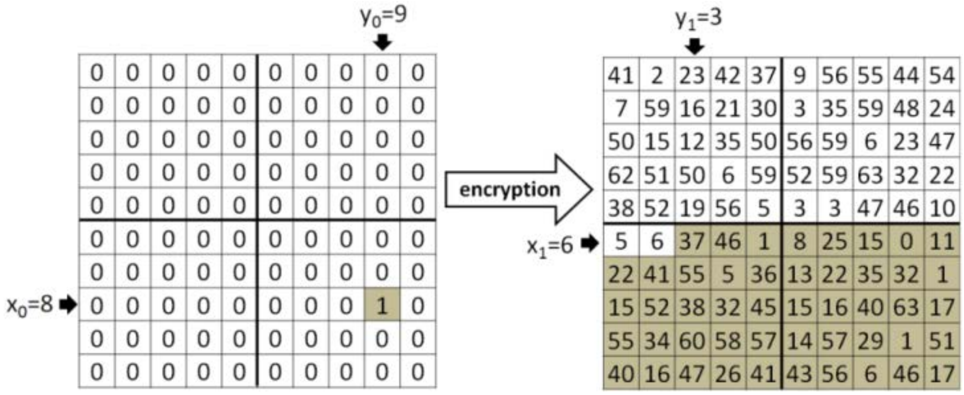


Figure 3. The procedure to recover the confusion rule in the chosen-plaintext attack for location (x_0, y_0).



(a) The expanded matrix of arbitrary plain image, M_{E_abr}, (the left), and its encrypted matrix, MT_{E_arb}, (the right)



(b) The expanded matrix of sample chosen plain image, $M_{E\_(x_0,y_0)}$, (the left), and its encrypted matrix, $MT_{E\_(x_0,y_0)}$, (the right)

Figure 4. Example of confusion attack

1	8	1	2	5	5	9	10	1	8
6	10	1	2	7	9	2	2	3	3
3	4	5	2	7	10	9	10	3	2
4	7	8	9	4	6	4	3	9	6
7	6	5	6	1	5	3	5	5	1
5	5	2	8	8	4	4	8	10	6
3	8	9	10	9	2	7	4	5	10
8	7	1	4	8	1	8	6	6	6
7	4	9	4	1	2	3	9	7	9
6	10	7	1	10	10	3	2	3	7

(a) the matrix ROW

1	9	3	8	9	4	1	7	9	6
8	3	8	4	2	10	7	3	10	6
1	4	7	1	7	2	9	4	9	2
1	10	8	2	5	4	7	8	4	5
9	10	1	9	7	6	7	8	5	6
2	10	10	2	4	2	6	3	10	1
5	10	5	8	7	6	1	10	3	1
1	6	10	9	5	2	7	6	3	7
5	3	3	8	5	5	3	6	3	8
2	9	4	4	5	6	2	9	4	8

(b) the matrix COL

Figure 5. Overall permutation rule. (a) Lookup table for row, (b) Lookup table for column

Attack on Diffusion

After the confusion process, the sequence of words for diffusion is constructed by scanning row by row of elements in the matrix $MP_E$from top to bottom; the 1-D array $A_E$is obtained for the diffusion. By observing the diffusion in Equation (4), it is clear that a current cipher word is dependent directly on its value, $ac(i)$, and values of appropriate elements from the random sequences $rand_2$ and $rand_3$. An element chosen from $rand_3$ for diffusion is only dependent on the location of current cipher word, i, while an element in $rand_2$ chosen for diffusion is only dependent on the value of cipher word standing immediately front, $cipher\_d(i-1)$, via $rand_1$. This is the avalanche effect in the diffusion process. The successful attack on the confusion process in the previous section helps to locate the beginning of affect by the diffusion in the cipher matrix MT_E, and the value at such the location is used for analysis. In the diffusion attack, encryption is carried out many times as change-and-observing process.

In the diffusion attack, the recovery of elements of random sequence named rcv_rd_2 (equivalent to $rand_2$) must be determined for all possible values of cipher words $cipher\_d(i - 1)$. Because cipher words and random sequences are represented by 6 bits, the value range of words is from 0 to 63. In other words, a resulted sequence rcv_rd_2 will have 64 elements, in which the value of $rcv\_rd_2(i)$ will be used for computation of a cipher word with its value of $i-1$. An initial value named $rcv\_rd_{2\_initial}$ (equivalent to $cipher\_d(0)$) should be found for computation of the first cipher word. In addition, a chosen element from $rand_3$ for diffusion is dependent on the location of current cipher word, so the attack for $rand_3$ must be carried out at every location of cipher words using every possible value of plain words. That is, the location range of i in $rand_3$ is from 1 to $4N^2$ and the value range of 6-bit plain words is from 0 to 63. Thus, a matrix named rcv_rd_3 (equivalent to $rand_3$) with the size of $4N^2 \times 64$ must be obtained as the result of attack for $rand_3$.

Let us take a close look on Equation (4), there is a XOR operation ($\oplus$) between $ac(i)$ and $rand_2(temp_2)$, the value of bits at different positions in $rand_2(temp_2)$, can be easily detected by observing resulted values of $cipher\_d(i)$ in the cases of $ac(i) = 0$ and $ac(i) \neq 0$. Bit values at different positions of $rand_2(temp_2)$ can be induced by means of bit tests for every bit position. Several values of $ac(i)$ are interested for detecting bit values in $rand_2(temp_2)$; those are $ac(i) = 1, 2, 4, 8,$ and 16 possibly corresponding to detection the bit value at positions b_0, b_1, b_2, b_3, and b_4 of $rand_2(temp_2)$. It is noted that $cipher_d(i)$takes the value of $(rand_2(temp_2) + rand_3(i))\ mod$ 64 when $ac(i) = 0$. To detect the bit value $b_0$of $rand_2(temp_2)$, $ac(i) = 1$ is applied to the encryptor. If the value of $cipher\_d(i)$ increases by 1 in compared with that when $ac(i) = 0$, the bit b_0 of $rand_2(temp_2)$ is of zero. However, if the value of $cipher\_d(i)$ decreases by 1 in comparison with that when $ac(i) = 0$, the bit b_0 of $rand_2(temp_2)$ is of one. Figure 6 illustrates the example to detect the value of bit b_0 of $rand_2(temp_2)$. Similarly, bits at different positions are tested to predict the value of other bits in $rand_2(temp_2)$ with different values of $ac(i)$ as given in Table 1.

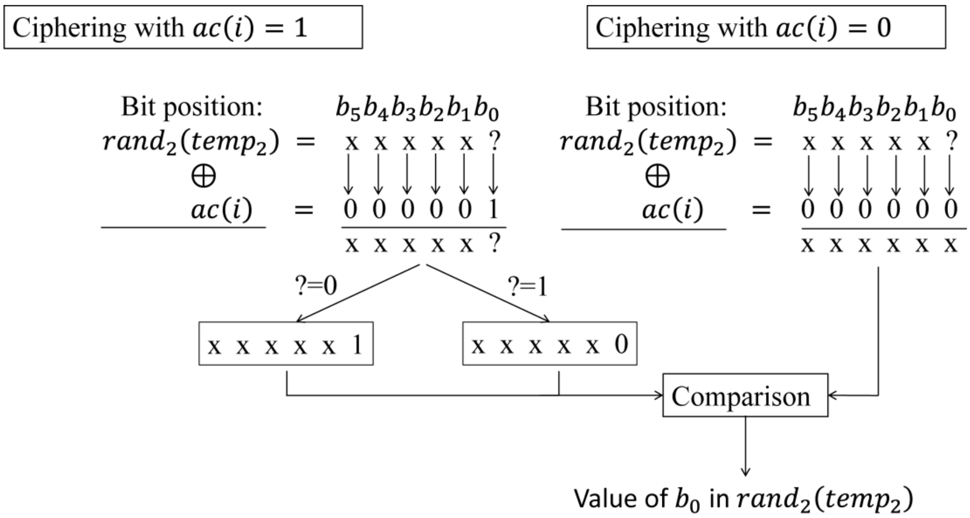


Figure 6. Example for value detection of bit b_0

Let us consider the value of b_5 of $rand_2(temp_2)$ as an exception due that its value causes large change in the output of test. The operation of mod to 64 in the diffusion equation in Equation (4) leads to two solutions in detecting the value of bit b_5 of $rand_2(temp_2)$. In order to illustrate the value detection for b_5 of $rand_2(temp_2)$ as an example shown in Figure 7, there $cipher\_d(i) = rand_2(temp_2) + rand_3(i)$ is equal to either 52 or 116 when $ac(i) = 0$. Either $b_5 = 0$ or $b_5 = 1$ leads to the result $cipher\_d(i) = 20$ when $ac(i) = 32$. This is always true for these values of $rand_2(temp_2)$ and $rand_3(i)$ with ($rand_2(temp_2), rand_3(i) \in [0, 63]$). Thus, it is concluded that there are two possible values of rcv_rd_2 by what the diffusion results right values of $cipher\_d(i)$ and correspondingly two possible values of $rcv\_rd_3(i)$ must be taken into account in the diffusion attack. In other words, two pairs of possible values of ($rcv\ rd_{2a}$,rcv_rd_{3a}) and

(rcv_rd_{2b}, rcv_rd_{3b}) by what the same value of $cipher\_d(i)$ is resulted. Therefore, in the example of diffusion attack dealing with 5×5 image, two sets of diffusion keys are obtained; two random sequences (named rcv_rd_{2a} and rcv_rd_{2b}) are achieved, each of sequences has 65 elements including initial ones for diffusing of the first element ($i = 1$), and two sequences (named rcv_rd_{3a} and rcv_rd_{3b}) are represented in the form of $4N^2 \times 64$ matrices. Note that the value of elements in rcv_rd_{3a} and rcv_rd_{3b} is derived from the constraint with respectively rcv_rd_{2a} and rcv_rd_{2b} for a certain value of $cipher\_d(i)$ and $ac(i)$. In other words, rcv_rd_{3a} and rcv_rd_{3b} are indirectly dependent on $cipher\_d(i-1)$. In the replica encryption using the recovered encryption keys rcv_rd_{3a} or rcv_rd_{3b}, the element $rcv\_rd_{3a}(i,j)$ or $rcv\_rd_{3b}(i,j)$, $i \in [1, 4N^2]$ and $j \in [1, 64]$, is used for computing for the cipher word $cipher\_d(i)$ with the value of $cipher\_d(i-1) = j$. As a result, the equation representing for replica diffusion using recovered keys as in Equation 6, where rcv_rd_2 and rcv_rd_3 is a certain pair of recovered keys.

$$\begin{cases} cipher\_d(0) = rcv\_rd_{2\_initial} \\ cipher\_d(i) = \Big(\big[ac(i) - rcv\_rd_2\big(cipher\_d(i-1)\big)\big] + rcv\_rd_3\big(i, cipher\_d(i-1)\big)\Big) \, mod \; 64, \\ \qquad for \; i = 1 \ldots 4N^2 \end{cases} \tag{6}$$

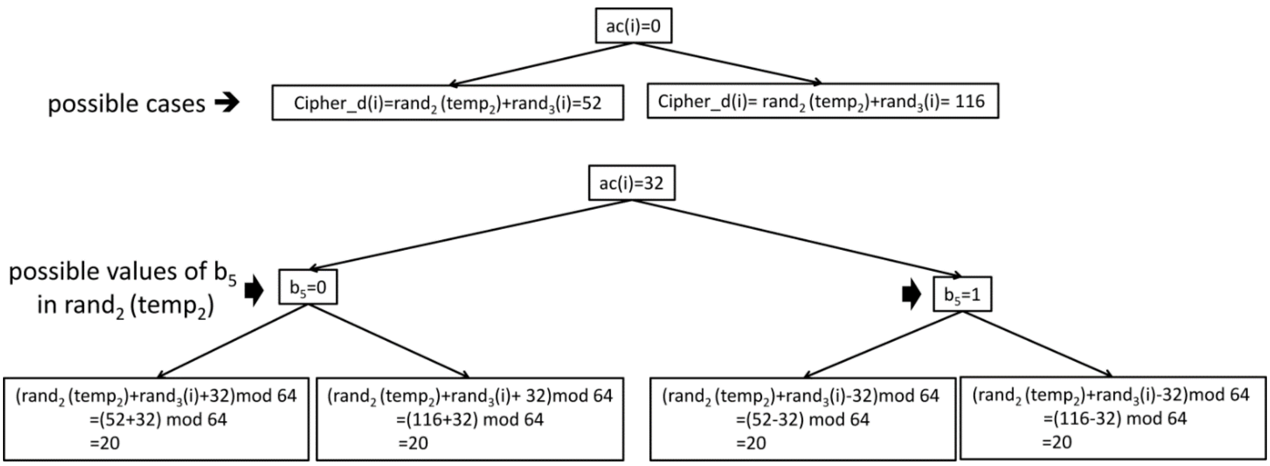


Figure 7. Example of bit value detection of b_5 of $rand_2(temp_2)$.

As an example considers a 5×5 RGB image with values representing for pixels in the R, G and B layers as in Figure 8(a) and its corresponding expanded matrix for encryption is composed by four squares I, II, III and IV as displayed in Figure 8(b). It is recalled that pixels in the quarters I, II and III are from 6 least significant bits of pixels of G, R and B color channels, respectively. Six least significant bits of pixels in the square IV are composed by merging 2 most significant bits from pixels of G, R and B color channels. The 10×10 expanded matrix in Figure 8(b) is ready for encryption. The original random sequence is shown in Figure 8(c). The recovered random sequence rcv_rd_{2a} and rcv_rd_{2b} are depicted in Figure 8(d) and 8(e), where the isolated ones are initial values of rcv_rd_2 for decrypting the first cipher word, $cipher\_d(1)$. The first rows of rcv_rd_{2a} and rcv_rd_{2b} are values of $cipher\_d(i-1)$ and the second rows are values of rcv_rd_2 corresponding to $cipher\_d(i-1)$. The recovered random arrays rcv_rd_{3a} and rcv_rd_{3b} are too large to depict in the figure. It is noted that the original random sequence $rand_2$ is completely different from the recovered ones. The cipher image in Figure 8(f) is obtained under the formal encryption in Equation 4 with original encryption keys as given in Figure 8(c). Figure 8(g) presents the cipher image by replicating the encryption using the recovered lookup tables as in Figure 5 and one pair of random sequences (rcv_rd_{2a} and rcv_rd_{3a}) with the diffusion equation in Equation (6). The cipher image obtained by replica encryption is identical to that using formal encryption. In other words, it is clear that the encryption algorithm cannot resist from the type of chosen-plaintext attack.

Table 1. Detection of Bit Values

Value of $ac(i)$ used for detecting the value of bits in $rand_2(temp_2)$	Amount of change in $cipher\_d(i)$ compared with $cipher\_d(i)$ when $ac(i) = 0$	Bit value b_i in $rand_2(temp_2)$
$ac(i) = 1$	$+1$	$b_0 = 0$
	-1	$b_0 = 1$
$ac(i) = 2$	$+2$	$b_1 = 0$
	-2	$b_1 = 1$
$ac(i) = 4$	$+4$	$b_2 = 0$
$ac(i) = 4$	-4	$b_2 = 1$
$ac(i) = 8$	$+8$	$b_3 = 0$
$ac(i) = 8$	-8	$b_3 = 1$
$ac(i) = 16$	$+16$	$b_4 = 0$
$ac(i) = 16$	-16	$b_4 = 1$

R

118	111	130	147	148
111	123	141	148	130
111	124	126	134	128
91	103	111	129	135
90	99	123	129	127

G

48	35	69	99	110
39	58	89	99	70
40	59	63	73	70
27	37	49	80	82
19	30	65	88	69

B

94	94	130	160	148
98	121	145	147	119
93	114	103	118	132
73	86	93	129	135
67	82	110	132	122

(a) RGB channels of plain image

	54	47	2	19	20	48	35	5	35	46	
	47	59	13	20	2	39	58	25	35	6	
II	47	60	62	6	0	40	59	63	9	6	I
	27	39	47	1	7	27	37	49	16	18	
	26	35	59	1	63	19	30	1	24	5	
	30	30	2	32	20	17	17	38	38	38	
	34	57	17	19	55	17	17	38	38	22	
III	29	50	39	54	4	17	17	17	22	38	IV
	9	22	29	1	7	17	17	17	38	38	
	3	18	46	4	58	17	17	21	38	21	

(b) Expanded matrix for encryption

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
11	43	6	47	10	9	18	13	28	8	49	22	55	4	23	46	50

17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
37	59	3	40	51	45	7	16	63	20	5	58	41	26	36	1

33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
60	24	35	57	44	39	17	0	52	53	38	54	19	56	29	11

49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
62	31	30	14	25	27	34	21	61	12	32	48	2	42	15	33

(c) Original random sequence $rand_2$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
11	16	13	1	10	21	28	0	20	3	14	16	18	25	23	24	12

17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
26	5	13	8	29	4	26	23	2	22	6	27	9	10	9	7

33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
2	17	28	15	24	6	4	1	18	29	7	0	19	19	8	17

49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
5	31	30	15	21	30	31	27	12	14	25	11	20	22	3	11

(d) The first recovered random sequence rcv_rd_{2a}

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
43	48	45	33	42	53	60	32	52	35	46	48	50	57	55	56	44
	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
	58	37	45	40	61	36	58	55	34	54	38	59	41	42	41	39
	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
	34	49	60	47	56	38	36	33	50	61	39	32	51	51	40	49
	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
	37	63	62	47	53	62	63	59	44	46	57	43	52	54	35	43

(e) The second recovered random sequence rcv_rd_{2b}

R

155	111	169	44	114
108	188	18	52	190
19	202	33	225	246
8	162	77	211	170
191	69	184	236	132

G

222	96	26	86	32
204	60	115	159	184
147	69	200	1	70
31	232	231	204	19
141	194	229	207	84

B

112	176	23	212	104
152	207	68	197	29
175	153	97	97	224
74	24	159	253	231
33	55	213	196	82

(f) RGB channels of cipher image under formal encryption

R

155	111	169	44	114
108	188	18	52	190
19	202	33	225	246
8	162	77	211	170
191	69	184	236	132

G

222	96	26	86	32
204	60	115	159	184
147	69	200	1	70
31	232	231	204	19
141	194	229	207	84

B

112	176	23	212	104
152	207	68	197	29
175	153	97	97	224
74	24	159	253	231
33	55	213	196	82

(g) RGB channels of cipher image under replica encryption

Figure 8. Chosen-plaintext attack on 5 × 5 image.

Chosen-ciphertext Attack

In the performance of chosen-ciphertext attack, the diffusion keys and confusion lookup tables are expected to be recovered. The following subsections present detailed procedures and examples for the chosen-ciphertext attack.

Attack on Inverse Confusion

In general, the strategy to attack the inverse confusion rule is as in Figure 9, and the technique to detect the inverse confusion rule is a bit different from that in the chosen-plaintext attack. It is obvious from Equation (4) that the diffusion process performs in the fashion of domino. Thus, the inverse confusion attack using the type of chosen-ciphertext must be started from the last element back to the first one of the extended matrix. However, at the decryption side, the inverse confusion process is carried out after inverse diffusion as in Figure 1. Therefore, inverse confusion will separate any pair of neighbors decrypted words in the extended matrix. Figure 9 illustrates the procedure to attack the confusion. Almost similar to the procedure of confusion attack using the type of chosen-plaintext given in the previous subsection, here, an arbitrary cipher image C_{arb} is chosen and its corresponding extended matrix for decryption, M_{D_arb}, and the recovered plain image P_{arb} are obtained. The extended matrix MP_{D_arb} produced by the recovered plain image P_{arb} (see Figure 1, 2 and Equation (3) for more details) is used throughout the confusion attack for detecting changes in element values of sample extended matrix $MP_{D\_(x0,y0)}$ by means of comparison. As a result, destination location (x_1, y_1) after inverse diffusion process is recognized by detecting the tolerance between values of elements in $MP_{D\_(x0,y0)}$ and that in MP_{D_arb}. As mentioned above on the fashion of diffusion, the inverse diffusion followed by inverse confusion leads to the distribution of elements. Therefore, the inverse confusion attack is started with the location $(x_0, y_0) = (N, N)$ by choosing the cipher image $C_{(x_0,y_0)}$ so that all elements of the sample extended matrix $M_{D\_(x0,y0)}$ are identical to those in M_{D_arb}, except for the element at location (x_0, y_0). After decryption, there is only one element at location (x_1, y_1) in $M_{D\_(x_0,y_0)}$ at which the value of element is different from that at the same location in MP_{D_arb}. It means that the element at location (x_0, y_0) is exchanged with that at location (x_1, y_1). Similarly, the attacking process is continued for the location $(x_0, y_0) = (N, N - 1)$ by choosing the cipher image $C_{(x_0,y_0)}$in the same way as mentioned above. Here, the inverse diffusion makes values of two last elements in $A_{D\_(x_0,y_0)}$ at $(N, N - 1)$ and (N, N) changed in comparison with those in A_{D_arb}. After inverse confusion, these two elements are distributed in $MP_{D\_(x_0,y_0)}$. As a technique to detect inverse confusion rule for (x_0, y_0), two elements in $MP_{D\_(x_0,y_0)}$ have values different from those in MP_{D_arb} detected. One of elements with the value tolerance is at the location for $(x_0, y_0) = (N, N)$ as previously recorded, the other one is for $(x_0, y_0) = (N, N - 1)$. In other words, the destination location $(x_1, y_1)$for $(x_0, y_0) = (N, N - 1)$ is found. The process is continued back to $(x_0, y_0) = (1,1)$ to accomplish the inverse confusion attack. The step-by-step procedure to recover confusion rule is as follows

Step 1: Choose arbitrary values for elements of extended matrix M_{D_arb}, e.g. equal to zeros.
Step 2: Shrink to become C_{arb} for decryption
Step 3: Decrypt C_{arb} to obtain P_{arb} at the output of decryptor
Step 4: Generate the extended matrix MP_{D_arb} using the recovered plaintext P_{arb}
Step 5: Select a current location for the inverse confusion attack, x_0 and y_0
Step 6: Assign $M_{D\_(x_0,y_0)} = M_{D\_arb}$, and modify the element's value of $M_{D\_(x_0,y_0)}$ at location (x_0, y_0) into a new value.
Step 7: Shrink $M_{D\_(x0,y0)}$ to become $C_{(x_0,y_0)}$for decryption
Step 8: Decrypt $C_{(x_0,y_0)}$and obtain $P_{(x_0,y_0)}$ at the output of decryptor
Step 9: Generate the extended matrix $MP_{D\_(x_0,y_0)}$ using the recovered plaintext $P_{(x_0,y_0)}$
Step 10: Compare two matrices MP_{D_arb} and $MP_{D\_(x_0,y_0)}$ to find all possible locations (x_1, y_1), at which the value tolerances occur
Step 11: Keep the only new location of (x_1, y_1), which has not existed in lookup tables
Step 12: Store the value of x_1 into location (x_0, y_0) of matrix ROW, and store the value of y_1 into location (x_0, y_0) of matrix COL
Step 13: Repeat Step 5 to Step 12 to scan all current locations and to find all destinations

Following example demonstrates the inverse confusion attack, in which the value of parameters for decryption is adopted same as in the above examples. Figure 10 illustrates example of 10×10 extended matrices to detect locations whose elements are exchanged with those in $(x_0, y_0) = (10,10)$ and $(x_0, y_0) = (10,9)$. The left panels of Figures 10(a) and 10(b) display the chosen arbitrary matrix M_{D_arb} with all elements of zeros and the sample extended one $M_{D\_(x0,y0)}$ with $(x_0, y_0) = (10, 10)$, respectively. The tolerance in values of elements in its corresponding MP_{D_arb} and $MP_{D_(x_0,y_0)}$after decryption is detected at location $(x_1, y_1) = (6, 9)$ as seen in the right panel of Figure 10(a) and 10(b). In other words, the element at $(x_0, y_0) = (10, 10)$ is exchanged with that at $(x_1, y_1) = (6, 9)$ in the inverse confusion. Continuously, the sample extended one $M_{D\_(x_0,y_0)}$ with $(x_0, y_0) = (10, 9)$ as on the left panel of Figure 10(c). After decryption and by comparing between MP_{D_arb} and $MP_{D\_(x_0,y_0)}$ respectively in the right panels of Figure 10(a) and 10(c), the tolerance in values of elements of MP_{D_arb} and $MP_{D\_(x_0,y_0)}$ is detected at locations $(6, 9)$ and $(10,2)$. The element at location $(x_0, y_0) = (10, 9)$ must be exchanged with that at $(x_1, y_1) = (10, 2)$ because the location $(6, 9)$ has been recorded for $(x_0, y_0) = (10, 10)$ as above. Consequently, the complete lookup tables for the inverse confusion in the decryptor dealing with 10×10 extended matrices are recovered as depicted in Figure 10(d); one is for row and the other is for column. Due to the same value set chosen as in the chosen-plaintext attack, thus the recovered lookup tables in this example are identical to those in Figure 5.

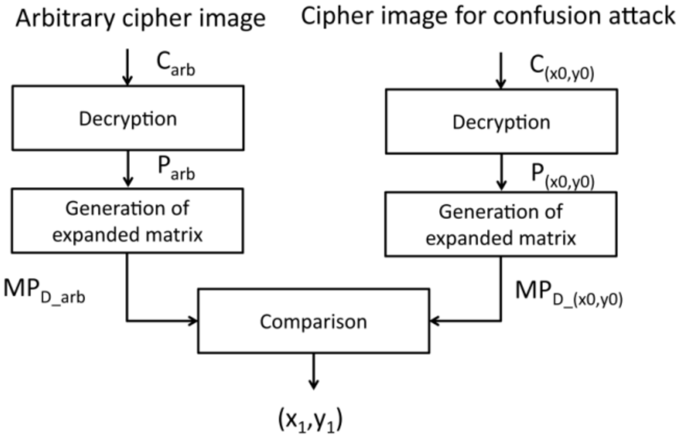
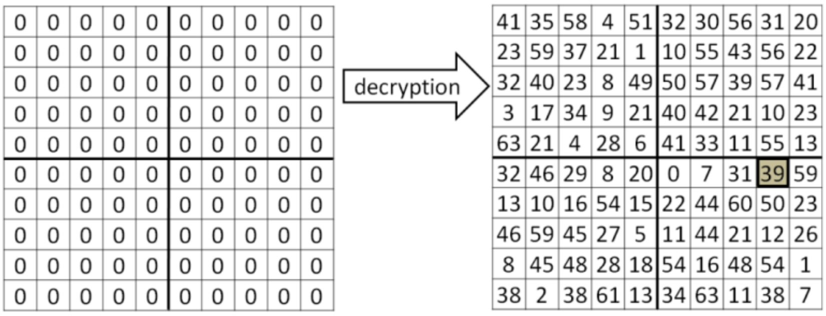
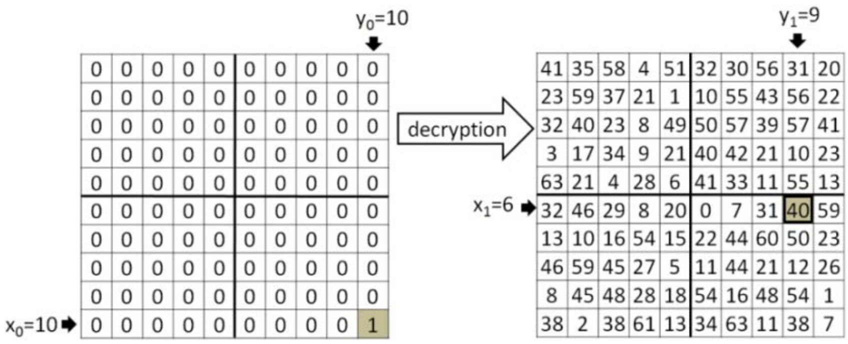


Figure 9. The procedure to recover the confusion rule in the cipher text attack for a pixel at location (x_0, y_0).



(a) Arbitrary values for elements of extended matrix and its recovered one



(b) Sample of chosen values for elements of extended matrix and its recovered one for $(x_0, y_0) = (10, 10)$

$y_0=10$ (column 10 marked); $x_0=10$ (row 10 marked)

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0

decryption →

$y_1=2$ (column 2 marked); $x_1=10$ (row 10 marked)

41	35	58	4	51	32	30	56	31	20
23	59	37	21	1	10	55	43	56	22
32	40	23	8	49	50	57	39	57	41
3	17	34	9	21	40	42	21	10	23
63	21	4	28	6	41	33	11	55	13
32	46	29	8	20	0	7	31	58	59
13	10	16	54	15	22	44	60	50	23
46	59	45	27	5	11	44	21	12	26
8	45	48	28	18	54	16	48	54	1
38	3	38	61	13	34	63	11	38	7

(c) Sample of chosen values for elements of extended matrix and its recovered one for $(x_0, y_0) = (10, 9)$

1	8	1	2	5	5	9	10	1	8
6	10	1	2	7	9	2	2	3	3
3	4	5	2	7	10	9	10	3	2
4	7	8	9	4	6	4	3	9	6
7	6	5	6	1	5	3	5	5	1
5	5	2	8	8	4	4	8	10	6
3	8	9	10	9	2	7	4	5	10
8	7	1	4	8	1	8	6	6	6
7	4	9	4	1	2	3	9	7	9
6	10	7	1	10	10	3	2	3	7

1	9	3	8	9	4	1	7	9	6
8	3	8	4	2	10	7	3	10	6
1	4	7	1	7	2	9	4	9	2
1	10	8	2	5	4	7	8	4	5
9	10	1	9	7	6	7	8	5	6
2	10	10	2	4	2	6	3	10	1
5	10	5	8	7	6	1	10	3	1
1	6	10	9	5	2	7	6	3	7
5	3	3	8	5	5	3	6	3	8
2	9	4	4	5	6	2	9	4	8

(d) Recovered lookup tables of decryptor, ROW (the left) and COL (the right)

Figure 10. Confusion attack in chosen-ciphertext on 10 × 10 extended matrices.

11

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
16	13	1	10	21	60	0	52	35	14	48	18	25	55	24	12

17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
26	5	45	40	61	4	58	23	34	22	6	27	41	42	9	7

33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
2	49	28	15	56	38	36	33	50	29	39	32	51	19	8	17

49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
37	63	62	47	53	30	31	59	44	46	57	11	20	54	3	43

(a) Recovered diffusion key rcv_rd_{2a}

43

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
48	45	33	42	53	28	32	20	3	46	16	50	57	23	56	44

17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
58	37	13	8	29	36	26	55	2	54	38	59	9	10	41	39

33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
34	17	60	47	24	6	4	1	18	61	7	0	19	51	40	49

49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
5	31	30	15	21	62	63	27	12	14	25	43	52	22	35	11

(b) Recovered diffusion key rcv_rd_{2b}

R

118	111	130	147	148
111	123	141	148	130
111	124	126	134	128
91	103	111	129	135
90	99	123	129	127

G

48	35	69	99	110
39	58	89	99	70
40	59	63	73	70
27	37	49	80	82
19	30	65	88	69

B

94	94	130	160	148
98	121	145	147	119
93	114	103	118	132
73	86	93	129	135
67	82	110	132	122

(c) RGB channels of decrypted plain image using recovered diffusion keys

Figure 11. Chosen-ciphertext attack on 5 × 5 image.

Attack on Inverse Diffusion

It is clear that the attack on the inverse diffusion can be preceded only if the inverse confusion rule has been known. Very similar to the diffusion process in the encryption, by observing the equation for decryption in Equation (5) that a cipher word is decrypted with the dependence on its value and the value of the cipher word immediately before. Thus, the approach to attack the inverse diffusion using chosen-ciphertext is almost similar to that in the chosen-plaintext attack as shown in the previous section; that is, sample ciphertexts are chosen for the decryption and corresponding outputs are collected to detect the inverse diffusion keys. The objective of this attack is to find possible inverse diffusion keys rcv_rd_2 and rcv_rd_3 equivalent to the original random sequences $rand_3$ and $rand_2$. In fact, the value of $rand_3(i)$ and $rand_2(temp_2)$ in Equation (5) cannot be directly derived from the availability of $ac(i)$ and $cipher\_d(i)$, thus the method of trial-and-error is utilized to find possible values of rcv_rd_2 and rcv_rd_3. It is clear that an element from $rand_2$ used for decrypting $cipher\_d(i)$ is dependent on $cipher\_d(i-1)$ via $rand_1$. In the inverse diffusion attack, this dependence is written as $rcv\_rd_2(cipher\_d(i-1))$. Thus, to find $rcv\_rd_2(cipher\_d(i-1))$ and $rcv\_rd_3(i)$ equivalent to $rand_2(temp_2)$ and $rand_3(i)$ in the inverse decryption of $cipher\_d(i)$ with a certain value of $cipher\_d(i-1)$, 64 sample extended matrices are chosen with different values of $cipher\_d(i)$ from 0 to 63 are decrypted to produce corresponding decrypted matrices, MP_D. Different values of $cipher\_d(i)$ and corresponding values of $ac(i)$ are used for deriving $rcv\_rd_2(cipher\_d(i-1))$ and $rcv\_rd_3(i)$ by means of computation. That is carried out based on these sequences, $cipher\_d(i)$ and $ac(i)$. By taking a close look on Equation (5), the second case of computation for $a(i)$ is always applied when $cipher\_d(i) = 63$; $cipher(i) \geq rand_3(i)$. This is used as a constraint in computation for possible values of $rcv\_rd_2(cipher\_d(i-1))$ and $rcv\_rd_3(i)$. In other words, for a certain value of $cipher\_d(i-1)$, $cipher\_d(i) = 63$ is chosen to search for possible values of $rcv\_rd_2(cipher\_d(i-1))$ and $rcv\_rd_3(i)$; $rcv\_rd_2(cipher\_d(i-1))$ scans from 0 to 63, and appropriate values of $rcv\_rd_3(i)$ are obtained under the given constraint. In addition, any appropriate pair of values of $rcv\_rd_2(cipher\_d(i-1))$ and $rcv\_rd_3(i)$ must fulfill Equation (5). So, each pair of possible values of $rcv\_rd_2(cipher\_d(i-1))$ and $rcv\_rd_3(i)$ are tried out to compute sequences of values of $cipher\_d(i)$ and $ac(i)$. Right values of $rcv\_rd_2(cipher\_d(i-1))$ and $rcv\_rd_3(i)$, equivalent to $rand_2(temp_2)$ and $rand_3(i)$, produce the sequences of values of $cipher\_d(i)$ and $ac(i)$ matching with those extracted from the above decryption.

Obviously, the XOR operation in Equation (5) leads to two pairs of correct values of $rcv\_rd_2(cipher\_d(i-1))$ and $rcv\_rd_3(i)$ corresponding to a certain value of $cipher\_d(i-1)$. If the value of $cipher\_d(i-1)$ is scanned for the range of from 0 to 63, two sets of correct sequences rcv_rd_2 and rcv_rd_3 are resulted and used as the decryption keys (rcv_rd_{2a}, rcv_rd_{3a}) and (rcv_rd_{2b}, rcv_rd_{3b}). Thus, each of rcv_rd_3 is organized in the form of $4N^2 \times 64$. It is noted that $temp_2$ is not cared in the inverse diffusion attack, instead the value of $cipher\_d(i-1)$ and location of cipher words (the index of i) are important information in the attack. The pseudo code for the diffusion attack is as follows

Input: arbitrary values for elements of extended matrix MD_{arb}
Output: equivalent arrays of random values rcv_rd_2 and rcv_rd_3
FOR $i = 1$ to $4N^2$
    FOR $m = 0$ to 63
        Set $cipher\_d(i-1) = m$
        FOR $n = 0$ to 63

Set $cipher\_d(i) = n$ for M_D
Shrink M_D to become the ciphertext C
Decrypt C to obtain the recovered plaintext P
Generate MP_D using the recovered plaintext P
Extract $ac(i)$
END
Obtain sequences $cipher\_d(i)$ and $ac(i)$ (*)
At $cipher\_d(i) = 63$ (denoted $cipher63$), find value of $ac(i)$ (denoted $ac63$)
FOR $s = 0$ to 63
Assume $rcv\_rd_2(cipher\_d(i-1)) = s$
Find $rcv\_rd_3(i) = cipher63 - [ac63 \oplus rcv\_rd_2(cipher\_d(i-1))$
FOR r=0 to 63
Compute $ac(i)$ using $cipher\_d(i) = r$, $rcv\_rd_2(cipher\_d(i-1))$ and $rcv\_rd_3(i)$ (**)
END
Compare sequences $cipher\_d(i)$ and $ac(i)$ in (*) and those in (**)
IF (TRUE)
Record $rcv\_rd_2(cipher\_d(i-1))$ and $rcv\_rd_3(i)$
END
END
END
END

As a result, two sets of right sequences, (rcv_rd_{2a}, rcv_rd_{3a}) and (rcv_rd_{2b}, rcv_rd_{3b}), are obtained. Each of rcv_rd_2 consists of 65 elements included an initial one for the decryption of the first cipher word, $cipher(1)$. Each of rcv_rd_3 is represented in the form of $4N^2 \times 64$ matrix, in which $rcv\_rd_3(i,j)$ is used for decrypting $cipher\_d(i)$ with $cipher\_d(i-1) = j$. For the replica decryption, these pairs of recovered keys can be used as decryption keys to obtain decrypted plain image, where the equation for inverse diffusion is

$$\begin{cases} cipher\_d(0) = rcv\_rd_{2\_initial} \\ ac(i) = \begin{cases} [64 + cipher\_d(i) - rcv\_rd_3(i, cipher\_d(i-1))] \oplus rcv\_rd_2(cipher\_d(i-1)), \\ \quad \ldots . for\ cipher\_d(i) < rcv\_rd_3(i, cipher\_d(i-1)) \\ [cipher\_d(i) - rcv\_rd_3(i, cipher\_d(i-1))] \oplus rcv\_rd_2(cipher\_d(i-1)), \\ \quad \ldots . for\ cipher\_d(i) \geq rcv\_rd_3(i, cipher\_d(i-1)) \end{cases} \end{cases} \quad (7)$$

Figure 11 displays the result of chosen-ciphertext attack, rcv_rd_{2a} and rcv_rd_{2b} in Figure 11(a) and 11(b), respectively. Note that, the isolate elements in Figure 11(a) and 11(b) are initial values of rcv_rd_{2a} and rcv_rd_{2b}. The original sequence $rand_2$ is as in Figure 8(c). Due to the space limit, $rcv_rd_{3a}$and rcv_rd_{3b} are not shown here. The 5×5 cipher image in Figure 8(f) is decrypted using the recovered rcv_rd_{2a}in Figure 11(a) and rcv_rd_{3a}, and the result is shown in Figure 11(c). It is observed that the decrypted plain image in Figure 11(c) is identical to the original plain image in Figure 8(a). It is obvious that the recovered inverse diffusion keys $rcv_rd_{2a}$and rcv_rd_{2b}in this example for the chosen-ciphertext attack as shown in Figure 11(a) and 11(b) are not identical to those in the example for the chosen-plaintext attack as given in in Figure 8(d) and 8(e). In general, most of recovered diffusion keys are different from original ones that are why they are called "equivalent keys". After thorough tests, the recovered lookup tables and the pairs of diffusion keys in encryptor and decryptor can be used equivalently to the original keys. In addition, the attack is efficient for images regardless to the image size and the number of permutation rounds.

Time Measurement for Attacks

For Confusion Attack

In this subsection, the time measurement of confusion attack is considered for both the chosen-plaintext and chosen-ciphertext attacks. It is measured by the number of encryption/decryption times and the amount of time in computation for an image with the size of $\mathrm{N} \times \mathrm{N}$. Note that the size of $2\mathrm{N} \times 2\mathrm{N}$ for matrices is taken into account in the computation. In both the chosen-plaintext and chosen-ciphertext attacks, confusion attack for a pair of elements in a matrix is required one encryption/decryption time, thus $2N \times 2N$ times of encryption/decryption are carried out for recovering lookup tables. It is assumed that an amount of time for encryption and decryption are T_{en} and T_{de}, respectively. In each time of encrytion/decryption, an amount of time for preparation of chosen-plaintext/-ciphertext images T_p and that for detecting changes in values of elements T_d are taken into account. This means that the amounts of time for the confusion attack for a pair of elements in matrices are $(T_p + T_{en} + T_d)$ and $(T_p + T_{de} + T_d)$ for the chosen-plaintext and chosen-ciphertext, respectively. However, values of pairs of elements are exchanged each other, thus as an optimum only one half of elements are to be considered. That is only true in the case that every element in the first half of extended matrix is exchanged with that in the second half. It does not occur in practical encryption. For a matrix with the size of$2N \times 2N$, the total amounts of time for the confusion attacks are

$$T_{confusion\_CP} = 4 \times N^2 \times (T_p + T_{en} + T_d) \tag{8}$$

for the chosen-plaintext and

$$T_{confusion\_CC} = 4 \times N^2 \times (T_p + T_{de} + T_d) \tag{9}$$

for the chosen-ciphertext.

Table 2. Attacking Time

Type of attack	Time for confusion attack	Time for diffusion attack
Chosen-plaintext attack	$4 \times N^2 \times (T_p + T_{en} + T_d)$	$256 \times N^2 \times [6 \times (T_{en} + T_p) + T_{d\_CP}]$
Chosen-ciphertext attack	$4 \times N^2 \times (T_p + T_{de} + T_d)$	$256 \times N^2 \times [6 \times (T_{de} + T_p) + T_{d\_CC}]$

For Diffusion Attack

In the diffusion attack with chosen-plaintext and chosen-ciphertext, the more encryption/decryption time and more computation is required while in attacking. Firstly, let us consider complexity for diffusion break in the type of chosen plaintext attack. As mentioned in the description of diffusion attack that $rand_2$ and $rand_3$ are dependent on the value of cipher words immediately before ($cipher\_d(i)$) and the location of cipher words, i, respectively. For a certain value of $cipher\_d(i)$, six encryption times is carried out to have sequences of $ac(i)$ versus $cipher\_d(i)$ for detection of values of bits in $rand_2$ and $rand_3$;

$ac(i) = [0, 1, 2, 4, 8, 16]$. In addition, all possible values of plain words $ac(i-1)$ for producing $cipher\_d(i-1)$ are from 0 to 63, in other words, consideration for detecting values of a pair of elements in $rand_2$ and $rand_3$ is required 64 times of encryption. Assumed that amount of time for detection of values of bits for a pair of elements in $rand_2$ and $rand_3$ is $T_{\mathrm{d\_CP}}$ and that for preparation for a plaintext image is T_p. Thus, amount of time for detecting a pair of elements equivalent to those in $rand_2$ and $rand_3$ are $64 \times [6 \times (T_{en} + T_p) + T_{d\_CP}]$. As a result, a matrix with the size of 2N × 2N is required totally

$$\mathrm{T}_{diffusion\_CP} = 256 \times \mathrm{N}^2 \times [6 \times (\mathrm{T}_{en} + \mathrm{T_p}) + \mathrm{T}_{d\_CP}]. \quad (10)$$

Secondly, the amount of time required for inverse diffusion attack in the chosen-ciphertext is considered. It is very similar to consideration for that in the chosen-plaintext attack, except that the number of 64 decryption times are carried out for a pair of elements what are equivalent to those in $rand_2$ and $rand_3$ rather than 6. It is noted that amount of time for analysing to find appropriate values of elements equivalent to $rand_2$ and $rand_3$ is T_{d_CC}. That is mostly spent for comparison between two matrices with the size of 2×64, obtained by $cipher\_d(i)$ and $ac(i)$. Thus, the total amount of time for attacking for a matrix with the size of $2N \times 2N$ is

$$\mathrm{T_{diffusion\_CC}} = 256 \times \mathrm{N}^2 \times [64 \times (\mathrm{T}_{de} + \mathrm{T_p}) + \mathrm{T}_{d\_CC}]. \quad (11)$$

Let us roughly compare the time consummation in the chosen-plaintext and chosen-ciphertext attacks. Total amount of time for the confusion attack in the chosen-plaintext (Equation (8)) is different from that in the chosen-ciphertext (in Equation (9)) with an amount of $\Delta T_{confusion} = 4 \times N^2 \times \delta_C$; where $\delta_C = |T_{de} - T_{en}|$. This tolerance is small when δ_C is negligible, or the encryption and decryption take almost the same amount of time. Furthermore, it is clear that the difference of time consummation for the diffusion attack between in the chosen-plaintext (Equation (8)) and in the chosen-ciphertext (Equation (9)) is pretty large, i.e. $256 \times N^2 \times [58 \times (T_{de} + T_p)]$, with the assumption of $T_{d\_CP} \approx T_{d\_CC}$ and $T_{en} \approx T_{de}$. This is considerably large in compared with amount of time for diffusion attack in the chosen-plaintext. As a consequence, a larger amount of time is required for the chosen-ciphertext attack in comparison with that for the chosen-plaintext attack. The summary of time consummation is shown in Table 2.

Discussion and Conclusion

According to cryptanalysis and examples illustrated in Figure 8 and 11, the recovered encryption/decryption keys are different from original ones, but those are equivalent to originals. The attacks do not require any knowledge about value of parameters for chaotic systems. In addition, as given in Table 2, amount of time for breaking the cryptosystem using the chosen-ciphertext is considerably larger than that using the chosen-plaintext, and that is strongly dependent on the size of image, i.e. N^2. Moreover, in the above examples for chosen-plaintext and chosen-ciphertext attacks, the extended matrices of plain image and cipher one chosen for comparison with the encryption and decryption results are of all pixels of zeros. In fact, any image can be employed for this purpose, but it is required that the value of element at a location being attacked in sample chosen extended matrices must be different from that in these ones.

The cryptosystem proposed by W. Zhang et al. with one encryption round of SPN does not provide security even multiple rounds of permutation followed by one diffusion process. By taking a close look on attack procedures, it does not depend on how many permutation rounds are before diffusion. In addition, lookup tables may not be recovered in case there is more than one encryption round, and accordingly attacking for diffusion must be failed. It means that, the cryptosystem can provide extremely high security if multiple encryption rounds are applied. In such the case, encryption time may reduce by reducing a number of permutation rounds to one. In this context, it is clear that the statistical analysis for the encryption does not mean that the security is assured. That only suggests a minimum number of rounds to ensure that the cipher image cannot be detected by human perspective. In summary, again one encryption round for of SPN is proved to be insecure. It is to suggest that cryptosystems based on the architecture of SPN must have more than one encryption round in order to get high security. In the case of multiple encryption round, these attack methods cannot be successful. This will be dealt in the future work of research.

Acknowledgment

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.02-2012.27

References

[1] A. Medio, and M. Lines, *Nonlinear Dynamics: A Primer*, Cambridge University Press, Cambridge, United Kingdom, 2001.

[2] J. Fridrich, “Symmetric ciphers based on two-dimensional chaotic maps,” *International Journal of Bifurcation and Chaos*, Vol. 8, pp. 1259–1284, 1998.

[3] L. Kocarev, and S. Lian, eds., *Chaos-based Cryptography: Theory, algorithms and applications*, Springer-Verlag Berlin Heidelberg, 2011.

[4] Y. Liu, S. Tian, W. Hu, and C. Xing, “Design and statistical analysis of a new chaotic block cipher for wireless sensor networks,” *Communications in Nonlinear Science and Numerical Simulation*,Vol. 17, pp. 3267–3278, 2012.

[5] E. Yavuz, R. Yazici, M.C. Kasapbasi, and E. Yamac, “A chaos-based image encryption algorithm with simple logical functions,” *Computers & Electrical Engineering*, Vol. 54, pp. 471-483, 2016.

[6] S.E. Assad, and M. Farajallah, “A new chaos-based image encryption system,” *Signal Processing: Image Communication*, Vol. 41, pp. 144-157, 2016.

[7] M. Farajallah, S.E. Assad, and O. Deforges, “Fast and Secure Chaos-Based Cryptosystem for Images,” *International Journal of Bifurcation and Chaos*, Vol. 26, No. 2, pp. 1650021-1 to 1650021-21, 2016.

[8] L.Y. Zhang, C. Li, K.W. Wong, S. Shu, and G. Chen, “Cryptanalyzing a chaos-based image encryption algorithm using alternate structure,” *Journal of Systems and Software*, Vol. 85 No. 9, pp. 2077–2085, 2012

[9] G. Alvarez, and S. Li, “Some basic cryptographic requirements for chaos-based cryptosystems,” *International Journal of Bifurcation and Chaos*, Vol. 16, pp. 2129–2151, 2006.

[10] E. Solak, C. Cokal, O.T. Yildiz, and T. Biyikoglu, “Cryptanalysis of Fridrich’s chaotic image encryption,” *International Journal of Bifurcation and Chaos*, Vol. 20, pp. 1405–1413, 2010.

[11] C. Li, L.Y. Zhang, R. Ou, K.W. Wong, and S. Shu, “Breaking a novel colour image encryption algorithm based on chaos,” *Nonlinear Dynamics*, Vol.70, pp. 2383–2388, 2012.

[12] C. Zhu, and C. Liao, and X. Deng, “Breaking and improving an image encryption scheme based on total shuffling scheme,” *Nonlinear Dynamics*, Vol. 71, pp. 25–34, 2013.

[13] H. Feistel, “Cryptography and computer privacy,” *Scientific American*, Vol. 228, pp. 15 – 23, 1973.

[14] D. Stinson, *Cryptography: Theory and Practice*, 3rd Edition, CRC Press, Boca Raton, Florida, United States of America, 2005.

[15] A. Belazi, A.A.A. El-Latif, and S. Belghith, “A novel image encryption scheme based on substitution-permutation network and chaos,” *Signal Processing*, Vol. 128, pp. 155-170, 2016.

[16] H. Heys, and S. Tavares, “Avalanche characteristics of substitution-permutation encryption networks,” *IEEE Transactions*, Vol. 44, pp. 1131–1139, 1995.

[17] X.J. Tong, “Design of an image encryption scheme based on a multiple chaotic map,” *Communications in Nonlinear Science and Numerical Simulation*, Vol. 18, pp. 1725–1733, 2013.

[18] S. Lian, J. Sun, and Z. Wang, “Security analysis of a chaos-based image encryption algorithm,” *Physica A: Statistical Mechanics and its Applications*, Vol. 35, pp. 1645-1661, 2005.

[19] A. Kanso, and M. Ghebleh, “A novel image encryption algorithm based on a 3D chaotic map,” *Communications in Nonlinear Science and Numerical Simulation*, Vol. 17, pp. 2943–2959, 2012.

[20] X. Zhang, L. Shao, Z. Zhao, and Z. Liang, “An image encryption scheme based on constructing large permutation with chaotic sequence,” *Computers & Electrical Engineering*, Vol. 40, pp. 931-941, 2014.

[21] K.W. Wong, B.S.H. Kwok, and W.S. Law, “A fast image encryption scheme based on chaotic standard map,” *Physics Letters A*, Vol. 372, pp. 2645 –2652, 2008.

[22] T.T.K. Hue, C.V. Lam, T.M. Hoang, and S.A. Assad, “Implementation of secure SPN chaos-based cryptosystem on FPGA,” In: *2012 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pp. 000129–000134, 2012. doi:10.1109/ISSPIT.2012.6621274

[23] Y. Zhang, D. Xiao, Y. Shu, and J. Li, “A novel image encryption scheme based on a linear hyperbolic chaotic system of partial differential equations,” *Signal Processing: Image Communication*, Vol. 28, pp. 292–300, 2013.

[24] W. Zhang, K.W. Wong, H. Yu, and Z.L. Zhu, “A symmetric color image encryption algorithm using the intrinsic features of bit distributions,” *Communications in Nonlinear Science and Numerical Simulation*, Vol. 18, pp. 584–600, 2013.

[25] Y. Mao, G. Chen, and S. Lian, “A novel fast image encryption scheme based on 3d chaotic baker maps,” *International Journal of Bifurcation and Chaos*, Vol. 14, pp. 3613–3624, 2004. doi:10.1142/S021812740401151X

[26] D. Arroyo, J. Diaz, and F. Rodriguez, “Cryptanalysis of a one round chaos-based substitution permutation network,” *Signal Processing* , Vol. 93, pp. 1358 –1364, 2013.

[27] F. Rannou, “Numerical study of discrete plane area-preserving mappings,” *Astronomy & Astrophysics*, Vol. 31, pp. 289–301, 1974.

[28] E.A. Jackson, *Perspectives of Nonlinear Dynamics*, Cambridge University Press, New York, United States of America, 1991.

[29] E.A. Arnold, and A. Avez, *Ergodic Problems of Classical Mechanics*, Benjamin, 1968.