Cryptanalysis of an Image Block Encryption Algorithm Based on Chaotic Maps

Yunling Ma, Chengqing Li

College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China

Abstract

Recently, an image block encryption algorithm was proposed based on some well-known chaotic maps. The authors claim that the encryption algorithm achieves enough security level and high encryption speed at the same time. However, we find that there are some critical security defects in the algorithm. In this paper, we give a thorough security analysis on the algorithm from the perspective of modern cryptography. Given one pair of known plain-image and its corresponding cipher-image, the attacker can obtain an equivalent secret key to successfully decrypt the other cipher-images encrypted with the same secret-key. In addition, each security metric adopted in the security evaluation on the algorithm is questioned. The drawn lessons are generally applicable to many other image encryption algorithms.

Keywords: image encryption, cryptanalysis, plaintext attack, image privacy, multimedia content protection.

1. Introduction

With the development of the Internet and social media platform, people often share their photos on the Internet with more and more higher frequency. However, these photos may contain rich privacy information. If they are accessed by some unauthorized person, serious consequences may happen for the involved parties. Therefore, in the process of image transmission on the network, protecting image information from being leaked and stolen has become a focus in the field of information security [1, 2]. In recent years, more and more image encryption algorithms have been proposed to withstand the challenge [3, 4, 5].

A number of image encryption algorithms based on various chaotic systems are proposed due to their special characteristics, such as sensitivity to change of initial condition, unpredictability, randomness and high complexity [6, 7]. For example, Hua’s encryption algorithm proposed in [7] uses 2-D Logistic-adjusted-sine map as a pseudorandom number generator (PRNG); 4-D piecewise logistic map with coupled parameters are adopted as a PRNG in [8]. However, in the design of image encryption algorithm based on chaotic systems, many designers ignored their negative side: obvious dynamics degradation of any chaotic system in a digital domain [9, 10, 11]. This defect may cause the invalidity of chaotic domain, so many insecure encryption algorithms have been successfully cracked [12, 13]. It can be seen that a chaos-based encryption algorithm may own several special security defects that do not exist in the non-chaotic encryption methods [12, 14, 15].

In [16], a chaotic image encryption algorithm was proposed, where three different chaotic maps are used as a PRNG for controlling pixel shuffling, blocking size, and value encryption. A simple chaotic map is used to determine block size and generate pseudo-random binary sequence for each block. Especially, the sum of pixels of plain-image is used to build up a sensitivity mechanism of the encryption result on the plaintext. However, we found the security defects of the chaos-based PRNG and canceled the sensitivity mechanism. As for one round version of the algorithm, we can derive the secret-key with a chosen-plaintext attack. In addition, each used security metric is questioned from the point of view of modern cryptanalysis.

The rest of the paper is organized as follows. Section 2 presents a description of the analyzed algorithm. Detailed cryptanalytic results are given in Sec. 3. The last section concludes the paper.

2. The image block encryption algorithm based on three chaotic maps

Assume that the gray-scale plain-image is denoted as matrix $I$ of size $M \times N$. As specified in [16], three chaotic maps are used, namely Arnold map, Baker map, and Logistic map. They are used for permuting position of plain-image, dividing the plain-image into four blocks, and gen-
erate the encryption sequences \( \{m_p\} \) and \( \{t_p\} \), respectively. The encryption algorithm can be described as follows.

- Generating permutation position with Arnold map

\[
\begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} 1 & b \\ a & ab + 1 \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix} \mod N,
\]

where \( a, b \) and \( N \) are parameters; \((i, j)\) represents the position of the pixel at the \( i \)-th row and the \( j \)-th column of the plain-image \( I \); \((p, q)\) represents the permutation position of \((i, j)\).

- Calculating block size with Baker map

\[
(x_{n+1}, y_{n+1}) = \begin{cases} (x_n, y_n) & \text{if } 0 < x \leq \mu; \\ ((x_n - \mu) \cdot \mu', y_n + \mu) & \text{if } \mu < x \leq 1, \end{cases}
\]

where \( \mu \) is the control parameter of the equation, \( \mu' = 1 - \mu \). Set the initial value of Baker map (2) as \((x_0, y_0)\), and iterate it \( n \) times. Then, transform the generated sequences \( \{x_i\}_{i=1}^n \) and \( \{y_i\}_{i=1}^n \) into integer vectors by functions \( f(x) = \lfloor M \cdot x \rfloor \) and \( f(x) = \lfloor N \cdot x \rfloor \), respectively. Finally, the last pair of sequences \((u, v)\) is selected as the block size of the first block.

- Generating the random encryption sequence \( \{m_p\} \) and \( \{t_p\} \) with Logistic map

\[
g(x) = r \cdot x \cdot (1 - x),
\]

where \( x \) is the state variable, \( r \) is the control parameter. Setting an initial value \( z_0 \), a sequence \( \{z_i\} \) can be generated by iterating Eq. (4), which is further transformed into an integer sequence \( \{m_p\} \) via

\[
h_1(x) = \lfloor x \cdot 10^5 \rfloor \mod 256.
\]

Then, set

\[
\begin{align*}
s_0(2) &= (s_0(1) + 0.5) \mod 1; \\
s_0(3) &= (s_0(1) + s_0(2)) \mod 1; \\
s_0(4) &= (s_0(2) + s_0(3)) \mod 1,
\end{align*}
\]

where \( s_0(1) \) is a sub-key. Setting \( s_0(1) \) as an initial condition, generate sequence \( \{s_i\} \) by iterating integer logistic map

\[
g^*(x) = 4 \cdot x \cdot (256 - x)/256.
\]

Then, transform it into integer sequence \( \{t_p\} \) by

\[
h_2(x) = \lfloor x \cdot 10^6 \rfloor \mod 256.
\]

- The encryption procedure:

\( \text{Step 1:} \) Permute the relative positions of every pixel of \( I \) with the permutation relation determined by Arnold map (1) and obtain shuffled image \( E \).

\( \text{Step 2:} \) Divide the shuffled image \( E \) into four sub-images based on Baker map (2). Then, exchange the position of pixel at entry \((0, 0)\) with that at entry \((u, v)\). The initial values will be selected to be \((x_0 + \text{mean}(I)/256 \mod 1)\) and \((y_0 + \text{mean}(I)/256 \mod 1)\), where \( \text{mean}() \) denotes the mean value of pixels of plain-image \( I \).

\( \text{Step 3:} \) Generate the random encryption control sequence \( \{t_p\} \) and \( \{m_p\} \) utilizing Logistic map (3) by using initial value \((z_0 + \text{mean}(I)/256 \mod 1)\).

\( \text{Step 4:} \) Encrypt the four sub-images by the generated sequences \( \{t_p\} \) and \( \{m_p\} \) via

\[
a_{ij}^* = \begin{cases} a_{ij} + m_p \cdot t_p \mod 256 & \text{if } m_p = 255; \\
a_{ij} + (m_p + 1) \cdot t_p \mod 256 & \text{otherwise}. \end{cases}
\]

(7)

\( \text{Step 5:} \) Combine all the encrypted sub-images and obtain encrypted image \( I' \).

- The decryption procedure is the inverse version of the encryption algorithm by using the same secret-key, namely

\[
a_{ij} = \begin{cases} a_{ij}^* - m_p \cdot t_p \mod 256 & \text{if } m_p = 255; \\
a_{ij}^* - (m_p + 1) \cdot t_p \mod 256 & \text{otherwise}. \end{cases}
\]

3. Cryptanalysis

3.1. Known-plaintext attack

In the original encryption algorithm, it is divided into three parts to encrypt the original image. Include image shuffling, image blocking, and sub-image encryption. The flow diagram is shown in Fig.1.

Through the analysis of the original paper, we can know that the generation of random sequence is closely related to the mean value of pixels of the original image. If the same secret keys are used, different random sequences will be generated for different mean value of pixels. However, although the images are different, it is still possible to produce the same mean value of pixels. Therefore, as long as the same mean value of pixels is used, it can be attacked with the same secret keys.

Given one pair of plain-image and corresponding encrypted image, when the following two conditions are true, we can attack the given ciphered image. Condition 1: the
ciphered image uses the same secret keys \((x_0, y_0, z_0, r)\) as the given encrypted image; condition 2: the decipher image has the same sum of pixels as the given plain-image. For the sake of illustration, we can transform Eq. (7) into
\[
a'_{ij} = (a_{ij} + P_{ij}) \mod 256, \quad (8)
\]
where \(P\) represents the equivalent encryption image, the results of the interaction of \(t_p\) and \(m_q\); \(P_{ij}\) is used to change the pixel value at the \(i\)-th row and the \(j\)-th column of the shuffled image \(E\).

### 3.2. Locating the permutation position of the first pixel of the plain-image

Assume that the sum of pixels of the plain-image is denoted as \(\eta\). If the generation of random sequence has nothing to do with the sum of pixels of the plain-image, we know that as long as we use an image with all zero pixels, we can get the equivalent encryption image \(P\). However, since the encryption algorithm is related to the sum of pixels of the plain-image, we need to ensure that the sum of pixels is the same as the plain-image, so we need to adjust the value of the first pixel to \(\eta\). There is only one different element between the encrypted image obtained from such an image and the equivalent encryption image \(P\) of the plain-image \(I\). Because of the permutation operation, the different element is the value at permutation position of the first pixel in the plain-image \(I\).

Therefore, we first need to find the permutation position of the first pixel in the plain-image, then obtain the equivalent encryption image \(P\) in Eq. (8). So, we choose three pairs of images whose sum of pixels are the same as the plain-image.

The permutation position of the first pixel in the plain-image can be computed by the following steps.

**Step 1: Choose a plain-image \(q_1\) to attack with all zero pixels except \(q_1(1, 1) = \eta\) and the corresponding cipher image is supposed to be \(Q_1\), where**

\[
q_1 = \begin{pmatrix} \eta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (9)
\]

**Step 2: Choose a plain-image \(q_2\) to attack with all zero pixels except \(q_2(1, 1) = \eta - 1\) and \(q_2(1, 2) = 1\), and the corresponding cipher image is supposed to be \(Q_2\), where**

\[
q_2 = \begin{pmatrix} \eta - 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (10)
\]

**Step 3: Choose a plain-image \(q_3\) to attack with all zero pixels except \(q_3(1, 1) = \eta - 1\) and \(q_3(1, 3) = 1\), and the corresponding cipher image is supposed to be \(Q_3\), where**

\[
q_3 = \begin{pmatrix} \eta - 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (11)
\]

**Step 4: Compare \(Q_1\) and \(Q_2\). The result of operation is recorded as \(Q_{12}\). We can find that there should be two different pixels between \(Q_1\) and \(Q_2\). Therefore, there would be two zero values in \(Q_{12}\). \(Q_{12}\) can be expressed as**

\[
Q_{12} = (Q_1 \equiv Q_2).
\]

We know that the positions of the two generated zero elements are the permutation positions of the (1, 1) and (1, 2) of the plain-image.

**Step 5: Compare difference between \(Q_3\) and \(Q_4\). The result of operation is recorded as \(Q_{13}\). We can find that there**
should be two different pixels between \(Q_1\) and \(Q_3\). Therefore, there would be two zero values in \(Q_{13}\). \(Q_{13}\) can be expressed as
\[
Q_{13} = (Q_1 == Q_3).
\]

We know that the positions of the two generated zero elements are the permutation positions of \((1,1)\) and \((1,3)\) of the plain-image.

**Step 6:** Compare the locations of step 4 with step 5. Find the same location, where the element is zero in \(Q_{13}\) and \(Q_{13}\), and that position is the permutation position of \((1,1)\).

The specific legend is shown in Fig. 2. In this legend, assuming that the sum of pixels of the plain-image is 575 \((\eta = 575)\), the three plain-images selected are shown in Fig. 2 in turn. The calculation method described above is shown in detail in the schematic diagram. Therefore, we can finally get the permutation position of the first pixel.

### 3.3. Finding the equivalent encryption image \(P\) and obtaining the shuffled image \(E\)

Encrypting the plain-image \(I\), and the corresponding encrypted image is \(I^*\). Based on the above analysis, we know \(Q_1\) is identical to \(P\) except \(Q_1(u,v)\), because of its value is modified. So, the pixel of \(Q_1(u,v)\) subtracts the sum of pixels \((\eta)\) of the plain-image, then we can derive the equivalent encryption image \(P = Q_1\), where
\[
Q_1(u,v) = (Q_1(u,v) - \eta) \mod 256. \tag{12}
\]

We have obtained the equivalent encryption image \(P\), for a given encrypted image, such as a cameraman, the encrypted image \(I^*\) subtracting the equivalent encryption image \(P\), we can derive the shuffled image \(E\) by
\[
E = (I^* - P) \mod 256. \tag{13}
\]

### 3.4. Obtaining the original image by permutation rule

After getting the shuffled image, we can use the method in [17] to get the permutation rule, and then obtain the plain-image.

If the plain-image \(I\) is stretched into a vector row by row, the pixel locations would be a vector by \(A_0 = \{k\}_{k=0}^{M \times N - 1}\), which can be expanded to a 2-digit representation in base 256 as follows. The location matrix \(O\) can be expressed as
\[
O = \begin{bmatrix}
(0)(0) & (0)(1) & (0)(2) & \cdots & (0)(255) \\
(1)(0) & (1)(1) & (1)(2) & \cdots & (1)(255) \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
(254)(0) & (254)(1) & (254)(2) & \cdots & (254)(255) \\
(255)(0) & (255)(1) & (255)(2) & \cdots & (255)(255)
\end{bmatrix}_{M \times N}.
\]

Each element in \(O\) is a number in base 256. Then, two chosen plain-images with entries 0, 1, 2, 254, 255 are obtained by splitting matrix \(O\) into two images
\[
O_0 = \begin{bmatrix}
0 & 1 & 2 & \cdots & 255 \\
0 & 1 & 2 & \cdots & 255 \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
0 & 1 & 2 & \cdots & 255
\end{bmatrix}_{M \times N}
\]
and
\[
O_1 = \begin{bmatrix}
0 & 0 & 0 & \cdots & 0 \\
1 & 1 & 1 & \cdots & 1 \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
254 & 254 & 254 & \cdots & 254 \\
255 & 255 & 255 & \cdots & 255
\end{bmatrix}_{M \times N}
\]

To keep the sum of pixels of these images consistent with the plain-image, as Sec. [5.2], the first pixels in two chosen plain-images can be calculated by
\[
\begin{align*}
\text{sum}O_\text{e} &= \sum_{i=1}^{N} \sum_{j=1}^{M} O_\text{e}(i, j) - O_\text{e}(1, 1), \\
O_\text{e}(1, 1) &= \eta - \text{sum}O_\text{e},
\end{align*}
\]
where \(e = (0,1)\).

The permutation cryptanalysis steps are expressed as follows:

**Step 1:** Encrypt two images \(O_0\) and \(O_1\), and obtain corresponding encrypted images \(C_0\) and \(C_1\).

**Step 2:** Obtain the shuffled images \(O_{M0}\) and \(O_{M1}\) in Sec. [5.3]

**Step 3:** Derive the permutation rule \(l\).

First, we should get \(O_l = [O_l(i, j)]_{i=1,j=1}^{M \times N}\) calculated by
\[
O_l = 256O_{M0} + O_{M1}.
\]

Because the first pixels in \(O_0\) and \(O_1\) are adjusted, the element \(O_{l}(u,v)\) is wrong. But, the rest locations are correct; the correct \(O_{l}(u,v)\) can be calculated by
\[
\begin{align*}
\text{sum}O_\text{e} &= \left\{ \sum_{i=1}^{M} \sum_{j=1}^{N} O_\text{e}(i, j) - O_\text{e}(u, v), \\
O_\text{e}(u, v) &= \left\{ \sum_{k=0}^{M \times N - 1} k \right\} - \text{sum}O_\text{e},
\end{align*}
\]

Therefore, the permutation rule \(l\) described by vector \(L_0\) can be obtained by stretching \(O_l\) to a vector row by row. Making use of \(L_0\), we can obtain the original plain-image \(I\) from the shuffled image \(E\). The specific legend is shown in Fig. 3.

From the figure, we can see that the permutation position of the first pixel is \((4,4)\), which is consistent with the result calculated in section [5.2] which proves that this method is correct. The sum of pixel of the legend image is \(575(\eta = 575)\).
3.5. The deciphering process of a given cipher image

Based on the above analysis, the decryption steps can be described as follows.

Step 1: Locate the permutation position of the first pixel of the plain-image.
Step 2: Find the equivalent encryption image \( P \).

Step 3: Get the shuffled image.
Step 4: Obtain the plain-image.

In order to testify the correctness of our attack method, we performed the experiment with some secret keys. When the secret keys and parameter are set as 166, namely \( x_0 = 0.123, y_0 = 0.456, n = 10000, \delta_0(1) = 0.789, \zeta_0 = 0.147 \)
and $r = 3.999$. The plain-image “cameraman” and its results of encryption and decryption are shown in Fig. 4 and Fig. 5.

Evaluate the uniformity of an encrypted image by calculating the variance of histogram. In [16], it is said that “The lower value of variances indicates the higher uniformity of encrypted images. Comparing with the variance of histogram of the plaintext image, we have that the proposed algorithm is efficient and can resist such statistical attacks effectively”. In fact, the variance of histogram can not measure the number of different pixel value in the encrypted histogram. Such as: the variances of two histograms “2, 2, 3, 3, 4, 7” and “2, 2, 3, 3, 5, 6” are different, but their number of different combinations are the same. Besides, in [19] can prove that the variance of histogram of encrypted image obtained by insecure encryption algorithm can also obtain very low.

3.6. Security defects

The original paper uses the general security evaluation criteria in most of the papers to prove the security of the proposed encryption algorithm. However, we try to prove that these evaluation criteria are not completely correct. We will refute and prove our opinions one by one according to the order of security analysis in the original paper.

- Histogram analysis
  In [16], it is stated that “The readability of image would be less effective with the smoother histogram distribution”. As shown in [10], although attackers can not get some meaningful information from the pixels of uniform histogram, they can restore some important statistical information about the plain-image by changing the counting objects of the histogram from pixel to bit. In addition, an example shows that the histogram of two encrypted graphs remains unchanged, but this encryption algorithm is also safe enough for some specific scenes, which has proposed in paper [18]. Therefore, the two histograms calculated in terms of pixel shown in [16] are not enough to prove that the proposed encryption algorithm has better security.

- Variance of histogram

- Key space analysis
  In [16], the precision of the secret key is fixed as $10^{-14}$. However, as shown in [20], in digital computer, the precision can only be precisely specified by a power of two. If data is represented in floating-point format binary32 or binary64, the distance between adjacent representable numbers is not even, so the length of mantissa fraction and exponential fraction need to be set carefully in [11]. Therefore, it is not entirely correct to say that there are very rich secret keys to resist different kinds of brute force attacks.

- Key sensitivity analysis
  In [16], through changing one of its five keys to a small change of $10^{-14}$, to show the impact of the so-called small changes of the secret key on the encryption and decryption process, and then prove the key sensitivity of the algorithm. However, in the computer, any calculation is based on binary system, so the sensitivity of secret keys is not credible by using the method of decimal system. Observing Eq. (3) and (5), one can see that $x$ and $(1 - x)(x \text{ and } 256 - x)$ are equivalent if Logistic map is implemented in a fixed-point arithmetic domain. Due to the modulo addition in Eq. (4), there may exist even much more equivalent secret keys. Besides these, quantization effects of the digital chaotic map may generate the same iteration orbit for different initial conditions. Therefore, the sensitivity of encryption results to key change is very weak.

- Correlation analysis
  In [16], a popular measurement method is used to calculate the correlation degree of adjacent pixels in the horizontal direction, vertical direction and diagonal direction. They think the ideal value should be close to
0, so the lower the calculated value between adjacent pixels, the better the encryption algorithm. However, there have been objections in recent years. In paper [19], several insecure encryption algorithms are selected to calculate the correlation among adjacent pixels in the horizontal direction, vertical direction and diagonal direction, and the value is close to 0, which proves the error of the measurement method. Finally, they come to a conclusion: “There is no clear (statistic) decision criterion for passing this test.” A visual inspection would thus mistake two insecure ciphers as secure.

- Information entropy analysis

Information entropy is used to measure the “randomness” of an encrypted image and a perfect encrypted image should thus get close to 8. But experiments show that this is only a necessary condition, not a sufficient condition. There are still many encryption algorithms with poor security, but the entropy value of the encrypted image is close to 8 perfectly. The example is proved in [19].

- Resistance to differential attack analysis

Two parameters NPCR and UACI are used to evaluate the sensitivity to plaintext. When the result achieves the ideal value, they think the encryption algorithm is extremely sensitive to plaintext and can resist differential attack effectively for different images. However, the values for four insecure ciphers are above 99%, which is comparable to the best NPCR values found in the literature on chaos-based ciphers. In particular, this value is virtually identical to that of the so far unbroken 2D logistic map encryption algorithm in [19]. Therefore, this criterion for measuring encryption security is not credible.

- Robustness analysis

In original paper, different levels of noise and data loss are used to prove the robustness of the algorithm. However, we find that there is a big hidden danger in the encryption algorithm. Before decryption, the decryptor needs to obtain the transmitted mean value of pixels of the original image accurately. However, in practice, this value is not easily transmitted correctly. Through the experimental simulation, we find that the original image can not be decrypted correctly when the mean value of pixels with different precision is only one bit different. However, we can not guarantee that every bit will not be lost in the transmission process. Therefore, we think this encryption algorithm is not robust enough.

- Algorithm speed analysis

In [16], by comparing several different encryption algorithms, the author gets the conclusion that their algorithm has the best operation speed. Actually, the fast speed is built on sacrificing security instead of better structure. In Eq. (4) and (6), authors use the general integer conversion functions, namely

$$f_m(x) = f(10^m \cdot x) \mod D,$$

where $m$ and $D$ are positive integers; $f(x)$ is an quantization function, for example ceil, round and floor. In computer, constant multiplication is performed by a series of bitwise shifts and addition operations. Therefore, the computational complexity of the conversion is proportional to $m$. Only $\left\lfloor \log_2 D \right\rfloor$ bits are available, the other $m\left(\log_2 10\right) - \left\lfloor \log_2 D \right\rfloor$ bits are wasted. Taking Eq. (4) as an example, the utilization percentage of the computation cost on iterating Logistic map (5) is only

$$\frac{\left\lfloor \log_2 D \right\rfloor - \left\lfloor \log_2 10 \right\rfloor}{\left\lfloor \log_2 10 \right\rfloor} \approx \frac{256}{512} = \frac{1}{2}.$$
