

Cryptanalyzing an Image Encryption Algorithm Underpinned by 2D Lag-Complex Logistic Map

Chengqing Li^{a,*}, Xianhui Shen^b, Sheng Liu^c

^a*School of Computer Science, Xiangtan University, Xiangtan 411105, Hunan, China*

^b*MOE (Ministry of Education) Key Laboratory of Intelligent Computing and Information Processing, Xiangtan University, Xiangtan 411105, China*

^c*School of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, Hunan, China*

Abstract

This paper analyzes security performance of an image encryption algorithm using 2D lag-complex Logistic map (LCLM), which adopts it as a pseudo-random number generator, and uses the sum of all pixel values of the plain-image as its initial value to control the random combination of the basic encryption operations. However, multiple factors make the final pseudo-random sequences controlling the encryption process may be the same for different plain-images. Based on this point, we proposed a chosen-plaintext attack by attacking the six encryption steps with a strategy of divide and conquer. Using the pitfalls of 2D-LCLM, the number of required chosen plain-images is further reduced to $5 \cdot \log_2(MN) + 95$, where MN is the number of pixels of the plain-image.

Keywords: Chosen-plaintext attack, cryptanalysis, image encryption, chaotic cryptography, image privacy.

1. Introduction

With the popularization of multimedia capture devices and the mobile Internet, more and more people become enjoying sharing and transmitting photos via a social platform, such as WeChat, Instagram and Facebook. Meanwhile, due to the openness of the network, multimedia information itself or the contained privacy may be leaked during the transmission process [1]. Such underlying threats are transparent for most people. Therefore, ensuring security and privacy of multimedia data, especially that of digital images, with proper balancing point with usability has become needs of everyone living with cyberspace. Due to the special characteristics of image data, such as the large amount of data itself and strong correlation existing between neighbouring pixels, it is generally impractical to encrypt the protected object with the classic text encryption algorithms by converting the object into one-dimensional text data. Under this background, a large number of image encryption algorithms were proposed in the past three decades [2, 3, 4]. Unfortunately, some were found insecure of different extents from the perspective of modern cryptanalysis [5, 6, 7, 8, 9, 10]. With the development of neural network technology, the form of attack and defense applied on image data has been changed much [4, 11, 12].

As a cryptosystem can be considered as a complex system defined on a digital domain, various nonlinear sciences, e.g. chaos, synchronization, were adopted as alternative methodologies to design new image encryption algorithms [5, 13, 14, 15]. However, the nice complex dynamics demonstrated in the original infinite-precision domain may be degenerated to a certain extent and even diminished in a finite-precision arithmetic domain [16, 17]. As shown in [6, 10], the randomness of the sequence obtained by iterating a discrete chaotic map on a computer with finite arithmetic precision is much lower than that expected by the designers: the periods of the sequences starting from some initial conditions are even smaller than three (see Fig. 1).

In [18], 2D lag-complex Logistic map (2D-LCLM) is designed to enhance dynamic complexity of the map and counteract its dynamic degradation in digital domain. Then, an image encryption algorithm based on the special Logistic map (IEALM) is designed and tested to demonstrate cryptographic application merits of the map. In the algorithm, the pseudo-random number sequences generated by iterating 2D-LCLM are used to obfuscate and diffuse the plain-image via complex cascade of modulo addition, XOR operation, and bit-level permutation. This paper reevaluated security performance of IEALM and reported the following points: 1) the correlation mechanism between the encryption process and the plain-image is invalid with a non-negligible probability; 2) some intrinsic properties of

*Corresponding author.

Email address: DrChengqingLi@gmail.com (Chengqing Li)

2D-LCLM are disclosed to support a chosen-plaintext attack; 3) the basic encryption parts can be separately attacked with some chosen plain-images in turn; 4) experimental results and complexity analysis of the attack are provided to demonstrate its effectiveness.

The rest of this paper is organized as follows. Section 2 briefly introduces the procedure of IEALM. Section 3 presents the insecurity metrics of 2D-LCLM, and the security vulnerabilities of IEALM, together with some experimental results. The last section concludes the paper.

2. Description of IEALM

The encrypted object of IEALM is an 8-bit RGB image of size $M \times N \times 3$. The algorithm does not consider the specific storage format of the plain-image, and directly splits it into three separate color channels, which are then processed by the same encryption process. For brevity, we describe the encryption procedure for only one channel. The plain-image is scanned in the raster order and then denoted by $I = \{I(i)\}_{i=0}^{MN-1}$. Likewise, the corresponding cipher-image is represented as $I'' = \{I''(i)\}_{i=0}^{MN-1}$. Then, IEALM can be described as follows.

- *The secret key*: two control parameters of 2D-LCLM

$$\begin{cases} x(i+1) = b \cdot x(i) \cdot (1 - z(i)), \\ y(i+1) = b \cdot y(i) \cdot (1 - z(i)), \\ z(i+1) = a \cdot x(i)^2 + y(i)^2, \end{cases} \quad (1)$$

and its two distinct initial conditions $K_1 = (x(0), y(0), z(0))$ and $K_2 = (x'(0), y'(0), z'(0))$, where $a = 2$ and $b \in [1.69, 2)$.

- *Initialization*:

- *Step 1*: Calculate two initial conditions through

$$\begin{cases} K_1 = (0.2 + X_r/10^9, 0.4 + Y_g/10^9, 0.1 + Z_b/10^9), \\ K_2 = (0.3 + X_r/10^9, 0.5 + Y_g/10^9, 0.2 + Z_b/10^9), \end{cases} \quad (2)$$

where X_r , Y_g , and Z_b are the sum of the pixel values of red, green, and blue channels of the plain-image, respectively.

- *Step 2*: Iterate Eq. (1) $2MN + 250$ times from the initial condition K_1 and discard the first 250 elements to get three chaotic sequences $X = \{x(i)\}_{i=0}^{2MN-1}$, $Y = \{y(i)\}_{i=0}^{2MN-1}$ and $Z = \{z(i)\}_{i=0}^{2MN-1}$. Then, generate a sequence $G = \{g(i)\}_{i=0}^{2MN-1}$, where

$$g(i) = \frac{x(i) + y(i) + z(i)}{3}.$$

- *Step 3*: Generate a 4-bit integer sequence $U = \{U(i)\}_{i=0}^{MN-1}$ and two 8-bit integer sequences $V = \{V(i)\}_{i=0}^{MN-1}$ and $W = \{W(i)\}_{i=0}^{MN-1}$ via

$$\begin{cases} U(i) = \lfloor |x(i)| \cdot 10^{15} \rfloor \bmod 16, \\ V(i) = \lfloor \text{Dec}(y(i)) \cdot 10^3 \rfloor \bmod 256, \\ W(i) = \lfloor \text{Dec}(z(i)) \cdot 10^3 \rfloor \bmod 256, \end{cases}$$

where $\text{Dec}(x) = x \cdot 10^3 - \lfloor x \cdot 10^3 \rfloor$.

- *Step 4*: Generate two permutation vectors $T_{1,0} = \{T_{1,0}(i)\}_{i=0}^{MN-1}$ and $T_{2,0} = \{T_{2,0}(i)\}_{i=0}^{MN-1}$, where $T_{1,0}(i)$ and $T_{2,0}(i)$ are the order of $X(i)$ and $X(i + MN)$ in sets $\{X(i)\}_{i=0}^{MN-1}$ and $\{X(i)\}_{i=MN}^{2MN-1}$, respectively. Similarly, obtain six permutation vectors of length MN , $T_{1,1}$, $T_{2,1}$, $T_{1,2}$, $T_{2,2}$, $T_{1,3}$ and $T_{2,3}$, from Y , Z , and G , respectively. Group the above vectors as $T_1 = \{T_{1,0}, T_{1,1}, T_{1,2}, T_{1,3}\}$ and $T_2 = \{T_{2,0}, T_{2,1}, T_{2,2}, T_{2,3}\}$.
- *Step 5*: Repeat *Step 2*, *3* and *4* using initial condition K_2 and produce the counterparts X' , Y' , Z' , G' , U' , V' , W' , $T_3 = \{T_{3,0}, T_{3,1}, T_{3,2}, T_{3,3}\}$ and $T_4 = \{T_{4,0}, T_{4,1}, T_{4,2}, T_{4,3}\}$.

- *The encryption procedure*:

- *Step 1*: Perform modulo addition on I to obtain a sequence $I^* = \{I^*(i)\}_{i=0}^{MN-1}$ by

$$I^*(i) = I(i) \boxplus V(i), \quad (3)$$

where $a \boxplus b = (a + b) \bmod 2^{n_0}$ and n_0 is the binary length of a and b .

- *Step 2*: Perform XOR operation on I^* to obtain sequence $I^{**} = \{I^{**}(i)\}_{i=0}^{MN-1}$ by

$$I^{**}(i) = W(i) \oplus I^*(i). \quad (4)$$

- *Step 3*: Divide I^{**} into two 4-bit sequences $L^{**} = \{L^{**}(i)\}_{i=0}^{MN-1}$, $H^{**} = \{H^{**}(i)\}_{i=0}^{MN-1}$ with function $\text{Spl}(X)$ that returns two sequences $X_L = \{X(i) \bmod 16\}_{i=0}^{MN-1}$ and $X_H = \{\lfloor X(i)/16 \rfloor\}_{i=0}^{MN-1}$, where $X = \{X(i)\}_{i=0}^{MN-1}$.

- *Step 4*: Perform bit-level permutation on L^{**} and H^{**} to get sequences $\tilde{L}^{**} = \{\tilde{L}^{**}(i)\}_{i=0}^{MN-1}$ and $\tilde{H}^{**} = \{\tilde{H}^{**}(i)\}_{i=0}^{MN-1}$: $\tilde{L}^{**}(i) = \sum_{k=0}^3 L_k^{**}(T_{1,k}(i)) \cdot 2^k$ and $\tilde{H}^{**}(i) = \sum_{k=0}^3 H_k^{**}(T_{2,k}(i)) \cdot 2^k$. Then produce sequence L' by

$$L'(i) = U(i) \boxplus \tilde{L}^{**}(i) \oplus \tilde{H}^{**}(i). \quad (5)$$

- *Step 5*: Perform bit-level permutation on L' and H^{**} to get sequences $\tilde{L}' = \{\tilde{L}'(i)\}_{i=0}^{MN-1}$ and $\tilde{H}^{**} =$

$\{\widehat{H}^{**}(i)\}_{i=0}^{MN-1}$; $\widehat{L}'(i) = \sum_{k=0}^3 L'_k(T_{3,k}(i)) \cdot 2^k$ and $\widehat{H}^{**}(i) = \sum_{k=0}^3 H_k^{**}(T_{4,k}(i)) \cdot 2^k$, where $L'(i) = \sum_{k=0}^3 L'_k(i) \cdot 2^k$. Then produce sequence $\mathbf{H}' = \{H'(i)\}_{i=0}^{MN-1}$ via

$$H'(i) = U'(i) \oplus \widehat{L}'(i) \oplus \widehat{H}^{**}(i). \quad (6)$$

– *Step 6:* Combine two 4-bit intermediate sequences \mathbf{L}' and \mathbf{H}' into one 8-bit sequence $\mathbf{I}' = \{I'(i)\}_{i=0}^{MN-1} = \{L'(i) + H'(i) \cdot 16\}_{i=0}^{MN-1}$. Then perform further confusion on \mathbf{I}' to obtain cipher-image \mathbf{I}'' , where

$$I''(i) = W'(i) \oplus (I'(i) \boxplus V'(i)). \quad (7)$$

3. Cryptanalysis of IEALM

The designers of IEALM emphasized that it can “well resist the chosen-plaintext attack and other classical attacks. Each image corresponds to a different secret key, so it has high security and can resist plaintext attacks” [18]. In this section, we dispute such claim based on the intrinsic properties of 2D-LCLM and IEALM.

3.1. The properties of 2D-LCLM

IEALM uses 2D-LCLM as a pseudo-random number generator to control the encryption process. According to Property 1 and 2, one can see that four equations on the variables of IEALM, $T_{1,0} = T_{1,1}$, $T_{2,0} = T_{2,1}$, $T_{3,0} = T_{3,1}$ and $T_{4,0} = T_{4,1}$ always hold, which make the algorithm is more vulnerable to chosen-plaintext attack.

Property 1. *The coordinates of 2D-LCLM, $x(i)$, $y(i)$, satisfy $x(i)/y(i) = x(j)/y(j)$ for any $i, j \in \{0, 1, \dots, MN - 1\}$.*

Proof. If $i = j$, it is obvious that $x(i)/y(i) = x(j)/y(j)$. Assuming $i > j$, one has

$$\begin{aligned} x(i) &= bx(i-1)(1-z(i)) \\ &= b^{i-j-1}x(j)(1-z(i)) \cdots (1-z(j+1)) \\ &= b^{i-j-1}x(j) \prod_{t=j+1}^i (1-z(t)). \end{aligned} \quad (8)$$

Similarly, one can get

$$y(i) = b^{i-j-1}y(j) \prod_{t=j+1}^i (1-z(t)).$$

Dividing the two sides of Eq. (8) with that of the above equation, one can get $x(i)/y(i) = x(j)/y(j)$. \square

Property 2. *If the coordinates of $\mathbf{X} = \{x(i)\}_{i=0}^N$ and $\mathbf{Y} = \{y(i)\}_{i=0}^N$ satisfy $x(i)/y(i) = x(j)/y(j)$, where $i, j \in \{0, 1, \dots, N\}$. Then the ranking order of $x(i)$ in \mathbf{X} is the same as that of $y(i)$ in \mathbf{Y} .*

Proof. According to property 1, it can be known that

$$y(j) = A(j)y(i),$$

where $A(j) = x(j)/x(i)$ and $j = 0, 1, \dots, N$. Let $\mathbf{A} = \{A(j)\}_{j=0}^N$, and then according to the known condition, one can know there are $k-1$ elements in \mathbf{A} that are less than 1. Then one can deduce that there are $k-1$ elements in \mathbf{Y} that are smaller than $y(i)$. Namely, $y(i)$ is ranked k in \mathbf{Y} . \square

Property 3. *2D-LCLM can be represented as*

$$\begin{cases} x(i+1) = bx(i)(1 - (a + (x(0)/y(0))^2)x(i-1)^2), \\ y(i+1) = by(i)(1 - (a(y(0)/x(0))^2 + 1)y(i-1)^2), \\ z(i+1) = b^2z(i)(1 - z(i-1))^2, \end{cases} \quad (9)$$

where $i \geq 1$, $(x(0), y(0), z(0))$ is the initial condition, $x(1) = bx(0)(1 - z(0))$, $y(1) = by(0)(1 - z(0))$, and $z(1) = ax(0)^2 + y(0)^2$.

Proof. According to Eq. (1) and Property 1, when $i \geq 1$, one has

$$\begin{aligned} x(i+1) &= bx(i)(1 - ax(i-1)^2 - y(i-1)^2) \\ &= bx(i)(1 - (a + (y(0)/x(0))^2)x(i-1)^2). \end{aligned}$$

Likewise, one can get

$$y(i+1) = by(i)(1 - (a(x(0)/y(0))^2 + 1)y(i-1)^2).$$

According to Eq. (1), when $i \geq 1$, one can obtain

$$\begin{aligned} z(i+1) &= ax(i)^2 + y(i)^2 \\ &= b^2(ax(i-1)^2 + y(i-1)^2)(1 - z(i-1))^2 \\ &= b^2z(i)(1 - z(i-1))^2. \end{aligned}$$

\square

The designers of IEALM stated “2D-LCLM is 3D in a real field” in [18], saying that it can show better properties than the other 2D chaotic maps. Unfortunately, Property 3 negates the statement, as it shows that three maps of 2D-LCLM are independent, and each independent map can not traverse the whole 3D domain.

To disclose the real structure of 2D-LCLM in a computer, we performed comprehensive tests on its functional graphs as [9, 10, 17]. Three typical examples are depicted in Fig. 1. Any orbit definitely enter a cycle after a transient process. Note that this rule always keep no matter how large the implementation precision is. As shown in [9, 17], some cycles of short period (even self-loop) always exist no matter which enhancement method is adopted, e.g.

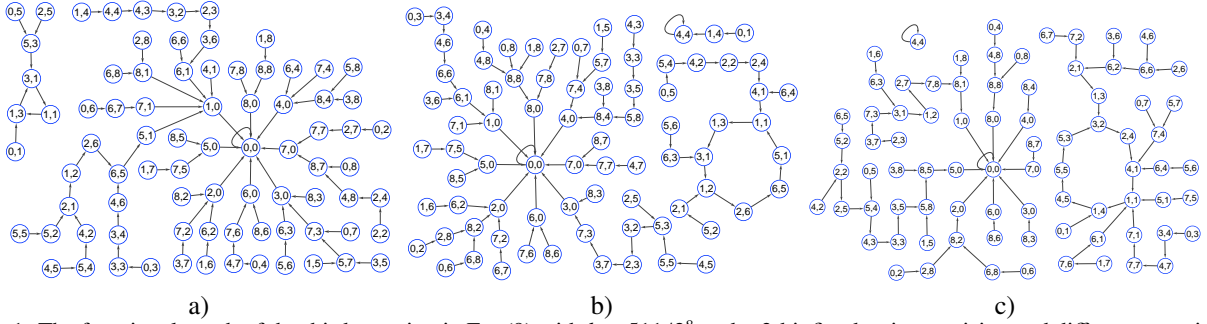


Figure 1: The functional graph of the third equation in Eq. (9) with $b = 511/2^8$ under 3-bit fixed-point precision and different quantization strategies: a) floor; b) round; c) ceil, where symbol (i, j) in each node denotes coordinate $(i/2^3, j/2^3)$.

increasing the arithmetic precision, perturbing states, perturbing the control parameters, switching among multiple chaotic maps and cascading more than one chaotic maps. If the initial state is located in a small-scale connected component or a cycle of short period in the functional graph of the used chaotic map, there are even no enough available states to be discarded. So, an adaptive threshold should be set to avoid this problem. But, it would cost additional computation.

3.2. Key distribution

To thwart known/chosen-plaintext attack, IEALM uses the information of plain-image to generate different secret keys when encrypting different plain-images, but the uniformity of key distribution is ignored. For an encryption algorithm, it is secure if the keys obey a uniform distribution, because all keys have the same probability of being selected. A classical example of insecurity is substitution cipher (e.g. Caesar cipher), which uses a codebook to directly replace one letter with another. Since the frequency of letters does not obey a uniform distribution, the algorithm cannot resist statistical attack.

In IEALM, the sum of pixel value of the plain-image is used to generate the initial value of 2D-LCLM. Since a natural image has its practical visual meaning, the sum is not random and does not obey a uniform distribution. To show such property of natural images, we tested the distribution of the average pixel value of 60,000 images of Mini-ImageNet [19] and found that the value follows a normal distribution, as shown in Fig. 2.

According to the above experimental conclusion, if we arbitrarily select a 256×256 image from Mini-ImageNet, in theory, a total of $4.7 \cdot 10^{21}$ possible keys can be generated. However, since the mean pixel values are normally distributed, the probabilities of these keys being selected are not equal. According to the property of the normal distribution, one can get $\text{Prob}(81.641 < x_r < 159.609) = 0.6827$, $\text{Prob}(77.388 < y_g < 151.382) = 0.6827$ and $\text{Prob}(60.422 <$

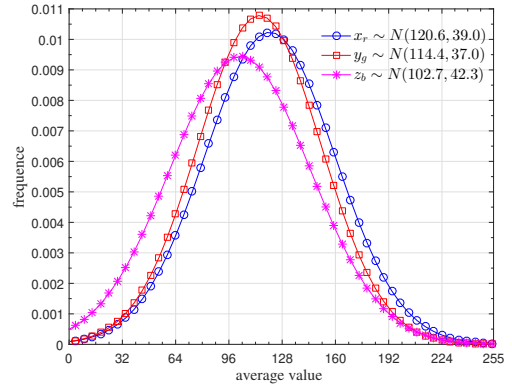


Figure 2: The distribution of the average pixel value of images of Mini-ImageNet, where x_r , y_g , and z_b are the average pixel values of red, green, and blue channels of images, respectively.

$z_b < 144.984) = 0.6827$. As for an attacker, only by enumerating this part of the secret key through a brute force attack, the success probability for attacking the algorithm is at least 30%. However, this part accounts for only 2.91% of the total key space.

3.3. Chosen-plaintext Attack

In [18], the designers adopted the sum of pixel values of each plain-image as the secret key, trying to achieve the function of “one-time passwords”. But it is also possible to generate the same key for different plain-images. Firstly, the same key must be generated for two identical plain-images. Secondly, for different plain-images it may also generate the same key, such as different images with the same sum of pixel values. In addition, due to the limited arithmetic precision of computer, even different plain-images owning different average pixel values may generate the same secret key through the calculation of Eq. (2).

As for plain-image I_0 and its corresponding cipher-image I'_0 , one can try to decrypt another cipher-image I''_1 if the following two conditions hold at the same time: 1) I''_1

is encrypted with the same secret key as I'_0 ; 2) the corresponding plain-image of I'_1 has the same result calculated with Eq. (2) as I_0 , namely the final sequences controlling the encryption process are the same.

- Simplification of IEALM

For simplicity, let $(W_L, W_H) = \text{Spl}(W)$, $(W'_L, W'_H) = \text{Spl}(W')$, $(V_L, V_H) = \text{Spl}(V)$, $(V'_L, V'_H) = \text{Spl}(V')$, and $(L^*, H^*) = \text{Spl}(I^*)$. From Eq. (5), as for the four lower bit-planes, one has

$$\begin{aligned} L'(i) &= U(i) \oplus \widetilde{W}_L(i) \oplus \widetilde{W}_H(i) \oplus \widetilde{L}^*(i) \oplus \widetilde{H}^*(i) \\ &= \beta(i) \oplus L^*(i), \end{aligned} \quad (10)$$

where

$$L^*(i) = \widetilde{L}^*(i) \oplus \widetilde{H}^*(i),$$

$\beta(i) = U(i) \oplus \widetilde{W}_L(i) \oplus \widetilde{W}_H(i)$, $\widetilde{W}_L(i) = \sum_{k=0}^3 W_{L,k}(T_{1,k}(i)) \cdot 2^k$, $\widetilde{W}_H(i) = \sum_{k=0}^3 W_{H,k}(T_{2,k}(i)) \cdot 2^k$, $\widetilde{L}^*(i) = \sum_{k=0}^3 L_k^*(T_{1,k}(i)) \cdot 2^k$, and $\widetilde{H}^*(i) = \sum_{k=0}^3 H_k^*(T_{2,k}(i)) \cdot 2^k$. According to Eq. (6), for the four higher bit-planes, one has

$$\begin{aligned} H'(i) &= U'(i) \oplus \widehat{L}^*(i) \oplus \widehat{W}_H(i) \oplus \widehat{H}^*(i) \\ &= U'(i) \oplus \widehat{\beta}(i) \oplus \widehat{L}^*(i) \oplus \widehat{W}_H(i) \oplus \widehat{H}^*(i) \\ &= \beta'(i) \oplus H^*(i), \end{aligned} \quad (11)$$

where

$$H^*(i) = \widehat{L}^*(i) \oplus \widehat{H}^*(i),$$

$\beta'(i) = U'(i) \oplus \widehat{\beta}(i) \oplus \widehat{W}_H(i)$, $\widehat{\beta}(i) = \sum_{k=0}^3 \beta_k(T_{3,k}(i)) \cdot 2^k$, $\widehat{W}_H(i) = \sum_{k=0}^3 W_{H,k}(T_{4,k}(i)) \cdot 2^k$, $\widehat{L}^*(i) = \sum_{k=0}^3 L_k^*(T_{3,k}(i)) \cdot 2^k$, and $\widehat{H}^*(i) = \sum_{k=0}^3 H_k^*(T_{4,k}(i)) \cdot 2^k$.

Fact 1. $\forall a, b \in \{0, 1, \dots, 2^{n_0} - 1\}$, $(a \oplus 2^{n_0-1}) \boxplus b = (a \boxplus b) \oplus 2^{n_0-1} = a \oplus (b \boxplus 2^{n_0-1})$.

According to Eq. (3), Eq. (4) and Fact 1, one can get

$$\begin{aligned} I^{**}(i) &= W(i) \oplus (I(i) \boxplus V(i)) \\ &= (W(i) \oplus 128) \oplus (I(i) \boxplus (V(i) \oplus 128)), \end{aligned}$$

where $i = 0, 1, \dots, MN-1$. This means that one can shift the MSB (most significant bit) of $V(i)$ to $W(i)$ without affecting the encryption result.

- Differential attack on IEALM

Given two plain-images I_0 and I_1 encrypted by IEALM with the same controlling sequences, we define the bit-wise XOR operation between them as $I_{0\oplus 1} = I_0 \oplus I_1$. Likewise, $L_{0\oplus 1} = L_0 \oplus L_1$ and $H_{0\oplus 1} = H_0 \oplus H_1$, where $(L_0, H_0) = \text{Spl}(I_0)$ and $(L_1, H_1) = \text{Spl}(I_1)$. As for the corresponding cipher-images I'_0 and I'_1 , one can get $(L''_0, H''_0) = \text{Spl}(I'_0)$ and $(L''_1, H''_1) = \text{Spl}(I'_1)$.

According to Eq. (3), one can obtain

$$\begin{cases} L_{0\oplus 1}^* = (L_0 \boxplus V_L) \oplus (L_1 \boxplus V_L), \\ H_{0\oplus 1}^* = (H_0 \boxplus V_H \boxplus r_0) \oplus (H_1 \boxplus V_H \boxplus r_1), \end{cases} \quad (12)$$

where $r_j = \lfloor (L'_j + V_L)/16 \rfloor$, $j = 0, 1$. Analyzing the above equation individually on each bit-plane, one can get

$$L_{0\oplus 1,k}^* = \begin{cases} L_{0\oplus 1,0}(i) & \text{when } k = 0; \\ L_{0\oplus 1,k}(i) \oplus \theta_{0\oplus 1,k}(i) & \text{when } k = 1, 2, 3, \end{cases} \quad (13)$$

and

$$H_{0\oplus 1,k}^* = \begin{cases} H_{0\oplus 1,0}(i) \oplus r_{0\oplus 1}(i) & \text{when } k = 0; \\ H_{0\oplus 1,k}(i) \oplus \lambda_{0\oplus 1,k}(i) & \text{when } k = 1, 2, 3, \end{cases} \quad (14)$$

where

$$\begin{cases} \theta_{j,k}(i) = \left\lfloor \frac{\sum_{t=0}^{k-1} (V_{L,t}(i) + L_{j,t}(i)) \cdot 2^t}{2^k} \right\rfloor, \\ \lambda_{j,k}(i) = \left\lfloor \frac{\sum_{t=0}^{k-1} (V_{H,t}(i) + H_{j,t}(i)) \cdot 2^t + r_j(i)}{2^k} \right\rfloor, \end{cases} \quad (15)$$

and $j = 0, 1$.

According to Eq. (10), by differentiating plain-images, one has

$$\begin{aligned} L'_{0\oplus 1}(i) &= L_{0\oplus 1}^*(i) \\ &= \widetilde{L}_{0\oplus 1}^*(i) \oplus \widetilde{H}_{0\oplus 1}^*(i). \end{aligned}$$

Then in the k -bit-plane, one can get

$$\begin{aligned} L'_{0\oplus 1,k}(i) &= L_{0\oplus 1,k}^*(i) \\ &= L_{0\oplus 1,k}^*(T_{1,k}(i)) \oplus H_{0\oplus 1,k}^*(T_{2,k}(i)). \end{aligned} \quad (16)$$

Similarly, from Eq. (11) one can get

$$\begin{aligned} H'_{0\oplus 1,k}(i) &= H_{0\oplus 1,k}^*(i) \\ &= L_{0\oplus 1,k}^*(T_{3,k}(i)) \oplus H_{0\oplus 1,k}^*(T_{4,k}(i)) \end{aligned} \quad (17)$$

According to Eq. (7), one has

$$\begin{cases} L''_{0\oplus 1} = (L'_0 \boxplus V'_L) \oplus (L'_1 \boxplus V'_L), \\ H''_{0\oplus 1} = (H'_0 \boxplus V'_H \boxplus r'_0) \oplus (H'_1 \boxplus V'_H \boxplus r'_1), \end{cases}$$

where $r'_j = \lfloor (L'_j + V'_L)/16 \rfloor$ and $j = 0, 1$. Then one can get

$$L''_{0\oplus 1,k}(i) = \begin{cases} L'_{0\oplus 1,0}(i) & \text{when } k = 0; \\ L'_{0\oplus 1,k}(i) \oplus \theta'_{0\oplus 1,k}(i) & \text{when } k = 1, 2, 3, \end{cases} \quad (18)$$

and

$$H''_{0\oplus 1,k}(i) = \begin{cases} H'_{0\oplus 1,0}(i) \oplus r'_{0\oplus 1}(i) & \text{when } k = 0; \\ H'_{0\oplus 1,k}(i) \oplus \lambda'_{0\oplus 1,k}(i) & \text{when } k = 1, 2, 3, \end{cases} \quad (19)$$

where

$$\begin{cases} \theta'_{j,k}(i) = \left\lfloor \frac{\sum_{t=0}^{k-1} (V'_{L,t}(i) + L'_{j,t}(i)) \cdot 2^t}{2^k} \right\rfloor, \\ \lambda'_{j,k}(i) = \left\lfloor \frac{\sum_{t=0}^{k-1} (V'_{H,t}(i) + H'_{j,t}(i)) \cdot 2^t + r'_j(i)}{2^k} \right\rfloor, \end{cases} \quad (20)$$

and $j = 0, 1$.

- **Determining T_2**

The bit-level permutation using $T_{2,k}$ can be determined with Eq. (14), (16), and (18). When $k = 0$, one just needs to eliminate $L_{0\oplus 1}^*(i)$ and $r_{0\oplus 1}(i)$ to attack the permutation. Through setting $L_{0\oplus 1}(i) = 0$, one gets $r_{0\oplus 1}(i) = 0$ and $L_{0\oplus 1}^*(i) = 0$, and then Eq. (16) becomes $L'_{0\oplus 1,k}(i) = H_{0\oplus 1,k}^*(T_{2,k}(i))$.

When $k > 0$, one needs to additionally eliminate the effect of the carries from the lower bit-planes, namely $\lambda_{0\oplus 1,k}(i)$ and $\theta'_{0\oplus 1,k}(i)$. Hence, for $k' = 0 \sim (k-1)$, set $H_{0\oplus 1,k'}(i) = 0$, and then $\lambda_{0\oplus 1,k'}(i) = 0$ according to Eq. (15). Furthermore, one can deduce $L'_{0\oplus 1,k'}(i) = 0$ and $\theta'_{0\oplus 1,k'}(i) = 0$ from Eq. (20). Based on these setting, for $k = 0 \sim 3$, Eq. (18) becomes

$$\begin{aligned} L''_{0\oplus 1,k}(i) &= L'_{0\oplus 1,k}(i) \\ &= H_{0\oplus 1,k}(T_{2,k}(i)). \end{aligned}$$

In this case, the encryption operation on $H_{0\oplus 1,k}(i)$ is actually permutation-only. As the chosen-plaintext attack on permutation-only encryption algorithm is already mature [20], we only briefly describe the attack procedure. The permutation vector $T_{2,k}$ can be exactly recovered by the following steps:

- *Step 1:* Choose a plain-image I_0 of fixed value zero and get its corresponding cipher-image I''_0 .
- *Step 2:* Choose n plain-images I_1, I_2, \dots, I_n that satisfy $I_t = \{H_t(i) \cdot 16\}_{i=0}^{MN-1}$, $t = 1, 2, \dots, n$, and the k' -th significant bit of $H_t(i)$ is

$$H_{t,k'}(i) = \begin{cases} \lfloor i/2^{t-1} \rfloor \bmod 2 & \text{if } k' = k; \\ 0 & \text{otherwise,} \end{cases}$$

where $n = \lceil \log_2(MN) \rceil$ and $k' = 0 \sim 3$.

- *Step 3:* Encrypt I_1, I_2, \dots, I_n and get the corresponding cipher-images $I''_1, I''_2, \dots, I''_n$. Then, for $\sum_{t=1}^n L''_{0\oplus 1,k}(i') \cdot 2^{t-1} = i$, one can confirm $T_{2,k}(i') = i$, where $L''_t(i') = I''_t(i') \bmod 16$.

Perform the above procedures for $k = 0 \sim 3$ to recover $T_{2,0}, T_{2,1}, T_{2,2}$, and $T_{2,3}$. In fact, since $T_{2,1} = T_{2,0}$, one can recover T_2 with only $\lceil \log_2(MN) \rceil + 1$ plain-images.

- **Determining V_L**

Let $H_{0\oplus 1}(i) = 0$, $L_0(i) = 0$ and $L_1(i) = c$ for $i = 0, 1, \dots, MN-1$, where $1 \leq c \leq 15$. For $k = 0$, Eq. (18) can be expressed as

$$\begin{aligned} L''_{0\oplus 1,0}(i) &= L'_{0\oplus 1,0}(i) \\ &= L_{0\oplus 1,0}^*(T_{1,0}(i)) \oplus H_{0\oplus 1,0}^*(T_{2,0}(i)) \\ &= L_{0\oplus 1,0}(T_{1,0}(i)) \oplus r_{0\oplus 1}(T_{2,0}(i)) \\ &= (c \bmod 2) \oplus r_1(T_{2,0}(i)). \end{aligned}$$

As $r_1(T_{2,0}(i)) = \lfloor (c + V_L(T_{2,0}(i))) / 16 \rfloor$, $r_1(T_{2,0}(i)) = 1$ if and only if $c + V_L(T_{2,0}(i)) \geq 16$. Therefore, one can obtain

$$L''_{0\oplus 1,0}(i) = \begin{cases} (c \bmod 2) \oplus 1 & \text{if } (c + V_L(T_{2,0}(i))) \geq 16; \\ c \bmod 2 & \text{otherwise.} \end{cases}$$

Setting $c = 1$, one can deduce that

$$V_L(T_{2,0}(i)) \in \begin{cases} \{15\} & \text{if } L''_{0\oplus 1,0}(i) = 0; \\ \{0, 1, \dots, 14\} & \text{otherwise.} \end{cases}$$

Namely, one can determine $V_L(T_{2,0}(i)) = 15$ by finding $L''_{0\oplus 1,0}(i) = 0$. Similarly, one can set $c = 2$ and then find $V_L(T_{2,0}(i)) = 14$ through

$$V_L(T_{2,0}(i)) \in \begin{cases} \{14, 15\} & \text{if } L''_{0\oplus 1,0}(i) = 1; \\ \{0, 1, \dots, 13\} & \text{otherwise.} \end{cases}$$

The remaining unknown elements of V_L can be obtained by choosing $c \in \{3, 4, \dots, 15\}$ in turn.

- **Determining T_1**

The permutation vector $T_{1,k}$ can be recovered using Eq. (13), (16), and (18). To eliminate $H_{0\oplus 1,k}^*(T_{2,k}(i))$ in Eq. (16), one can set $H_0(i) = r_1(i)$ and $H_1(i) = r_0(i)$ to get $H_{0\oplus 1}^*(i) = 0$. When $k > 0$, choosing $L_{0\oplus 1,k'}(i) = 0$ for $k' = 0 \sim (k-1)$, one can deduce $\theta_{0\oplus 1,k}(i) = 0$ from Eq. (15). Furthermore, because $L'_{0\oplus 1,k'}(i) = L_{0\oplus 1,k'}^*(T_{1,k'}(i)) = L_{0\oplus 1,k'}(i) = 0$, the carry $\theta'_{0\oplus 1,k}(i) = 0$ according to Eq. (20). Then, Eq. (18) becomes

$$\begin{aligned} L''_{0\oplus 1,k}(i) &= L'_{0\oplus 1,k}(i) \\ &= L_{0\oplus 1,k}^*(T_{1,k}(i)) \\ &= L_{0\oplus 1,k}(T_{1,k}(i)). \end{aligned}$$

Therefore, T_1 can be exactly determined by utilizing the recovery procedure of T_2 .

- Determining V_H

The three LSB of V_H can be guessed by observing the corresponding carry incurred by the addition in Eq. (12). Setting $I_0(i) = 0$, one has $L_0^*(i) = V_L(i)$ and $r_0(i) = 0$. Since V_L is known, one can directly choose $L_1^*(i) = L_0^*(i) \oplus 2^k$ and $H_1(i) = 2^k \boxplus r_1(i)$ to determine $V_{H,k}$, where $k < 3$ and $a \boxplus b = (a + b) \bmod 2^{n_0}$. Note that $r_1(i)$ is also known and $L_1(i) = L_1^*(i) \boxplus V_L(i)$. Then Eq. (12) becomes

$$H_{0\oplus 1}^*(i) = V_H(i) \oplus (2^k \boxplus V_H(i)).$$

Then one can deduce $L_{0\oplus 1,k}^*(i) = H_{0\oplus 1,k}^*(i) = 1$. When $k > 0$, for lower bit-planes, one has $L_{0\oplus 1,k'}^*(i) = H_{0\oplus 1,k'}^*(i) = 0$, where $k' = 0 \sim (k-1)$. From Eq. (16), one can get $L'_{0\oplus 1,k'}(i) = 0$ and $L'_{0\oplus 1,k}(i) = 0$. Furthermore, the carry $\theta'_{0\oplus 1,k+1}(i) = 0$ according to Eq. (20). Hence, Eq. (18) becomes

$$\begin{aligned} L''_{0\oplus 1,k+1}(i) &= L'_{0\oplus 1,k+1}(i) \\ &= L_{0\oplus 1,k+1}^*(T_{1,k+1}(i)) \oplus H_{0\oplus 1,k+1}^*(T_{2,k+1}(i)). \end{aligned}$$

Considering $L_{0\oplus 1,k+1}^*(T_{1,k+1}(i)) = 0$, $H_{0\oplus 1,k+1}(T_{2,k+1}(i)) = 0$, and $\lambda_{0,k+1}(T_{2,k+1}(i)) = 0$, the above equation can further become

$$L''_{0\oplus 1,k+1}(i) = \lambda_{1,k+1}(T_{2,k+1}(i)).$$

Incorporating $\sum_{t=0}^k H_{1,t} \cdot 2^t + r_1 = 2^k$ into Eq. (15), one can get

$$\begin{aligned} \lambda_{1,k+1}(i) &= \left\lfloor \frac{\sum_{t=0}^k V_{H,t}(i) \cdot 2^t + 2^k}{2^{k+1}} \right\rfloor \\ &= \begin{cases} 0 & \text{when } V_{H,k}(i) = 0; \\ 1 & \text{when } V_{H,k}(i) = 1, \end{cases} \end{aligned}$$

Namely, $\lambda_{1,k+1}(i) = V_{H,k}(i)$. Combining the above two equations, one has

$$V_{H,k}(T_{2,k+1}(i)) = L''_{0\oplus 1,k+1}(i).$$

Then one can recover $V_{H,0}(i)$, $V_{H,1}(i)$, $V_{H,2}(i)$ by setting $k = 0, 1, 2$ in turn. Shifting $V_{H,3}(i)$ to $W_{H,3}(i)$ directly, one can set $V_{H,3}(i) = 0$ according to Fact 1.

Since the equivalent version of V is known, one can convert any I^* to I via $I = I^* \boxplus V$. Therefore, we only consider choosing I^* without specifying the plain-image I in subsequent discussion.

- Determining T_4

The permutation vector $T_{4,k}$ can be determined by Eq. (17) and (19). Set $I_0^*(i) = 0$ and $\widetilde{L}_1^*(i) = \widetilde{H}_1^*(i)$ to get $L'_{0\oplus 1}(i) = L_{0\oplus 1}^*(i) = 0$ and $r'_0(i) = r'_1(i)$. From Eq. (17), one can know that $H'_{0\oplus 1,k}(i) = H_{0\oplus 1,k}^*(T_{4,k}(i))$. When $k > 0$, set $H_{1,k'}^*(i) = 0$ for $k' = 0 \sim (k-1)$ to eliminate the effect of carry $\lambda'_{0\oplus 1,k}(i)$. Then one can get $H'_{0\oplus 1,k'}(i) = 0$ and then $\lambda'_{0\oplus 1,k}(i) = 0$ from Eq. (20). Hence, for $k = 0 \sim 3$, Eq. (19) becomes

$$\begin{aligned} H''_{0\oplus 1,k}(i) &= H'_{0\oplus 1,k}(i) \\ &= H_{0\oplus 1,k}^*(T_{4,k}(i)). \end{aligned}$$

Likewise, one can recover T_4 exactly by utilizing the recovery procedure of T_2 .

- Determining T_3

Let $L_0^*(i) = H_0^*(i) = H_1^*(i) = 0$ for $i = 0, 1, \dots, (MN-1)$. From $L^*(i) = \widetilde{L}^*(i) \oplus \widetilde{H}^*(i)$ and $H^*(i) = \widetilde{L}^*(i) \oplus \widetilde{H}^*(i)$, one can get $L_0^*(i) = 0$, $L_1^*(i) = \widetilde{L}_1^*(i)$, $H_0^*(i) = 0$, and $H_{1,k}^*(i) = L_{1,k}^*(T_{3,k}(i))$. Since $L_1^*(i) = \widetilde{L}_1^*(i)$ and T_1 is known, we directly choose L_1^* without specifying the corresponding plain-image in the succeeding discussion.

To determine $T_{3,0}$, according to Eq. (17) and (19), one has

$$\begin{aligned} H''_{0\oplus 1,0}(i) &= H'_{0\oplus 1,0}(i) \oplus r'_{0\oplus 1}(i) \\ &= H_{0\oplus 1,0}^*(i) \oplus r'_{0\oplus 1}(i) \\ &= L_{1,0}^*(T_{3,0}(i)) \oplus r'_{0\oplus 1}(i), \end{aligned} \quad (21)$$

where $r'_{0\oplus 1}(i) = \lfloor (\beta(i) + V'_L(i))/16 \rfloor \oplus \lfloor ((\beta(i) \oplus L_1^*(i)) + V'_L(i))/16 \rfloor$. Obviously, $r'_{0\oplus 1}(i)$ is only dependent on $L_1^*(i)$, so we use $\Phi(i, L_1^*(i))$ to represent $r'_{0\oplus 1}(i)$ corresponding to different $L_1^*(i)$. Then, Eq. (21) can be represented as

$$H''_{0\oplus 1,0}(i) \oplus \Phi(i, L_1^*(i)) = L_{1,0}^*(T_{3,0}(i)).$$

For $L_1^*(i) = 0$, one can see $\Phi(i, 0) = 0$. To get $\Phi(i, 1)$, one can set $L_1^*(i) = 1$ for $i = 0 \sim (MN-1)$ and then $\Phi(i, 1) = H''_{0\oplus 1,0}(i) \oplus 1$ from the above equation. Choosing $L_1^*(i) \in \{0, 1\}$, since $\Phi(i, L_1^*(i))$ is also known, one can recover $T_{3,0}$ with the preceding equation referring to the recovery process of $T_{2,k}$.

To determine $T_{3,k}$ for $k \in \{1, 2, 3\}$, likewise, one has

$$\begin{aligned} H''_{0\oplus 1,k}(i) &= H'_{0\oplus 1,k}(i) \oplus \lambda'_{0\oplus 1,k}(i) \\ &= L_{1,k}^*(T_{3,k}(i)) \oplus \lambda'_{0\oplus 1,k}(i). \end{aligned} \quad (22)$$

From Eq. (11), one can get $H'_0(i) = \beta'(i)$. Choosing $L_{1,t}^*(i) \in \{0, 2^k\}$, for $t = 0 \sim (k-1)$, $H_{1,t}^*(i) = L_{1,t}^*(T_{3,t}(i)) =$

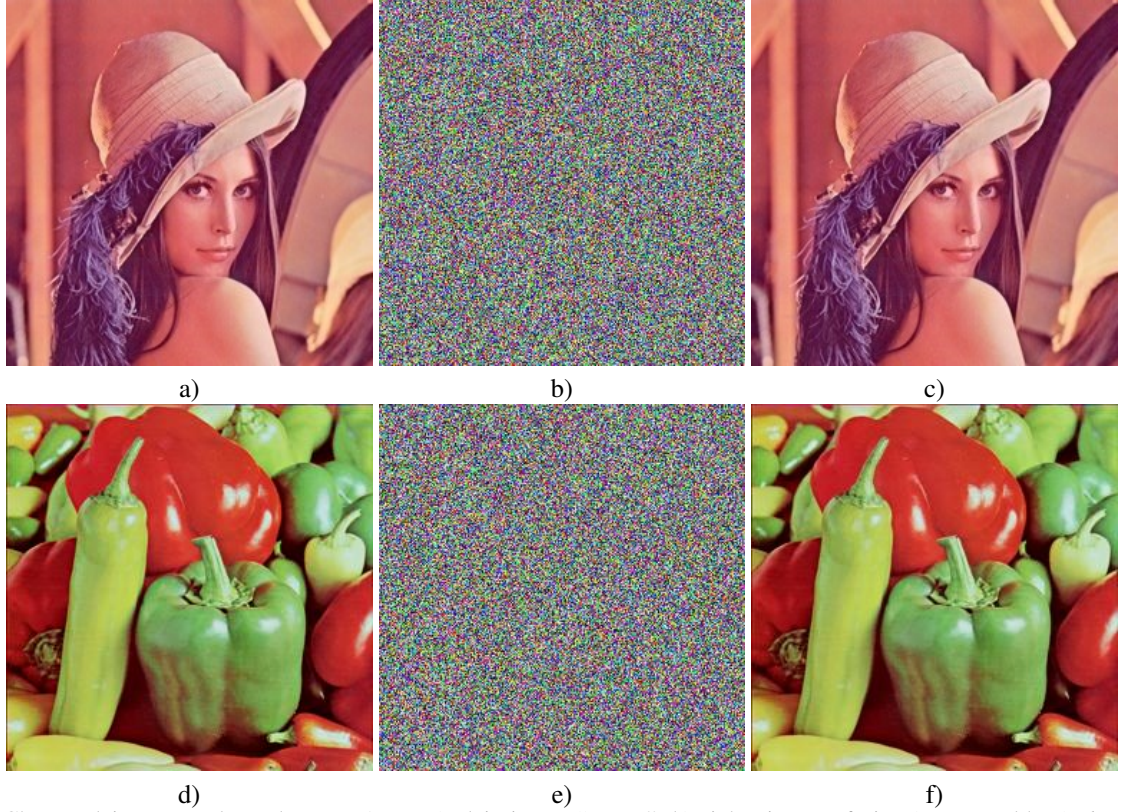


Figure 3: Chosen-plaintext attack results on IEALM: a) plain-image ‘‘Lenna’’; b) cipher-image of Fig. a) encrypted by $(a, b, x_r, y_g, z_b) = (2, 1.99, 29676, 9202, 62299)$; c) the recovered plain-image with the obtained equivalent secret key; d) plain-image ‘‘Peppers’’; e) cipher-image of Fig. d) encrypted by $(a, b, x_r, y_g, z_b) = (2, 1.99, 29232, 54749, 57603)$; f) the recovered plain-image with the obtained equivalent secret key.

0 and then $H'_{1,t}(i) = \beta'_t(i)$. According to Eq. (20), one has

$$\begin{aligned} \lambda'_{0\oplus 1,k}(i) &= \left[\frac{\alpha_k(i) + r'_0(i)}{2^k} \right] \oplus \left[\frac{\alpha_k(i) + r'_1(i)}{2^k} \right] \\ &= \left[\frac{\alpha_k(i) + \lfloor (\beta(i) + V'_L(i))/16 \rfloor}{2^k} \right] \\ &\oplus \left[\frac{\alpha_k(i) + \lfloor (\beta(i) \oplus L_1^*(i) + V'_L(i))/16 \rfloor}{2^k} \right], \end{aligned}$$

where $\alpha_k(i) = \sum_{t=0}^{k-1} (V'_{H,t}(i) + \beta'_t(i)) \cdot 2^t$. Likewise, we use $\Psi(i, L_1^*(i))$ to represent $\lambda'_{0\oplus 1,k}(i)$ corresponding to different $L_1^*(i)$. Then Eq. (22) can be expressed as

$$H''_{0\oplus 1,k}(i) = L_{1,k}^*(T_{3,k}(i)) \oplus \Psi(i, L_1^*(i)).$$

For $L_1^*(i) = 0$, one can know $\Psi(i, 0) = 0$. To obtain $\Psi(i, 2^k)$, set $L_1^*(i) = 2^k$ for $i = 0 \sim (MN - 1)$ and then get $\Psi(i, L_1^*(i)) = H''_{0\oplus 1,k}(i) \oplus 1$ from the preceding equation. Choosing $L_1^*(i) \in \{0, 2^k\}$, since $\Psi(i, L_1^*(i))$ is known, one can recover $T_{3,k}$ using the above equation and the recovery process of $T_{2,k}$. Therefore, $T_3 = \{T_{3,0}, T_{3,1}, T_{3,2}, T_{3,3}\}$ is determined.

- Attacking the other encryption operations

Once all permutation vectors were successfully recovered, one can get

$$\begin{aligned} I''(i) &= W'(i) \oplus (I'(i) \boxplus V'(i)) \\ &= W'(i) \oplus (((\beta(i) + 16 \cdot \beta'(i)) \oplus I^*(i)) \boxplus V'(i)) \end{aligned}$$

from Eq. (7), (10) and (11), where $I^*(i) = L^*(i) + H^*(i) \cdot 16$. In the above equation, $I''(i)$ can be determined by $I^*(i)$ and vice versa. In other words, $I''(i)$ and $I^*(i)$ are one-to-one correspondence. Note that one can directly choose any I^* and then obtain its plain-image I . Accordingly, choose $I^*(i) = c$ for $c = 0 \sim 255$, and then get $I''(i)$ to obtain a map $F(i, I''(i)) = c$, which can be used to recover $I^*(i)$ from $I''(i)$.

- Recovering the plain-image

As for a cipher-image I'' , first recover I^* by $I^*(i) = F(i, I''(i))$ and calculate $(L^*, H^*) = \text{Spl}(I^*)$. Perform the permutation with T_3 on L^* to get \tilde{L}^* , and derive \tilde{H}^* by $\tilde{H}^*(i) = H^*(i) \oplus \tilde{L}^*(i)$. Apply the inverse version of

Table 1: Some data in the encryption process of the blue channel of Fig. 3b).

Index Item	1	2	4	8	16	32	64	128	256	512	1024
$I(i)$	125	123	122	114	110	83	100	106	125	122	114
$T^{2,0}(i)$	63654	41166	44389	5418	60541	8324	8394	52758	10693	18236	12940
$T^{1,1}(i)$	62246	12618	22576	424	5892	47186	18568	14185	4948	47571	6740
$T^{4,2}(i)$	8436	37177	13122	24285	25840	24911	350	52730	12436	30075	132
$T^{3,3}(i)$	1357	27981	60186	16982	691	9877	32352	30284	62723	61986	27694
$V(i)$	72	61	201	128	210	239	54	92	42	22	199
$I^*(i)$	197	184	67	242	64	66	154	198	167	144	57
$I'(i)$	241	191	78	14	227	4	172	169	53	31	252
$I''(i)$	207	70	75	7	173	226	225	87	5	190	196

Table 2: The items obtain in an attack corresponding to the data shown in Table 1.

Index Item	1	2	4	8	16	32	64	128	256	512	1024
$T^{2,0}(i)$	63654	41166	44389	5418	60541	8324	8394	52758	10693	18236	12940
$T^{1,1}(i)$	62246	12618	22576	424	5892	47186	18568	14185	4948	47571	6740
$T^{4,2}(i)$	8436	37177	13122	24285	25840	24911	350	52730	12436	30075	132
$T^{3,3}(i)$	1357	27981	60186	16982	691	9877	32352	30284	62723	61986	27694
$V(i)$	72	61	<u>73</u>	<u>0</u>	<u>82</u>	<u>111</u>	54	92	42	22	<u>71</u>
$I^*(i)$	197	184	<u>195</u>	<u>114</u>	<u>192</u>	<u>194</u>	154	198	167	144	<u>185</u>
$I(i)$	125	123	<u>122</u>	<u>114</u>	<u>110</u>	<u>83</u>	100	106	125	122	<u>114</u>

permutation with T_4 on \widehat{H}^* to recover H^* . Permute H^* with T_2 to get \widetilde{H}^* , and derive \widetilde{L}^* by $\widetilde{L}^*(i) = L^*(i) \oplus \widetilde{H}^*(i)$. Then employ the inverse version of permutation with T_1 on \widetilde{L}^* to recover L^* . Finally, one can get plain-image I by $I(i) = (L^*(i) + 16 \cdot H^*(i)) \boxminus V(i)$.

To verify the effectiveness of the above attacking process, we performed a number of experiments with some random secret keys¹. The attack results on a typical image is shown in Fig. 3. The success rate of the attack is 100%. And some recovered data during the attack are listed in Tables 1 and 2 as a reference. Note that the recovered MSB of $V(i)$ and $I^*(i)$ may be different from that in Table 1 (the underlined elements) as it has no influence on the decryption result.

3.4. Complexity of the attack

Recalling the attack process, consider $T_{1,0} = T_{1,1}$, $T_{2,0} = T_{2,1}$, $T_{3,0} = T_{3,1}$ and $T_{4,0} = T_{4,1}$, one can see that the images required in procedure determining T_2 and T_4 are $P_1 = P_2 = \lceil \log_2(MN) \rceil + 1$. And the number of images needed to recover V is $P_3 = 6$. Likewise, one can get determining T_1 and T_3 need $P_4 = 2 \cdot \lceil \log_2(MN) \rceil$

¹The source codes of this paper are available at <https://github.com/ChengqingLi/MM-IEALM>

and $P_5 = \lceil \log_2(MN) \rceil + 1$ images, respectively. The final procedure of attack requires $P_6 = 86$ images. Since the complexity of each procedure is $O(P_i MN)$, the complexity of the attack is $O(PMN) \approx O(MN \log_2(MN))$, where $P = P_1 + P_2 + \dots + P_6$.

As for the image shown in Fig. 3 c), the numbers of chosen plain-images required for the above attacking procedures are 17, 17, 6, 32, 17, and 86, respectively. The complexity of the attack is $O(175MN)$, this attack takes only a few seconds on a personal computer.

3.5. Analysis of the key space

The designers of IEALM claimed that its key space is greater than 10^{120} [18]. In fact, the real key space of IEALM depends on arithmetic precision L and the pixel values of the plain-image. Although the pseudorandom number generator has six initial values, they are only determined by the pixel values of the plain-image. This makes the actual key space of IEALM can not reach that estimated by the designers. Strictly speaking, the key space of IEALM is only $2^{2L} \cdot (256MN)^3$. As for a plain-image of size 2048×2048 , the sizes of key space of IEALM implemented with precision 32-bit and 64-bit are 10^{46} and 10^{65} , respectively. In contrast, for a plain-image of size 256×256 , the sizes are reduced to 10^{38} and 10^{57} , respectively. In any case, the size of key space is far smaller than 10^{120} claimed by the designers. Note that the above estimation is only the

maximum theoretical value of the key space. As the keys are not uniformly distributed and not all control parameters of a chaotic map can generate sequences with enough randomness, the real effective key space is even much smaller.

4. Conclusion

This paper analyzed the security performance of an image encryption algorithm based on 2D lag-complex logistic map from the viewpoint of modern cryptanalysis. The sensitivity mechanism between the controlling pseudo-random sequences and the plain-image can be cancelled for some special plain-images. Due to the algorithm is composed by cascading some independent basic encryption parts, their equivalents were recovered with some chosen plain-images one by one. The properties of the underlying chaotic map were explored to enhance the attacking efficiency. Both theoretical analysis and experimental results were provided to show the effectiveness of the attack. The presented results show that designing a secure and efficient image encryption algorithm should consider many factors, such as special storage format of image data, randomness of the adopted pseudo-random number generator, application scenarios and real structure of the encryption algorithm.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 61772447 and Scientific Research Fund of Hunan Provincial Education Department under Grant 20C1759.

References

- [1] K. Tajik, A. Gunasekaran, R. Dutta, B. Ellis, R. B. Bobba, M. Rosulek, C. Wright, V. and W.-c. Feng, "Balancing image privacy and usability with thumbnail-preserving encryption," in *the 26th Annual Network and Distributed System Security Symposium (NDSS 2019)*, 2019.
- [2] G. Ye and X. Huang, "An image encryption algorithm based on auto-blocking and electrocardiography," *IEEE MultiMedia*, vol. 23, no. 2, pp. 64–71, 2016.
- [3] X. Li, L. Zhou, and F. Tan, "An image encryption scheme based on finite-time cluster synchronization of two-layer complex dynamic networks," *Soft Computing*, vol. 26, no. 2, pp. 511–525, 2021.
- [4] K. Kong, X. Wu, D. You, and H. Kan, "3D-BCNN based image encryption with finite computing precision," *IEEE MultiMedia*, 2022.
- [5] M. Li, H. Fan, Y. Xiang, Y. Li, and Y. Zhang, "Cryptanalysis and improvement of a chaotic image encryption by first-order time-delay system," *IEEE Multimedia*, vol. 25, no. 3, pp. 92–101, 2018.
- [6] Y. Ma, C. Li, and B. Ou, "Cryptanalysis of an image block encryption algorithm based on chaotic maps," *Journal of Information Security and Applications*, vol. 54, p. art. no. 102566, 2020.
- [7] J. Chen, L. Chen, and Y. Zhou, "Universal chosen-ciphertext attack for a family of image encryption schemes," *IEEE Transactions on Multimedia*, vol. 23, pp. 2372–2385, 2022.
- [8] L. Qu, F. Chen, S. Zhang, and H. He, "Cryptanalysis of reversible data hiding in encrypted images by block permutation and co-modulation," *IEEE Transactions on Multimedia*, vol. 24, pp. 2924–2937, 2022.
- [9] S. Liu, C. Li, and Q. Hu, "Cryptanalyzing two image encryption algorithms based on a first-order time-delay system," *IEEE MultiMedia*, vol. 29, no. 1, pp. 74–84, 2022.
- [10] L. Chen, C. Li, and C. Li, "Security measurement of a medical image communication scheme based on chaos and DNA coding," *Journal of Visual Communication and Image Representation*, vol. 83, p. art. no. 103424, 2022.
- [11] S. Zhou, Y. He, Y. Liu, C. Li, and J. Zhang, "Multi-channel deep networks for block-based image compressive sensing," *IEEE Transactions on Multimedia*, vol. 23, pp. 2627–2640, 2021.
- [12] S. Zhou, X. Deng, C. Li, Y. Liu, and H. Jiang, "Recognition-oriented image compressive sensing with deep learning," *IEEE Transactions on Multimedia*, 2022.
- [13] P. Ping, F. Xu, Y. Mao, and Z. Wang, "Designing permutation-substitution image encryption networks with Henon map," *Neurocomputing*, vol. 283, pp. 53–63, 2018.
- [14] Z. Hua, Z. Zhu, Y. Chen, and Y. Li, "Color image encryption using orthogonal Latin squares and a new 2D chaotic system," *Nonlinear Dynamics*, vol. 104, pp. 4505–4522, 2021.
- [15] Y. He, Y.-Q. Zhang, and X.-Y. Wang, "A new image encryption algorithm based on two-dimensional spatiotemporal chaotic system," *Neural Computing and Applications*, vol. 32, pp. 247–260, 2020.
- [16] F. Min, Y. Chen, L. Lu, and X. Li, "Extreme multistability and anti-monotonicity in a shinriki oscillator with two flux-controlled memristors," *International Journal of Bifurcation and Chaos*, vol. 31, no. 11, p. 2150167, 2021.
- [17] C. Li, B. Feng, S. Li, J. Kurths, and G. Chen, "Dynamic analysis of digital chaotic maps via state-mapping networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 6, pp. 2322–2335, 2019.
- [18] F. Zhang, X. Zhang, M. Cao, F. Ma, and Z. Li, "Characteristic analysis of 2D lag-complex Logistic map and its application in image encryption," *IEEE MultiMedia*, vol. 28, no. 4, pp. 96–106, 2021.
- [19] O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)*, vol. 29, 2016, pp. 3637–3645.
- [20] C. Li, D. Lin, and J. Lü, "Cryptanalyzing an image-scrambling encryption algorithm of pixel bits," *IEEE MultiMedia*, vol. 24, no. 3, pp. 64–71, 2017.