American Journal of Circuits, Systems and Signal Processing
Vol. 1, No. 3, 2015, pp. 105-113
http://www.aiscience.org/journal/ajcssp

Comments on "An Image Encryption Scheme Based on Rotation Matrix Bit-Level Permutation and Block Diffusion"

Yong Zhang[*]

School of Software and Communication Engineering, Jiangxi University of Finance and Economics, Nanchang, P. R. China

Abstract

Recently, an image encryption scheme based on rotation matrix bit-level permutation and block diffusion was proposed [Y Zhang, D Xiao. Commun Nonlinear Sci Numer Simulat. 2014, 19:74-82]. In this paper, this image encryption scheme was studied in detail and its defects of low encryption speed and weak security were pointed out. This scheme with one round was crypt-analyzed successfully with the chosen plaintext method. The simulation results show that their scheme cannot be used in practical communications.

Keywords

Chaotic System, Chosen Plaintext Attack, Cryptanalysis, Image Encryption

Received: June 27, 2015 / Accepted: July 11, 2015 / Published online: July 23, 2015

1. Introduction

In recent years, a number of image encryption schemes based on chaotic systems were proposed [1-7]. In such image encryption systems, the encryption algorithms are sensitive to secret keys and plain images, while the decryption algorithms are sensitive to secret keys and cipher images. The latter makes the cipher images generated by such image encryption systems cannot be transmitted in noise channel directly due to the slight changing in cipher images will lead to complete failure in decryption process. Currently, there are still some scientists engaged in the anti-noise image encryption technology based on chaotic system. For example, A. N. Pisarchik and M. Zanin presented a color image encryption scheme with chaotically coupled chaotic maps. This encryption scheme utilizes chaotic confusion of image pixels using chaotic coupling between chaotic maps, each of which in turn induces chaotic diffusion of pixels' color values. They claimed that their scheme not only makes the known plaintext attack unfeasible, but also is robust against noise and other external disturbances [8]. But soon, D. Arroyo, S. Li, and etc. pointed out that the encryption architecture of this cryptosystem possesses serious security problems related to its implementation and its robustness against noise [9].

Recently, Y. Zhang and D. Xiao proposed an image encryption scheme based on rotation matrix bit-level permutation and block diffusion, and claimed that their scheme not only achieves a satisfactory security performance, but also has the suitability for a parallel mode and the robustness against noise in communication system [10]. Their scheme was named as ZX2014. In this paper, we crypt-analyzed the ZX2014, provided some security vulnerabilities, and attacked the ZX2014 with the chosen plaintext method. This paper is organized as follows: Section 2 reviews the ZX2014; Section 3 analyzes the security problems of ZX2014; Section 4 discusses the chosen plaintext attack method on ZX2014; Section 5 gives some simulation results; Section 6 concludes the paper.

* Corresponding author
E-mail address: zhangyong@jxufe.edu.cn

2. Encryption Algorithm of ZX2014

The encryption algorithm of ZX2014 consists of three stages: (I) scrambling operation based on the entire plain image; (II) bit-level permutation based on each small image block; (III) diffusion operation based on each small image block. In Stages II and III, the operations of permutation and diffusion on small blocks are independent to each other, the aim of this is: (1) preventing the noise in cipher image from spreading to the whole image during the decryption process; (2) implementing the parallel computing. The encryption algorithm of ZX2014 will be discussed in detail as below:

Suppose that the plain image is 8-bit gray scale image, denoted by P with size of $N\times N$. N mod 8=0 is required. The secret key of ZX2014 is denoted by $K=\{key_1, key_2, key_3, key_{41}, key_{42}, key_5, key_6\}$, where, key_1, key_2, key_3 and key_5 are float-point numbers in interval (0,1) used as initial values of Logistic map with the precision of 10^{-16}, key_{41} and key_{42} are two integers range of [1,8], and key_6 is an integer range of [0,255]. Logistic map is used in ZX2014 to generate the pseudo random sequence, and its equation is as follows:

$$x_{i+1}=\mu x_i(1-x_i),\ i=0,1,2,... \qquad (1)$$

where, μ=3.9999.

Stage I. Scrambling operation based on the entire image

Rearrange matrix P row by row to get a vector of length N^2, denoted by A, such that $A((i-1)\times N+j)=P(i,j)$, $i,j=1,2,...,N$. Let the initial value of Eq. (1) be $x_0=key_2$, and iterate Eq. (1) as transient states for t_1 times, then continue to iterate Eq.(1) for N^2 times to get a state variable vector of length N^2, named by $\{x(i), i=1,2,...,N^2\}$, whose index vector is denoted by $\{I(i)=i, i=1,2,...,N^2\}$. Sort vector x in ascending order to obtain another vector named by $\{y(i), i=1,2,...,N^2\}$, whose index vector is denoted by $\{J(\mathrm{i}), i=1,2,...,N^2\}$, such that $x(J(i))=y(I(i))$, $i=1,2,...,N^2$. Then, employ vector J as the subscript to scramble vector A, such that $A(I(i))$ and $A(J(i))$ are exchanged by position, for every $i=1,2,...,N^2$. Note that the vector J is one of the target equivalent keys for attacker.

The scrambled vector A is converted into a matrix with size of $N\times N$, denoted by D, such that $D(i,j)=A((i-1)\times N+j)$, $i,j=1,2,...,N$.

Stage II. Bit-level permutation based on each small image block

Step 1. Divide the image D into a series of non-overlapping small image blocks with size of 8×8 from left to right then from top to bottom sequentially, denoted by $B(i)$, $i=1,2,...,L$, where, $L=N^2/64$. By expanding each 8-bit pixel into the form of bit group, represent each block $B(i)$ into a three-dimensional bit cube with size of 8 × 8 × 8, denoted by $DB(i)$, $i=1,2,...,L$.

Step 2. Let the initial value of Eq. (1) be $x_0=key_3$. After iterating Eq. (1) for t_2 times as the transient states, continue to iterate Eq. (1) for L times to get a vector of length L, named by $\{x(i), i=1,2,...,L\}$. Then convert the vector x into an integer vector, denoted by $\{idx(i), i=1,2,...,L\}$, with the following formula:

$$idx(\mathrm{i})=\mathrm{floor}(x(i)\times 10^{10}) \bmod 6 +1,\ i=1,2,...,L \qquad (2)$$

Note that the vector idx is one of target equivalent keys for attacker.

Step 3. Generate eight pieces of bit matrices with size of 8×8, denoted by $\{PM(k), k=1,2,...,8\}$, with the following algorithms:

(1) Let the initial value of Eq. (1) be $x_0=key_1$, then iterate Eq. (1) for t_3+64 times (include t_3 times of transient states iteration) to get a vector of length 64, denoted by $\{\boldsymbol{x}(i), i=1,2,...,64\}$.

(2) Initialize all elements of each $PM(i)$, i=1,2,...,8 to 0.

```
(3) For $i$=1 to 8 Do
Introduce a vector $\boldsymbol{y}_1=\boldsymbol{x}(8\times(i-1)+1$ to $8\times(i-1)+8)$;
Sort vector $\boldsymbol{y}_1$ in ascending order to get its sorted index sequence, denoted by $\boldsymbol{y}_2$;
For $k$=1 to 8 Do
Set the element of position $(\boldsymbol{y}_2(k),k)$ in $PM(i)$ to 1;
End
End
```

Note that the matrices $\{PM(i), i=1,2,...,8\}$ are part of target equivalent keys for attacker.

Step 4. For each $DB(i)$, conduct the following bit permutation operations:

(1) Introduce two variables k_1 and k_2, such that $k_1=key_{41}$ and $k_2=key_{42}$.

Note that key_{41} and key_{42} are part of target equivalent keys for attacker.

(2) Rotate matrix $DB(i)$ according to the value of $idx(i)$ with one of the following six cases:

Case $idx(i)$=1: $DB(i)$ remains unchanged;

Case $idx(i)$=2: $DB(i)$ is rotated by 180 degrees;

Case $idx(i)$=3: $DB(i)$ is rotated by 90 degrees from left to right;

Case $idx(i)$=4: $DB(i)$ is rotated by 90 degrees from right to left;

Case $idx(i)=5$: $DB(i)$ is rotated by 90 degrees from front to back;

Case $idx(i)=6$: $DB(i)$ is rotated by 90 degrees from back to front.

(3) For $j=1$ to 8 Do

Swap the k_1-th row and the k_2-th row in $PM(j)$;

Multiply the j-th layer of cube $DB(i)$ with matrix $PM(j)$ to get a matrix denoted by $PB(j)$;

$k_1=k_2$, $k_2=\text{sum}(PB(j)) \bmod 8 + 1$

End

(4) Arrange matrices $PB(1)$, $PB(2)$,..., $PB(8)$ from top to bottom to obtain a new bit cube, denoted by $DD(i)$, which is the permutated bit cube of $DB(i)$. Convert $DD(i)$ into a decimal 8×8 matrix, denoted by $H(i)$.

Stage III. Diffusion operation based on each small image block

The size of the small image blocks was suggested to be 8×8 or 16×16 in ZX2014. If its size is 8×8, the matrices of $H(i)$, $i=1,2,...,L$ generated in Stage II can be used in this stage directly; otherwise, all the $H(i)$-s should be combined into a whole image and be re-segmented. Without loss of generality, we assume that the size of image block used in Stage III is 8 × 8.

The diffusion operation for each $H(i)$ is independent to realize parallel processing in ZX2014. Set the initial **value** of Eq. (1) as $x_0=key_5$, then iterate Eq. (1) for t_4+64 times (include t_4 times of transient states iteration) to obtain a vector of length 64, denoted by $\{x(j), j=1,2,...,64\}$. Convert x into an integer vector, denoted by $\{X(j), j=1,2,...,64\}$, with the following formula:

$$X(j)=(x(j)\times 10^{10}) \bmod 256, j=1,2,...,64 \qquad (3)$$

Then diffuse matrix $H(i)$ with the following steps:

Step 1. Expand $H(i)$ row by row to obtain a vector of length 64, denoted by $\{E(j), j=1,2,...,64\}$.

Step 2. Transform E into a new vector, denoted by $\{F(j), j=1,2,...,64\}$, with the following formula:

$$F(j)=F(j-1) \oplus X(j) \oplus (E(j)+X(j) \bmod 256), j=1,2,...,64 \quad (4)$$

Where, ‘$\oplus$’ means bitwise XOR operation, and $F(0)=key_6$.

Note that the matrix of X and key_6 are part of target equivalent keys for attacker.

Convert vector F into two-dimensional matrix G, such that $G(k,j)=F(8\times(k-1)+j)$, $k,j=1,2,...,8$. Then assign G to $H(i)$ to update $H(i)$, i.e. new $H(i)=G$.

After the diffusion of all $H(i)$-s in $\{H(i), i=1,2,...,L\}$, rearrange all $H(i)$-s to obtain a matrix of size $N\times N$, denoted by C. Then C is the cipher image. The above process may be repeated for n times to enhance the security.

The encryption scheme of ZX2014 is as shown in Fig. 1, and more detail about ZX2014 can be referred to [10].

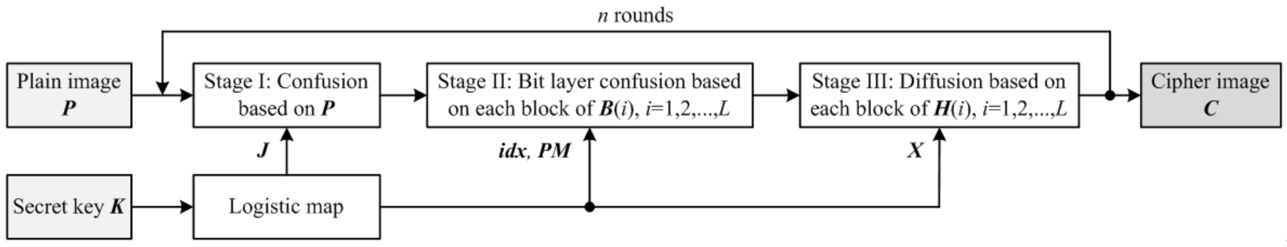
n rounds
Plain image P
Stage I: Confusion based on P
Stage II: Bit layer confusion based on each block of B(i), i=1,2,...,L
Stage III: Diffusion based on each block of H(i), i=1,2,...,L
Cipher image C
J
idx, PM
X
Secret key K
Logistic map

Figure 1. Encryption scheme of ZX2014.

3. Some Comments on ZX2014

Table 1. The PER between the decrypted images and original image.

Salt and pepper noise	Pixel Error Rate (%) No. of rounds=1	No. of rounds=2	No. of rounds=3
0.1%	0.7523	5.9555	31.4301
0.2%	1.4389	10.1395	45.9686
0.5%	3.7643	21.9452	63.5925
1.0%	7.0786	36.1359	74.8337
10.0%	42.8238	79.7211	97.0474

Based on the analysis of ZX2014 in Section 2, some comments are given as below:

(1) ZX2014 can fight against the salt and pepper noise.

We took plain image Lena with size of 256×256 as example. When the cipher images of Lena were obtained under the condition that the round number of encryption process is 1, 2 and 3, and were polluted by the salt and pepper noise with the ratio of 0.1%, 0.2%, 0.5%, 1.0% and 10%. We analyzed the pixel error rate (PER) between the decrypted images and the original image, and listed the results in Table 1. Then we illustrated the decrypted images when the cipher images were polluted by 10% of salt and pepper noise in Fig. 2.

From Table 1 and Fig. 2, it can be seen that ZX2014 can fight against the salt and pepper noise. When the cipher images are polluted by 10% of salt and pepper noise and the number of rounds is 1 and 2, the decryption algorithm of ZX2014 can still recover the outline of the images clearly. However, from another perspective, this demonstrates that the decryption algorithm of ZX2014 is not sensitive to the change in cipher image, i.e., the small changes in the cipher image cannot be spread to the whole decrypted image.(2) The encryption algorithm of ZX2014 is insensitive to the small changes in plain image, when the number of rounds is 1, 2, 3 and 4.

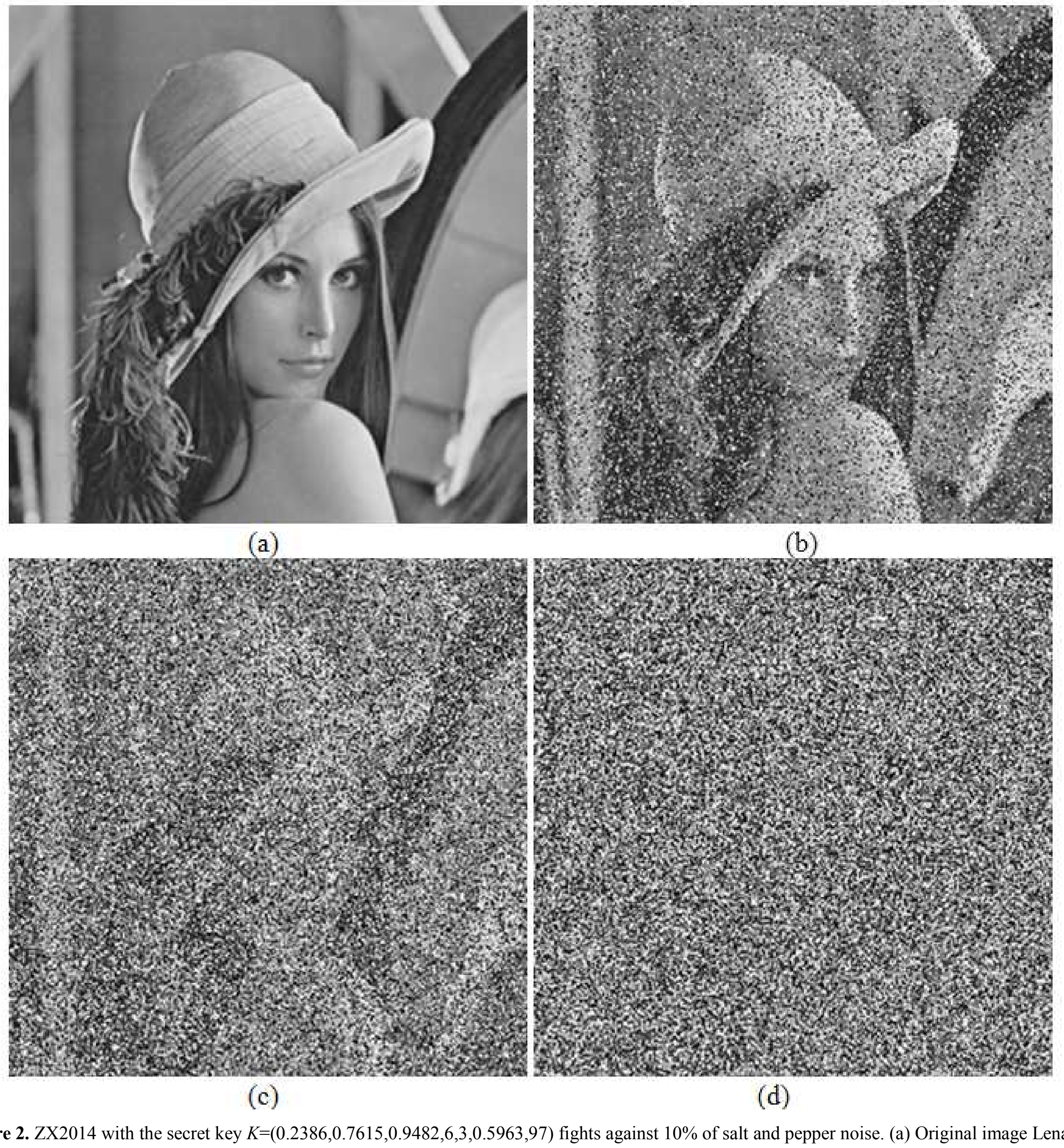

Figure 2. ZX2014 with the secret key K=(0.2386,0.7615,0.9482,6,3,0.5963,97) fights against 10% of salt and pepper noise. (a) Original image Lena; (b) Recovered image when the number of rounds is 1; (c) Recovered image when the number of rounds is 2; (d) Recovered image when the number of rounds is 3.

In general, the NPCR (number of pixels change rate) and UACI (unified average changing intensity) are used to measure the sensitivity of cryptosystem [3]. NPCR and UACI are calculated by the following formulas:

$$\text{NPCR}=\frac{\sum_{i,j}\boldsymbol{D}(i,j)}{M\times N}\times 100\% \tag{5}$$

$$\text{UACI}=\frac{1}{M\times N}\sum_{i,j}\frac{|\boldsymbol{C}_1(i,j)-\boldsymbol{C}_2(i,j)|}{255}\times 100\% \tag{6}$$

Where, C_1 and C_2 are two cipher images with size of $M\times N$. If $C_1(i,j)=C_2(i,j)$, then $D(i,j)=0$; otherwise, $D(i,j)=1$. Their theoretical values of NPCR and UACI are 255/256 ≈ 99.6094% and 257/768 ≈ 33.4635%, respectively [11].

Take the plain image Lena of size 256×256 as example. Under the condition that the number of rounds is 1, 2, 3 or 4, randomly select one pixel from the plain image and change its value by 1. Encrypt the original and changed images to get two cipher images, respectively. And then calculate the NPCR and UACI of these two cipher images. We did the above test for 100 times to calculate the average values of NPCR and UACI, and the results were listed in Table 2.

Table 2. The results of sensitivity test on plain image.

	No. of rounds=1	No. of rounds=2	No. of rounds=3	No. of rounds=4
NPCR (%)	0.0594	1.8273	46.6931	96.0555
UACI (%)	0.0131	0.5710	15.5069	32.2344

It should be noted that the result for rounds=2 or 3 in table 2 is obviously different from the result in [10]. The possible reason for this is we use same secret keys and blocks with same size in every round, while [10] used different secret keys and blocks with different size in every round.

From Table 2, it can be seen that the calculated values of NPCR and UACI deviate from their theoretical values seriously, respectively, when the number of rounds is 1, 2 and 3, while the calculated values of NPCR and UACI are much closer to their theoretical values, respectively, when the number of rounds is 4. This shows that the encryption algorithm of ZX2014 is not sensitive to the changes in plain image. Therefore, there could be security loopholes in ZX2014 for differential attacks. So, the number of round should be 4 or above in ZX2014.

Meantime, we cannot evaluate the plain image sensitivity of ZX2014 simply by the values of NPCR and UACI, because the encryption algorithm of ZX2014 is a block encryption strategy. However, the calculated values of NPCR and UACI deviating far from their theoretical values, at least explains that the diffusion level between image blocks in ZX2014 is limited.

Table 3. The encryption and decryption time of ZX2014 (s).

	No. of rounds=1	No. of rounds=2	No. of rounds=3	No. of rounds=4
Encryption time(s)	3.6589	7.2638	10.9361	14.5965
Decryption time(s)	3.6660	7.3067	10.9778	14.7705

(3) The encryption speed of ZX2014 is slow.

The computer used was configured with Intel Duo Core I5 M460@2.53GHz, 2GB DDR3 RAM, Windows 7 and MATLAB 8.3. Without loss of generality, the secret key in ZX2014 is taken as K = (0.2386, 0.7615, 0.9482, 6, 3, 0.5963, 97). The encryption and decryption time with the number of rounds being 1, 2, 3 and 4 is listed in Table 3.

In the same computer configuration, encrypt plain image of size 357×317with the encryption scheme presented in [11] will cost 0.3511s around. However, it can be seen from Table 3 that the time cost is 3.6589s around for encrypting image of size 256×256 by ZX2014 with one round, which is about 10 times slower that the scheme in [11]. Even taking into account the parallel nature of Stage II and III in ZX2014 with six levels of pipelined execution mode, the time consumed is still greater than 0.6s. Needless to mention that the encryption/decryption time will be multiplied when the number of rounds is more than 1.

(4) The encryption algorithm of ZX2014 is weak sensitive to part of secret keys.

Because the precision of double type in MATLAB is limited, the precision of key_1, key_2, key_3 and key_5 is taken as 10^{-14} (in [10], it is 10^{-16}). Below, we will examine the sensitivity of the secret key.

Firstly, do 100 trials and randomly generate a secret key K_1 in each trial, and change the value of key_1, key_2, key_3 or key_5 by 10^{-14} to get a new secret key named by K_2. Set the number of rounds to 1 for ZX2014, encrypt the image (as shown in Fig. 2a) using ZX2014 with keys K_1 and K_2 to get two cipher images, denoted by C_1 and C_2, respectively. And then calculate the NPCR and UACI based on C_1 and C_2. At last, calculate the average values of NPCR and UACI for the 100 trials.

Secondly, do 100 trials and in each trial change the value of key_{41} or key_{42} by 1 to calculate the average values of NPCR and UACI.

Thirdly, do 100 trials and in each trial change the value of key_6 by 1 to calculate the average values of NPCR and UACI.

Finally, set the number of rounds to 2 for ZX2014 and repeat the above tests to get a new group of NPCR and UACI.

The test results are listed in Table 4.

Table 4. Sensitivity test results for secret keys.

No. of rounds	key_1, key_2, key_3, key_5		key_{41}, key_{42}		key_6	
	1	2	1	2	1	2
NPCR (%)	94.3456	98.5647	89.8703	99.4934	100.0000	98.8924
UACI (%)	31.6286	33.1208	28.0381	33.4066	1.7888	30.9464

As can be seen from Table 4, the calculated values of NPCR and UACI are way different from their theoretical values, it indicates that the encryption algorithm is weak sensitive to the changes of key_{41}, key_{42} and key_6 when the number of rounds is 1 in ZX2014. Especially, small changes of key_6 will make all of the pixels in the cipher images change (NPCR=100%), but the changes are tiny (UACI=1.7888%). Meanwhile, key_{41}, key_{42} and key_6 are part of direct target secret keys for attacker, so their poor sensitivity makes them very vulnerable.

(5) For the Stages II and III of ZX2014, there're special images can remain unchanged even go through these two stages. For example, the images whose pixel values are all identical will keep unchanged after permutated by the algorithm of Stage II; the all-0s and all-255s images remain unchanged after permutated by the joint algorithms of Stages II and III. Furthermore, any image permutated by the joint algorithms of Stages II and III has the unchanged number of bit 1. These are weaknesses for the chosen plaintext attack.

(6) ZX2014 has an interesting loophole. If the plain image is all-0s, and the number of round is 1, the corresponding cipher image will have identical pixel value, i.e. key_6. This is because the Stages II and III have no permutation effect on the images when they are all-0s image. And from Eq. (4), we can get $\boldsymbol{F}(j)=\boldsymbol{F}_0$, $j=1,2,..,64$. This loophole even makes ZX2014 with two rounds unsafe for the chosen plaintext attack.

Through the above analysis, we can see that the encryption speed is slow, but the security is stronger in ZX2014 when the number of rounds is more than 2. It will take a very long time to crack it using personal computers though the slowness make it cannot be applied in the actual communications. However, Ref. [10] implies that ZX2014 can work well with one round. In the following section, we will discuss the crack on ZX2014 with one round using the chosen plaintext attack method on personal computer.

4. Chosen Plaintext Attack on ZX2014 with One Round

From Section 2, we can see that the attacker can break ZX2014 only by attacking the equivalent keys of key_{41}, key_{42}, key_6, J, idx, $PM(i)$, $i=1,2,\ldots,8$ and X. The attack algorithm is discussed in details as follows:

(1) Attack key_6

Select all-0s image as the plain image, and encrypt the plain image by ZX2014 to get a cipher image, denoted by C. Then, the values of all the pixels in C are key_6.

(2) Attack part of $idx(i)$, $i=1,2,\ldots,L$, $L=N^2/64$.

Step 1. Select all-254s ('1111 1110' in binary) image as the plain image. Encrypt the plain image with ZX2014 to get a cipher image, denoted by C_1. Divide C_1 into small image blocks all of size 8×8, denoted by $CB_1(i)$, $i=1,2,\ldots,L$. For all of the i, when $idx(i)=1$, 3 or 4, the corresponding blocks of $CB_1(i)$ are identical; for $idx(i)=2$, $idx(i)=5$, and $idx(i)=6$, the corresponding blocks of $CB_1(i)$ are identical too, respectively. Since the value of $idx(i)$ are range in $\{1,2,3,4,5,6\}$, $CB_1(i)$,$i=1,2,\ldots,L$ have only four different types of matrices (image blocks). In all the block of $CB_1(i)$, 3/6 of them correspond to the $idx(i)$ whose values are range in $\{1,3,4\}$; others correspond to the $idx(i)$ whose values are range in $\{2,5,6\}$.

Step 2. Select all-126s ('0111 1110' in binary) image as the plain image. Encrypt the plain image with ZX2014 to get a cipher image, denoted by C_2. Divide C_2 into small image blocks all of size 8×8, denoted by $CB_2(i)$, $i=1,2,\ldots,L$. For all of the i, when $idx(i)=1$, 2, 3 or 4, the corresponding blocks of $CB_2(i)$ are identical; when $idx(i)=5$ or 6, the corresponding blocks of $CB_2(i)$ are identical too. Since the value of $idx(i)$ are range in $\{1,2,3,4,5,6\}$, $CB_2(i)$,$i=1,2,\ldots,L$ have only two different types of matrices (image blocks). In all the block of $CB_1(i)$, 4/6 of them correspond to the $idx(i)$ whose values are range in $\{1,2,3,4\}$; others correspond to the $idx(i)$ whose values are range in $\{2,5,6\}$.

Comparing the idx in the above two steps, we can distinguish the i-s corresponding to $idx(i)=\{1,3,4\}$, $idx(i)=2$, or $idx(i)=\{5,6\}$. To simplify the below discussion, we assume that $idx(i_1)=2$, where, i_1 is a certain i, and i_1 will be used to attack X in following operation.

(3) Attack X.

Step 1. Sequentially select all-2^k, $k=0,1,2,\ldots,7$ as the plain images, denoted by P_{k+1}, $k=0,1,2,..,7$, and encrypt then with ZX2014 to get their corresponding cipher images, denoted by C_{k+1}, $k=0,1,2,\ldots,7$, respectively. Divide P_{k+1} into small image blocks of size 8×8, denoted by $PB_{k+1}(i)$, $i=1,2,\ldots,L$. Divide C_{k+1} into small image blocks of size 8×8, denoted by $CB_{k+1}(i)$, $i=1,2,..,L$.

Step 2. For each k, choose the i_1-th block $PB_{k+1}(i_1)$, and convert each element of $PB_{k+1}(i_1)$ from 2^k to 2^{7-k} to get a matrix, denoted by $EB_{k+1}(i_1)$, and then expand $EB_{k+1}(i_1)$ row by row to obtain a vector of length 64, denote by $E_{k+1,i1}(j)$, $j=1,2,\ldots,64$.

This step is also described as: Generate eight vectors of length 64, denoted by $E_{k+1,i1}(j)$, $j=1,2,\ldots,64$, $k=0,1,2\ldots,7$, whose elements are all set to 2^{7-k}.

Step 3. For each k, pick up the i_1-th cipher image block $CB_{k+1}(i_1)$, and expand this block row by row to get a vector of length 64, denoted by $F_{k+1,i1}(j)$, $j=1,2,\ldots,64$.

Step 4. According to the algorithm in Stage III, there are following relationship exists:

$$F_{k+1,i1}(j)=F_{k+1,i1}(j\text{-}1) \oplus X(j) \oplus (E_{k+1,i1}(j)+X(j) \bmod 256), \quad j=1,2,...,64 \tag{7}$$

Where, $k=0,1,\ldots,7$. We can get the values of $X(j)$, $j=1,2,\ldots,64$ by solving Eq. (7).

Note: For any $j \in \{1,2,\ldots,64\}$, there's existing $X(j) \oplus (E_{k+1,i1}(j)+X(j) \bmod 256)=(X(j)+128) \oplus (E_{k+1,i1}(j)+X(j)+128 \bmod 256)$, which is workable regardless of the value of $E_{k+1,i1}(j)$. Therefore, for each j, resolve Eq. (7) will get two legal values for $X(j)$, which differ by 128. This means that both $X(j)$ and $X(j)+128 \bmod 256$, $j=1,2,\ldots,64$ are legal equivalent keys, and this also is a loophole of ZX2014.

(4) Attack PM, key_{41} and key_{42}.

Step 1. Search the i_2 which satisfying the following two conditions:

Condition 1: Select an all-0s image as the plain image except that on a certain position (u,v) whose pixel value is changed from 0 to 255 (0xFF in hexadecimal). Encrypt this image to get a cipher image, denoted by C. Divide C from left to right and top to bottom into small image blocks of size 8×8, and the i_2-th block contains pixels whose values is not equal to key_6.

Condition 2: $idx(i_2)=2$.

In vector idx, about 1/6 of the elements have a value of 2, so theoretically, changing the values of six adjacent pixels in the plain image sequentially, we can find a position (u,v), whose corresponding pixel in the cipher image will fall into the i_2-th block. In the actual experiment, the time of this process may be slightly larger than 6. Therefore, we need to choose about six pieces of plain images in this step, and execute the encryption algorithm of ZX2014 about 6 times.

Step 2. In the i_2–th block of C, find the first element whose value is not equal to key_6, and denote its position by (row,col).

Step 3. Construct a small image block of size 8×8 and make its elements are all-zero except that the pixel in position (row,col) is set to 255 (0xFF in hexadecimal). Then expand this image block to a bit cube of size 8×8×8, denoted by DB_{i2}.

Step 4. Convert the i_2–th block of C to a vector of length 64, denoted by $F_{i2}(j)$, $j=1,2,\ldots,64$. Based on the obtained X and key_6, calculate a new vector, denoted by $E_{i2}(j)$, $j=1,2,\ldots,64$, according to the following formula.

$$E_{i2}(j)=(256+(\mathrm{Fi2}(j-1) \oplus \mathrm{Fi2}(j) \oplus \mathrm{X}(j))-\mathrm{X}(j)) \bmod 256 \quad (8)$$

Step 5. Rearrange vector E_{i2} to get a matrix of size 8×8, and then expand it to a bit cube of size 8×8×8, denoted by DD_{i2}. Obviously, DD_{i2} is the resultant bit-cube obtained from DB_{i2} by the algorithm in Step 4 of Stage II, in ZX2014.

Step 6. According to the reverse operation in Step 4 of Stage II in ZX2014, we can obtain the values of PM, key_{41} and key_{42} from DD_{i2} and DB_{i2}.

(5) Attack the indeterminate part of idx.

For a certain i,$i=1,2,\ldots,L$, if $idx(i)=\{1,3,4\}$, then let $idx(i)=1$ because the effect of the left or right turning of bit-cube can be substituted by the scrambling effect or vector J. If $idx(i)=\{5,6\}$, the value of $idx(i)$ can be defined by the following steps:

Step 1. Select all-15s image as the plain image, denoted by P. Encrypt P with ZX2014 to get a cipher image, denoted by C. Divide C from left to right and top to bottom into small image blocks all of size 8×8, denoted by F_i, $i=1,2,\ldots,L$.

Step 2. For a certain i, $i=1,2,..,L$, if $idx(i)=\{5,6\}$, $F_i(1,1)$ will be transformed into a new value by Eq. (8) with the obtained X and key_6, denote the new value by $E_i(1,1)$. If $E_i(1,1)=0$, then $idx(i)=5$; If $E_i(1,1)=255$, then $idx(i)=6$.

(6) Attack J.

Assume that $A_1(k)$, $k=1,2,\ldots,N^2$ is a vector of length N^2. Attack J with steps as follows:

Step 1. If $k<256$, then let $A_1(k)=k$; If $k>=256$, then let $A(k)=0$.

Step 2. Convert A_1 into a matrix of size $N×N$, denoted by P. Encrypt P with ZX2014 to get a cipher image, denoted by C. With the help of the obtained X, key_6, key_{41}, key_{42}, idx and PM, execute the reverse operations on C as the description in Stages III and II of ZX2014 to get a new image matrix, denoted by D. Expend D row by row to get a vector of length $N×N$, denoted by A_2.

Step 3. Compare the elements of A_1 and A_2 to find those corresponding elements whose values are equal to get the values of $J(k)$, $k=1,2,..,255$.

Now let $A_1(k)=k-255$, $256<=k<256×2$; $A_1(k)=0$, otherwise. According to the similar steps described above, we can get the values of $J(k)$, $256<=k<511$. So, using the above methods we need to select floor(N^2/255)+1 pieces of images to get the whole values of $J(k)$, $k=1,2,\ldots,N^2$.

Through the above analysis, cracking the equivalent keys of ZX2014 need 19+floor(N^2/255) pieces of chosen images. So, cracking the plain image of size 256×256 need 276 chosen images.

Note that in the cracked equivalent keys, key_{41}, key_{42}, key_6 and PM are identical to the original secret keys, while there are two legal values for each element in the cracked vector X, and the cracked idx and J are different from the original secret keys due to the permutation effect of $idx(i)=3$ or 4 being substituted by the scrambling of J.

5. Simulation Results

We did multiple tests to confirm the availability of proposed crack algorithm. Without loss of generality, assume the secret key of ZX2014 is $K=(0.5574,0.9015,0.8421,2,7,0.7893,169)$, and the plain images are taken as Lena, Baboon and Pepper all of size 256×256. The original images and their corresponding cipher images generated by ZX2014 are as shown in Figs. 3a-3c and Figs. 3d-3f respectively. Cracking ZX2014 with the chosen plaintext attack method described in Section 4, we can get the equivalent secret keys of key_{41}, key_{42}, key_6, PM, idx, X and J, and then use them to decipher Figs. 3d-3f to get the recovered images as shown in Figs. 3g-3i, respectively. From Figs. 3a-3c and 3g-3i we can see the recovered images are identical to the original images respectively. The time consumed is about 1577s.

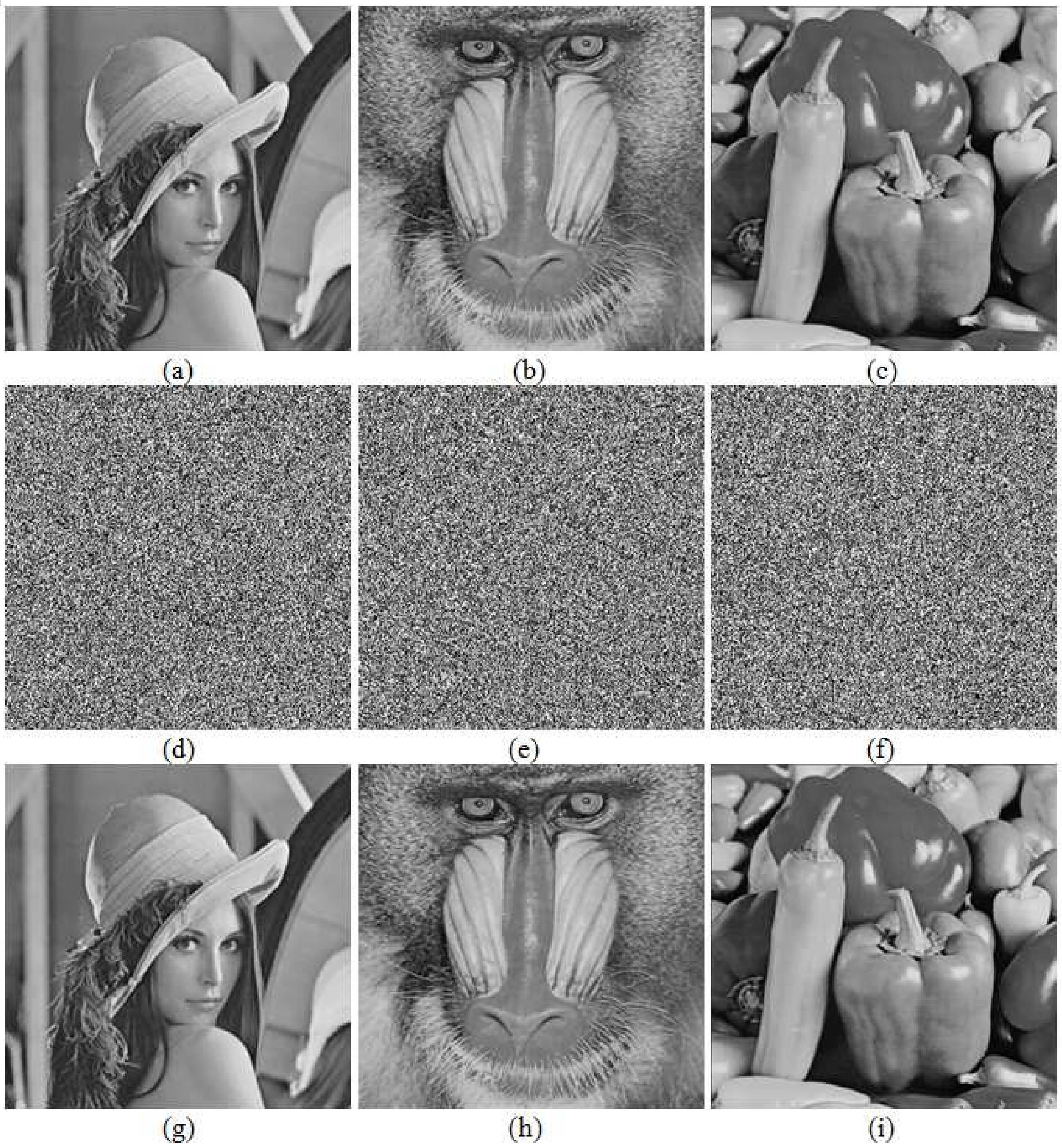

Figure 3. Simulation results. (a)-(c) Plain images of Lena, Baboon and Pepper, respectively; (d)-(f) Cipher images of (a)-(c), respectively; (g)-(i) Cracked images of (d)-(f).

When the cipher images are disturbed by the salt and pepper noise, the cipher images still can be cracked with the obtained equivalent secret keys to get the recovered images, and the effect is similar to the results as shown in Table 1 and Fig. 2.

6. Conclusion

This paper analyzed the encryption scheme of ZX2014 in detail, and pointed out that there are drawbacks such as low encryption speed and security loopholes. This paper cracked ZX2014 with one round successfully. Time consumed for the image of size 256×256 is about 1577s. The cracked equivalent secret keys are not exactly identical to the original keys, which show that there are multiple equivalent keys in the key space of ZX2014. Our study work demonstrates that ZX2014 is weak on security and cannot be applied in the actual communications.

Acknowledgement

This work was fully supported by the Natural Science Foundation of Jiangxi Province (Grant No. 20122BAB201036).

References

[1] J. Fridrich. Symmetric ciphers based on two-dimensional chaotic maps. International Journal of Bifurcation and Chaos, 1998, 8(6): 1259-1284.

[2] N. K. Pareek, V. Patidar, K. K. Sud. Cryptography using multiple one-dimensional chaotic maps. Communications in Nonlinear Science and Numerical Simulation, 2005, 10(7): 715-723.

[3] G. R. Chen, Y. Mao, C. K. Chui. A symmetric image encryption scheme based on 3D chaotic cat maps. Chaos, Solitons and Fractals, 2004, 21(3): 749-761.

[4] G. Alvarez, S. J. Li. Some basic cryptographic requirements for chaos-based cryptosystems. International Journal of Bifurcation and Chaos, 2006, 16(8): 2129-2151.

[5] J. S. A. Eyebe Fouda, J. Y. Effa, S. L. Sabat, M. Ali. A fast chaotic block cipher for image encryption. Commun. Nonlinear Sci. Numer. Simulat., 2014, 19(3): 578-588.

[6] G. Ye. A block image encryption algorithm based on wave transmission and chaotic systems. Nonlinear Dyn. 2014, 75(3): 417-427.

[7] P. Cheng, H. Yang, P. Wei, W. Zhang. A fast image encryption algorithm based on chaotic and lookup table. Nonlinear Dynamics, 2015,79(3): 2121-2131.

[8] A. N. Pisarchik, M. Zanin. Image encryption with chaotically coupled chaotic maps. Physica D, 2008, 237(20): 2638-2648.

[9] D. Arroyo, S. Li, J. M. Amigó, G. Alvarez, R. Rhouma. Comments on "Image encryption with chaotically coupled chaotic maps". Physica D, 2010, 239(12): 1002-1006.

[10] Y. Zhang, D. Xiao. An image encryption scheme based on rotation matrix bit-level permutation and block diffusion. Commun Nonlinear Sci Numer Simulat, 2014, 19(1): 74-82.

[11] Y. Zhang. A chaotic system based image encryption algorithm using plaintext-related confusion. TELKOMNIKA, 2014, 12(11): 7952-7962.