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

Chengqing Li and Xianhui Shen, Xiangtan University, Xiangtan, 411105, Hunan, China
Sheng Liu, Hunan University, Changsha, 410082, Hunan, China

This article conducts a security analysis of an image encryption algorithm that employs 2-D lag-complex logistic map (LCLM) as a pseudorandom number generator (PRNG). This algorithm uses the sum of all pixel values in the plain image as the initial value for the PRNG, thereby influencing the randomization of basic encryption operations. However, it is observed that certain factors lead to the generation of identical pseudorandom sequences for different plain images. Capitalizing on this vulnerability, we propose a chosen-plaintext attack strategy that effectively cracks the six encryption steps through a divide-and-conquer approach. By exploiting weaknesses inherent in the 2-D LCLM, the number of required chosen plain images is significantly reduced to \(5 \cdot \log_2(MN) + 95\), where \(MN\) represents the total pixel count of the plain image.

With the widespread adoption of multimedia capture devices and the growth of mobile Internet, an increasing number of individuals are enjoying the sharing and transmission of photos via social platforms, such as WeChat, Instagram, and Facebook. Concurrently, the openness of online networks poses a risk of multimedia information leakage during transmission, potentially exposing private content.1,2 This threat often goes unnoticed by the general populace. As a result, securing the privacy of multimedia data, particularly digital images, and finding a balance with usability has become imperative in our digitally interconnected world. Owing to the unique properties of image data, such as their substantial volume and the strong correlation between adjacent pixels, traditional text encryption methods—which convert objects into 1-D text—are largely impractical for image encryption. In light of this, a large number of image encryption algorithms have been proposed over the past three decades.3,4,5 Regrettably, from a modern cryptanalysis standpoint, some of these algorithms have been deemed insecure to varying degrees.6,7,8,9,10,11 Furthermore, with the advancement of neural network technology, the landscape of attack and defense mechanisms in image data security has undergone significant changes.5,12,13

As a cryptosystem can be conceptualized as a complex system within a digital domain, various nonlinear scientific approaches, such as chaos theory and synchronization, have been explored to innovate in the realm of image encryption algorithm design.6,14,15,16,17 However, the sophisticated dynamics, often evident in an infinite-precision domain, can be diminished or, in some cases, entirely lost when implemented in a finite-precision arithmetic domain.17 As demonstrated by Ma et al.7 and Chen et al.,11 the randomness of sequences generated by iterating a discrete chaotic map in a finite arithmetic precision computer falls short of expectations: notably, some sequences start with initial conditions that yield periods of less than three, as illustrated in Figure 1.

In Zhang et al.,18 the 2-D lag-complex logistic map (2-D LCLM) was introduced to bolster the dynamic complexity of the map and mitigate dynamic degradation in a digital domain. Following this, an image encryption algorithm based on this special logistic map [image encryption algorithm based on logistic map (IEALM)] was developed and evaluated for its cryptographic efficacy. The algorithm employs pseudorandom number sequences generated by iterating the 2-D LCLM to obscure and diffuse the plain image through a complex cascade of three design primitives: modular addition, bit-level permutation, and XOR operations.
This article reexamines the security performance of IEALM, highlighting several key findings: 1) the correlation mechanism between the encryption process and the plain image is occasionally ineffective; 2) certain inherent characteristics of the 2-D LCLM facilitate a chosen-plaintext attack; 3) the fundamental encryption components can be individually targeted using specifically chosen plain images; and 4) the attack’s efficacy is substantiated through experimental results and complexity analysis.

The rest of this article is organized as follows. The “Description of IEALM” section briefly introduces the procedure of IEALM. The “Cryptanalysis of IEALM” section presents the insecurity metrics of 2-D LCLM and the security vulnerabilities of IEALM together with some experimental results. The last section concludes the article.

**DESCRIPTION OF IEALM**

The encryption target of IEALM is an 8-bit red, green, blue (RGB) image of size $M \times N \times 3$. The algorithm directly separates the image into three distinct color channels for processing. For the sake of conciseness, the encryption process is described for only one channel. The plain image is scanned in the raster order and represented as $I = \{I(i)\}_{i=0}^{MN-1}$. Correspondingly, the cipher image is denoted as $I'' = \{P''(i)\}_{i=0}^{MN-1}$. The IEALM encryption method is delineated as follows.

The Secret Key

This comprises two control parameters of the 2-D 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 two distinct initial conditions: $K_1 = (x(0), y(0), z(0))$ and $K_2 = (x'(0), y'(0), z'(0))$. Here, $a = 2$, and $b$ is a value in the range of $[1.69, 2]$.

**Initialization**

- **Step 1**: Calculate two initial conditions through
  $$\begin{align*}
  K_1 &= (0.2 + X_r/10^9, 0.4 + Y_g/10^9, 0.1 + Z_b/10^9) \\
  K_2 &= (0.3 + X_r/10^9, 0.5 + Y_g/10^9, 0.2 + Z_b/10^9)
  \end{align*}$$

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

- **Step 2**: Iterate (1) $2MN + 250$ times from the initial condition $K_1$ and discard the first 250 elements to get three chaotic sequences
  $$X = \{x(i)\}_{i=0}^{2MN-1}, \quad Y = \{y(i)\}_{i=0}^{2MN-1}, \quad \text{and} \quad Z = \{z(i)\}_{i=0}^{2MN-1}.$$ 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
  $$\begin{align*}
  U(i) &= \lfloor x(i) \cdot 10^{15} \rfloor \mod 16 \\
  V(i) &= \lfloor \text{Dec}(y(i)) \cdot 10^{21} \rfloor \mod 256 \\
  W(i) &= \lfloor \text{Dec}(z(i)) \cdot 10^{21} \rfloor \mod 256
  \end{align*}$$

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

- **Step 4**: Generate two permutation vectors $T_1 = \{T_{1i}\}_{i=0}^{MN-1}$ and $T_2 = \{T_{2i}\}_{i=0}^{MN-1}$, where $T_{1i}$ and $T_{2i}$ are the order of $X(i)$ and $X(i+MN)$ in sets $\{X(i)\}_{i=0}^{MN-1}$ and $\{X(i)\}_{i=0}^{2MN-1}$, respectively. Similarly, obtain six permutation vectors of length $MN$, $T_1, T_2, T_3$, $T_4, T_5$, and $T_6$, from $Y$, $Z$, and $G$, respectively. Group these vectors as $T_1 = \{T_{1i}\}, T_2 = \{T_{2i}\}, T_3 = \{T_{3i}\}, T_4 = \{T_{4i}\}, T_5 = \{T_{5i}\}, T_6 = \{T_{6i}\}$.

- **Step 5**: Repeat steps 2, 3, and 4 using initial condition $K_2$ and produce the counterparts $X'$.
The Encryption Procedure

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

\[
I'(i) = I(i) ⊕ V(i)
\]

where \( a ⊕ b = (a+b) \mod 2^n \), and \( n_0 \) is the binary length of \( a \) and \( b \).

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

\[
I^*(i) = W(i) ⊕ I'(i)
\]

Step 3: Divide \( I^* \) into two 4-bit sequences \( L^* = \{ L^*(i) \}_{i=0}^{MN-1} \) and \( H^* = \{ H^*(i) \}_{i=0}^{MN-1} \) with function \( Split(X) \) that returns two sequences \( X_L = \{ X(i) \mod 16 \}_{i=0}^{MN-1} \) 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 \( L'' = \{ L''(i) \}_{i=0}^{MN-1} \) and \( H'' = \{ H''(i) \}_{i=0}^{MN-1} \): \( L''(i) = \sum_{k=0}^{3} L_k^* (T_{12}(i)) \cdot 2^k \) and \( H''(i) = \sum_{k=0}^{3} H_k^* (T_{12}(i)) \cdot 2^k \). Then, produce sequence \( L' \) by

\[
L'(i) = U(i) ⊕ L''(i) \oplus H''(i)
\]

Step 5: Perform bit-level permutation on \( L' \) and \( H'' \) to get sequences \( L' = \{ L'(i) \}_{i=0}^{MN-1} \) and \( H = \{ H(i) \}_{i=0}^{MN-1} \): \( L'(i) = \sum_{k=0}^{3} L_k^* (T_{12}(i)) \cdot 2^k \) and \( H(i) = \sum_{k=0}^{3} H_k^* (T_{12}(i)) \cdot 2^k \). Then, produce sequence \( H' \) by

\[
H'(i) = H(i) \oplus L'(i) \oplus H''(i)
\]

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

\[
I''(i) = W''(i) \oplus (I'(i) \oplus V'(i))
\]
Similarly, one can obtain
\[ y(i + 1) = b(y(i)(1 - (a(x(0)/y(0))^2 + 1)y(i - 1)^2). \]
Further, utilizing (1) for \( i \geq 1 \), one can obtain
\[ z(i + 1) = a(x(i))^2 + y(i)^2 \]
\[ = b^2(ax(i - 1)^2 + y(i - 1)^2)(1 - z(i - 1))^2 \]
\[ = b^2z(i)(1 - z(i - 1))^2. \]

The designers of IELM claimed that “2-D LCLM is 3-D in a real field,” as stated by Zhang et al.\textsuperscript{18} suggesting its superiority over other 2-D chaotic maps. However, Property 3 contradicts this assertion by demonstrating that the three maps of 2-D LCLM are independent and incapable of traversing the entire 3-D domain.

To further investigate the actual structure of 2-D LCLM within a computational environment, we conducted extensive tests on its functional graphs, following methodologies similar to those of Liu et al.\textsuperscript{10} and Chen et al.\textsuperscript{11} Three representative examples are illustrated in Figure 1. It was observed that every orbit inevitably enters a cycle after a transient phase, a phenomenon that persists regardless of the precision level implemented. Previous studies\textsuperscript{10} have shown that, alternating between multiple chaotic maps; or cascading states, or control parameters of the nonlinear map; the number of states available for discarding period cycle in the functional graph of the chaotic map, the number of states available for discarding becomes insufficient. Consequently, setting an adaptive threshold to circumvent this issue is necessary, though it introduces additional computational overhead.

Key Distribution
To counteract known/chosen-plaintext attacks, IELM employs plain-image information to generate distinct secret keys for different plain images. However, the uniformity of the key distribution has been overlooked. For every secret key to be selected with equal probability, it must adhere to a uniform distribution. A classic example that violates this principle is the Caesar cipher, which employs a codebook for direct letter substitution. Given that the frequency of letters in a language does not follow a uniform distribution, the cipher is vulnerable to statistical attacks.

In the case of IELM, the algorithm uses the sum of pixel values of the plain image to generate the initial values for 2-D LCLM. Since natural images inherently carry practical visual significance, their pixel sum is not random and deviates from a uniform distribution. To illustrate this characteristic of natural images, we analyzed the distribution of the average pixel value across 60,000 images from the Mini-ImageNet dataset.\textsuperscript{19} Our findings reveal that these values adhere to a normal distribution, as depicted in Figure 2.

According to the aforementioned experimental findings, selecting an arbitrary image of size 256 × 256 from Mini-ImageNet theoretically allows for the generation of approximately 4.7 \times 10^{21} potential keys. However, due to the normally distributed mean pixel values, the distribution of the resultant secret keys is non-uniform. Based on the characteristics of the normal distribution, it is found that \( \text{Prob}(81.641 < x_r < 159.609) = 0.6827 \), \( \text{Prob}(77.388 < y_g < 151.382) = 0.6827 \), and \( \text{Prob}(60.422 < z_b < 144.984) = 0.6827 \). For an attacker, targeting only this subset of the key space through a brute-force approach increases the probability of successfully compromising the algorithm to at least 30%. Intriguingly, this subset constitutes merely 2.91% of the entire key space.

Chosen-Plaintext Attack
Zhang et al.\textsuperscript{18} utilized the sum of pixel values of each plain image as the basis for the secret key, aiming to mimic the functionality of “one-time pads.” However, different plain images can yield identical keys. First, identical plain images will inevitably generate the same key. Second, distinct images with equal sums of pixel values can also result in identical secret keys. Furthermore, due to the finite arithmetic precision of

![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.](image-url)
computers, even plain images with differing average pixel values may generate the same secret key when processed through (2).

When one considers a plain image \(I_0\) and its corresponding cipher image \(I'_0\), it becomes feasible to decrypt another cipher image \(I'_0\) if two conditions are met simultaneously: 1) \(I'_0\) is encrypted using the same secret key as \(I'_0\), and 2) the corresponding plain image of \(I'_0\) produces the same outcome when calculated with (2) as \(I_0\), implying that the final sequences controlling the encryption process are identical.

**Simplification of IEALM**

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

\[
L'(i) = U(i) \oplus \tilde{W}_L(i) \oplus \tilde{W}_H(i) \oplus \tilde{L}^*(i) \oplus \tilde{H}^*(i) \tag{10}
\]

where

\[
L^*(i) = \tilde{L}^*(i) \oplus \tilde{H}^*(i)
\]

\[
\beta(i) = U(i) \oplus \tilde{W}_L(i) \oplus \tilde{W}_H(i) \oplus \tilde{L}^*(i) \oplus \tilde{H}^*(i) = \tilde{L}^*(i) \oplus \tilde{H}^*(i)
\]

\[
H'(i) = U'(i) \oplus \tilde{L}^*(i) \oplus \tilde{W}_H(i) \oplus \tilde{H}^*(i) = \tilde{L}^*(i) \oplus \tilde{H}^*(i) \tag{11}
\]

where

\[
H^*(i) = \tilde{L}^*(i) \oplus \tilde{H}^*(i)
\]

and it becomes feasible to get \(L'_0, H'_0\) when calculated with (2) as \(I_0\) and \(I'_0\), respectively. The corresponding cipher images \(I'_0\) and \(I''_0\) can be obtained using the same controlling sequences, we define the bit-wise XOR operation between them as \(I'_0 \oplus I''_0 = I'_0 \oplus I''_0\).

**Differential Attack on IEALM**

Given two plain images \(I_0\) and \(I_1\) encrypted by IEALM with the same controlling sequences, we define the bitwise XOR operation between them as \(I_{0,1} = I_0 \oplus I_1\). Likewise, define \(I'_{0,1} = I'_0 \oplus I'_1\) and \(H'_{0,1} = H'_0 \oplus H'_1\) where \((L_0, H_0) = \text{SpI}(I_0)\) and \((L_1, H_1) = \text{SpI}(I_1)\). For the corresponding cipher images \(I'_0\) and \(I''_0\), one can get \(L'_0, H'_0 = \text{SpI}(L'_0)\) and \(L''_0, H''_0 = \text{SpI}(L''_0)\). According to (3), one can obtain

\[
\begin{align*}
L'_{0,1} &= (L_0 \oplus V_L) \oplus (L_1 \oplus V_L) \\
H'_{0,1} &= (H_0 \oplus V_H) \oplus (H_1 \oplus V_H) \tag{12}
\end{align*}
\]

where \(r'_j = \lfloor (L'_j + V'_j)/16 \rfloor, j = 0, 1\). Analyzing (12) individually on each bit plane, one can get

\[
\begin{align*}
L'_{0,1}(i) &= \begin{cases} 
L_{0,1}(i) \oplus r_{0,1}(i) & \text{when } k = 0 \\
L_{0,1}(i) \oplus \theta_{0,1}(i) & \text{when } k = 1, 2, 3
\end{cases} \\
H_{0,1}(i) &= \begin{cases} 
H_{0,1}(i) \oplus r_{0,1}(i) & \text{when } k = 0 \\
H_{0,1}(i) \oplus \lambda_{0,1}(i) & \text{when } k = 1, 2, 3
\end{cases}
\end{align*} \tag{13}
\]

and

\[
H_{0,1}(i) = \begin{cases} 
H_{0,1}(i) \oplus r_{0,1}(i) & \text{when } k = 0 \\
H_{0,1}(i) \oplus \lambda_{0,1}(i) & \text{when } k = 1, 2, 3
\end{cases}
\tag{14}
\]

where

\[
\theta_{j,k}(i) = \frac{\sum_{i=0}^{k-1} (V_{L,j}(i) + L_{j,k}(i)) \cdot 2^j}{2^k}
\]

\[
\lambda_{j,k}(i) = \frac{\sum_{i=0}^{k-1} (V_{H,j}(i) + H_{j,k}(i)) \cdot 2^j + r_j(i)}{2^k}
\]

and \(j = 0, 1\). According to (10), by differentiating the plain images, one has

\[
L'_0(i) = L'_0(i) \oplus \tilde{L}'_0(i)
\]

Then, in the \(k\) th bit plane, one can get

\[
L'_{0,1}(i) = L'_{0,1}(i) \oplus \tilde{L}'_{0,1}(i)
\]

Similarly, from (11), one can get

\[
H'_{0,1}(i) = H'_{0,1}(i) \oplus \tilde{H}'_{0,1}(i)
\]

According to (7), one has

\[
\begin{align*}
L''_{0,1} &= \begin{cases} 
L''_0(i) \oplus V'_L(i) \oplus L'_1(i) \oplus V'_L(i) & \text{when } k = 0 \\
L''_{0,1}(i) \oplus H'_0(i) \oplus V'_H(i) \oplus H'_1(i) \oplus V'_H(i) & \text{when } k = 1, 2, 3
\end{cases} \\
H''_{0,1} &= \begin{cases} 
H''_0(i) \oplus r'_0(i) \oplus H'_1(i) \oplus V'_H(i) \oplus r'_1(i) & \text{when } k = 0 \\
H''_{0,1}(i) \oplus \lambda''_{0,1}(i) & \text{when } k = 1, 2, 3
\end{cases}
\end{align*} \tag{18}
\]

and

\[
H''_{0,1}(i) = \begin{cases} 
H''_{0,1}(i) \oplus r''_{0,1}(i) & \text{when } k = 0 \\
H''_{0,1}(i) \oplus \lambda''_{0,1}(i) & \text{when } k = 1, 2, 3
\end{cases}
\tag{19}
\]
where

\[
\begin{align*}
\theta'_{i,k}(i) &= \left[ \sum_{j=0}^{k-1} (V_{L,i}(i) + L'_{j,i}(i)) \cdot 2^j \right] \mod 2^k \\
\chi'_{i,k}(i) &= \left[ \sum_{j=0}^{k-1} (V_{L,i}(i) + H'_{j,i}(i)) \cdot 2^j + r'_{j,i}(i) \right] \mod 2^k
\end{align*}
\]

(20)

and \( j = 0, 1 \).

**Determining \( T_2 \)**

The bit-level permutation using \( T_{2,k} \) can be determined using (14), (16), and (18). For the case when \( k = 0 \), it is sufficient to nullify \( L'_{0@1}(i) \) and \( r_{0@1}(i) \) to facilitate the attack on the permutation. By setting \( L'_{0@1}(i) = 0 \), it follows that \( r_{0@1}(i) = 0 \) and \( L'_{0@1}(i) = 0 \), leading to (16) to simplify to \( L''_{0@1,k}(i) = H^*_{0@1,k}(T_{2,k}(i)) \).

When \( k > 0 \), one needs to additionally eliminate the influence of the carries from the lower bit planes, namely, \( \lambda_{0@1,k}(i) \) and \( \theta_{0@1,k}(i) \). Therefore, for each \( k' = 0 \sim (k - 1) \), setting \( H_{0@1,k'}(i) = 0 \) results in \( \lambda_{0@1,k}(i) = 0 \), as indicated by (15). This also leads to \( L''_{0@1,k}(i) = 0 \) and \( \theta_{0@1,k}(i) = 0 \), as inferred from (20).

Under these conditions, for \( k = 0 \sim 3 \), (18) is reduced to

\[
L''_{0@1,k}(i) = L'_{0@1,k}(i) = H_{0@1,k}(T_{2,k}(i)).
\]

In this scenario, the encryption operation on \( H_{0@1,k}(i) \) effectively becomes a permutation-only process. Given the maturity of chosen-plaintext attacks on permutation-only encryption algorithms, the attack procedure can be succinctly outlined. The permutation vector \( T_{2,k} \) can be precisely reconstructed through 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 \( I_1, I_2, \ldots, I_n \) that satisfy \( I_i = (H_i(i) \cdot 16)^{MN-1} \), and the \( k' \)th significant bit of \( H_i(i) \) is

\[
H_{i,k'}(i) = \begin{cases} 
\lfloor i/2^{k'-1} \rfloor \mod 2 & \text{if } k' = k \\
0 & \text{otherwise}
\end{cases}
\]

where \( n = \lceil \log_2(MN) \rceil \), \( t \in \{1, 2, \ldots, n\} \), and \( k' \in \{0, 1, 2, 3\} \).

- **Step 3:** Encrypt \( I_1, I_2, \ldots, I_n \) and get the corresponding cipher images \( I''_{0@1,k} \). Then, for \( \sum_{i=1}^{n} L''_{0@1,k}(i') \cdot 2^{k'-1} = i \); one can confirm \( T_{2,k}(i') = i \), where \( L''_{0@1,k}(i') \equiv I''_{0@1}(i') \mod 16 \).

Perform these 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 \( H_{0@1}(i) = 0 \), \( L_0(i) = 0 \), and \( L_1(i) = c \) for \( i = 0, 1, \ldots, MN - 1 \), where \( c \in \{1, 2, \ldots, 15\} \). For \( k = 0 \), (18) can be expressed as

\[
L''_{0@1,k}(i) = L''_{0@1,k}(i) = L''_{0@1,k}(T_{2,0}(i))
\]

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

\[
L''_{0@1,k}(i) = \begin{cases} 
(c \mod 2) \oplus 1 & \text{if } (c + V_L(T_{2,0}(i))) \geq 16 \\
(c \mod 2) & \text{otherwise}
\end{cases}
\]

Setting \( c = 1 \), one can deduce that

\[
V_L(T_{2,0}(i)) \in \begin{cases} 
\{15\} & \text{if } L''_{0@1,k}(i) = 0 \\
\{0, 1, \ldots, 14\} & \text{otherwise}
\end{cases}
\]

Namely, one can determine \( V_L(T_{2,0}(i)) = 15 \) by finding \( L''_{0@1,k}(i) = 0 \). Similarly, one can set \( c = 2 \) and get \( V_L(T_{2,0}(i)) = 14 \) by checking

\[
V_L(T_{2,0}(i)) \in \begin{cases} 
\{14, 15\} & \text{if } L''_{0@1,k}(i) = 1 \\
\{0, 1, \ldots, 13\} & \text{otherwise}
\end{cases}
\]

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

**Determining \( T_1 \)**

The permutation vector \( T_{1,k} \) can be recovered using (13), (16), and (18). To eliminate \( H''_{0@1,k}(T_{2,k}(i)) \) in (16), one can set \( H_0(i) = r_1(i) \) and \( H_1(i) = r_0(i) \) to get \( H''_{0@1,k}(i) = 0 \). When \( k > 0 \), choosing \( L'_{0@1,k}(i) = 0 \) for \( k' = 0 \sim (k - 1) \), one can deduce \( \theta_{0@1,k}(i) = 0 \) from (15).

Furthermore, because \( L''_{0@1,k}(i) = H''_{0@1,k}(T_{1,k}(i)) = L''_{0@1,k}(i) = 0 \), the carry \( \theta_{0@1,k}(i) = 0 \) according to (20).

Then, (18) becomes

\[
L''_{0@1,k}(i) = L''_{0@1,k}(i) = L''_{0@1,k}(T_{1,k}(i))
\]

Therefore, \( T_1 \) can be exactly determined by utilizing the recovery procedure of \( T_2 \). In this case, the required number of plain images is \( 2 \cdot \lceil \log_2(MN) \rceil + 1 \) since the settings of \( I_0 \) and \( I_1 \) are correlated.

**Determining \( V_H \)**

The three least significant bits of \( V_H \) can be guessed by observing the corresponding carry incurred by the addition in (12). Setting \( I_0(i) = 0 \), it follows that
From (16), one can express $V_{h,k}$ where $k < 3$, and $a \equiv b \mod{2^3}$.

Noting that $r_1(i)$ is also known and that $L_1(i) = L^{\dagger}_1(i) \oplus V_{j}(i)$, (12) becomes
\[
H^0_{011}(i) = V_{h}(i) \oplus (2^3 \oplus V_{b}(i)).
\]

Consequently, one can deduce $L^0_{011}(i) = H^0_{011}(i) = H^0_{011}(i) = 1$. For $k > 0$, in the case of lower bit planes, $L^0_{011}(i) = H^0_{011}(i) = 0$ holds, where $k' = 0 \sim (k-1)$. From (16), one can get $L^0_{011}(i) = 0$ and $L^0_{011}(i) = 0$. Furthermore, the carry $0^0_{011}(i) = 0$ according to (20). Hence, (18) becomes
\[
L^0_{011}(i) = H^0_{011}(i) \oplus H^0_{011}(i) \oplus H^0_{011}(i).
\]

When one considers $L^0_{011}(i) = 0$, $H^0_{011}(i) = (T_{2k+1}(i)) = 0$, and $\lambda_{0}(i1) = (T_{2k+1}(i)) = 0$, the equation simplifies to
\[
L^0_{011}(i) = \lambda_{1}(i1) = (T_{2k+1}(i)).
\]

Incorporating $\sum_{i=0}^{k} H_{i}(i) \cdot 2^i + r_1 = 2^k$ into (15) yields
\[
\lambda_{1}(i1) = \left\{ \begin{array}{cl} 0 & \text{when } H_{1}(i) = 0 \\ 1 & \text{when } H_{1}(i) = 1 \end{array} \right.
\]

which implies $\lambda_{1}(i1) = H_{1}(i)$. Thus,
\[
V_{h,k}(T_{2k+1}(i)) = H^0_{011}(i).
\]

Then, one can recover $V_{h0}(i), V_{h1}(i), V_{h2}(i)$ by setting $k = 0, 1, 2$ in turn. Shifting $V_{h3}(i)$ to $W_{h3}(i)$, one can set $V_{h3}(i) = 0$ according to Fact 1.

Since the equivalent version of $V$ is known, any $I^*$ can be converted to $I$ via $I = I^* \boxplus V$. Therefore, in subsequent discussions, we only consider the selection of $I^*$ without specifying the plain image $I$.

Determining $T_1$: The permutation vector $T_{1k}$ can be determined using (17) and (19). By setting $L^0_{011}(i) = 0$ and $\xi^0_{011}(i) = H^0_{011}(i)$, it follows that $L^0_{011}(i) = L^0_{011}(i) = 0$, and $r^0_{01}(i) = r^0_{01}(i)$. From (17), it can be deduced that $H^0_{011}(i) = H^0_{011}(i) = (T_{1k}(i))$. For $k > 0$, setting $H_{011}(i) = 0$ for each $k' = 0 \sim (k-1)$ helps in eliminating the influence of the carry $\lambda^0_{011}(i)$, leading to $H^0_{011}(i) = 0$. Consequently, $\lambda^0_{011}(i)$ equals zero, as per (20). Therefore, for $k = 0 \sim 3$, (19) simplifies to
\[
H^0_{011}(i) = H^0_{011}(i) = H^0_{011}(i).
\]

Similarly, $T_1$ can be precisely recovered by applying the same recovery procedure as that used for $T_2$.

Determining $T_3$: Let $L^0_{011}(i) = H^0_{011}(i) = H^0_{011}(i) = 0$ for $i = 0, 1, \cdots, (MN-1)$. From $L^0_{011}(i) = L^0_{011}(i) \oplus H^0_{011}(i)$, one can get $L^0_{011}(i) = 0, L^0_{011}(i) = L^0_{011}(i)$, $H^0_{011}(i) = 0$, and $H^0_{011}(i) = H^0_{011}(T_{3k}(i))$. Since $L^0_{011}(i) = L^0_{011}(i)$, and $T_1$ is known, we directly choose $T^0_{1k}$ without specifying the corresponding plain image in the succeeding discussion.

Based on (17) and (19), one has
\[
H^0_{011}(i) = H^0_{011}(i) \oplus r^0_{011}(i) = L^0_{011}(T_{3k}(i)) \oplus r^0_{011}(i).
\]

When $L^0_{011}(i) = 0$, one has $\Phi(i) = 0$. To determine $\Phi(i)$, assign $L^0_{011}(i) = 1$ for $i = 0 \sim (MN-1)$, leading to $\Phi(i) = H^0_{011}(i) \oplus 1$ from preceding equation. By choosing $L^0_{011}(i) = \{0, 1\}$ and knowing $\Phi(i), L^0_{011}(i)$, $T_{3k}$ can be recovered using the aforementioned equation, similar to the recovery process of $T_{2k}$.

To determine $T_{3k}$ for $k \in \{1, 2, 3\},$ consider
\[
H^0_{011}(i) = H^0_{011}(i) \oplus L^0_{011}(i) \oplus L^0_{011}(i) = L^0_{011}(T_{3k}(i)) \oplus L^0_{011}(i).
\]

From (11), one can get $H^0_{011}(i) = \beta^0_{011}(i)$. Setting $L^0_{011}(i) \in \{0, 2^k\}$, for $i = 0 \sim (k-1)$, $H^0_{011}(i) = L^0_{011}(T_{3k}(i)) = 0$ and then $\beta^0_{011}(i) = \beta^0_{011}(i)$.

According to (20), one has
\[
\lambda^0_{011}(i) = \left[ \begin{array}{c} 2x(i) + r^0_{011}(i) \\ 2^k \end{array} \right] \oplus \left[ \begin{array}{c} z_0(i) + \beta^0_{011}(i) \\ 2^k \end{array} \right] = \left[ \begin{array}{c} 2x(i) + |(\beta^0_{011}(i) + V_{i}^0(i))/16| \\ 2^k \end{array} \right] \oplus \left[ \begin{array}{c} z_0(i) + \beta^0_{011}(i) + V_{i}^0(i)/16 \\ 2^k \end{array} \right]
\]

where $x(i) = \sum_{0<i} \left( V_{i}^0(i) + \beta^0_{011}(i) \right) \cdot 2^i$. We use $\Psi(i, L^0_{011}(i))$ to represent $L^0_{011}(i)$ for different $L^0_{011}(i)$ values. Then, (22) can be formulated as
\[
H^0_{011}(i) \oplus \Psi(i, L^0_{011}(i)) = L^0_{011}(T_{3k}(i)).
\]

For $L^0_{011}(i) = 0$, one can know $\Psi(i) = 0$. To find $\Psi(i, 2^k)$, set $L^0_{011}(i) = 2^k$ for $i = 0 \sim (MN-1)$; then, $\Psi(i, 2^k) = H^0_{011}(i) \oplus 1$ is derived from the equation. By selecting $L^0_{011}(i) \in \{0, 2^k\}$ and knowing $\Psi(i, L^0_{011}(i))$, $T_{3k}$ can be recovered using the equation and the recovery process analogous to $T_{2k}$. Thus, $T_3 = \{T_{30}, T_{31}, T_{32}, T_{33}\}$ is effectively determined.
Attacking the Other Encryption Operations

Once all permutation vectors are successfully recovered, one can get
\[ I_{00}(i) = W_{0}(i) / H_{20003}(I_{00}(i) \boxplus V_{0}(i)) = W_{0}(i) / H_{20003}(b(i) + 16 / C_{1} b_{0}(i)) / H_{20003}(I(i) \boxplus V(i)) \]
from (7), (10) and (11), where
\[ I_{00}(i) = L_{0}(i) + H_{0}(i) \cdot 16. \]
In the preceding equation, \( I_{00}(i) \) is determined by \( I'(i) \), and vice versa. Consequently, any \( I' \) can be chosen to obtain its corresponding plain image \( I \). By setting \( I'(i) = c \) for \( c = 0 \sim 255 \), \( I_{00}(i) \) can be obtained, leading to the formation of a map \( F(i, I_{00}(i)) = c \). This map can then be used to recover \( I'(i) \) from \( I_{00}(i) \).

Recovering the Plain Image

To recover the corresponding plain image from a given cipher image \( I' \), the following steps should be taken:

1) Recover \( I' \) using \( I'(i) = F(i, I_{00}(i)) \) and then calculate \( (L^*, H^*) = \text{SPl}(I') \).

2) Perform the permutation with \( T_3 \) on \( L^* \) to obtain \( \hat{L}^* \) and derive \( H^* \) as \( \hat{H}^*(i) = H^*(i) \oplus \hat{L}^*(i) \).

3) Apply the inverse permutation with \( T_4 \) on \( \hat{H}^* \) to recover \( H^* \).

4) Permute \( H^* \) with \( T_2 \) to obtain \( \hat{H}^* \) and derive \( \hat{L}^* \) as \( \hat{L}^*(i) = L^*(i) \oplus \hat{H}^*(i) \).

5) Employ the inverse permutation with \( T_1 \) on \( \hat{L}^* \) to recover \( L^* \).

6) Reproduce the plain image \( I \) using the formula
\[ I(i) = (L^*(i) + 16 \cdot H^*(i)) \boxplus V(i). \]

To validate the effectiveness of the proposed attack method, numerous experiments were conducted using various random secret keys. The results of these attacks on a representative image are depicted in Figure 3. Remarkably, the success rate of

Figure 3. Chosen-plaintext attack results on IEALM. (a) Plain image “Lenna.” (b) Cipher image of Figure 3(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 Figure 3(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.

The source codes for this research are available at https://github.com/chengqingli/mm-iealm.
TABLE 1. Some data in the encryption process of the blue channel of Figure 3(b).

| Item   | 1  | 2  | 4  | 8  | 16 | 32 | 64 | 128 | 256 | 512 | 1024 |
|--------|----|----|----|----|----|----|----|-----|-----|-----|-----|
|        | 125| 123| 122| 114| 110| 83 | 100| 106 | 125 | 122 | 114  |
| $T^{20}(i)$ | 63,654 | 41,166 | 44,389 | 5418 | 60,541 | 8324 | 8394 | 52,758 | 10,693 | 18,236 | 12,940 |
| $T^{13}(i)$ | 62,246 | 12,618 | 22,576 | 424 | 5892 | 47,186 | 18,568 | 14,185 | 4948 | 47,571 | 6740 |
| $T^{12}(i)$ | 8436 | 37,177 | 13,122 | 24,285 | 25,840 | 24,911 | 350 | 52,730 | 12,436 | 30,075 | 132  |
| $T^{33}(i)$ | 1357 | 27,981 | 60186 | 16,982 | 691 | 9877 | 32,352 | 30,284 | 62,723 | 61,986 | 27,694 |
| $V(i)$ | 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  |

The items obtained in an attack corresponding to the data shown in Table 1.

The items obtained in an attack corresponding to the data shown in Table 1.

Complexity of the Attack
Recalling the attack process, considering $T_{1,0} = T_{1,1}$, $T_{2,0} = T_{2,1}$, $T_{3,0} = T_{3,1}$, and $T_{4,0} = T_{4,1}$, one can see that the required number of RGB images in the procedure for determining $T_2$ and $T_1$ is $\log_2(MN) + 1$ each. Recovering $V$ needs six plain images. The determination of $T_1$ and $T_3$ requires $2 \cdot \log_2(MN)$ and $\log_2(MN)$ + 1 plain images, respectively. The final step to obtain the map $F(i, I''(i))$ demands 86 images. Thus, the complete chosen-plaintext attack entails $96 + 5 \cdot \log_2(MN)$ RGB plain images. The time complexity of this attack is approximately $O(MN \cdot \log_2(MN))$, primarily expended in determining the permutation vectors and formulating the map. For instance, attacking a $256 \times 256$-sized image, as shown in Figure 3(c), requires 175 plain images and can be executed in just a few seconds on a standard personal computer.

Analysis of the Key Space
The designers of IEALM\(^{18}\) claimed that its key space is greater than $10^{120}$. However, the actual key space of IEALM is contingent upon the arithmetic precision $e$ and the pixel values of the plain image. Despite the pseudorandom number generator having six initial values, they are entirely dependent on the pixel values of the plain image. Consequently, the actual key space of IEALM falls short of the designers’ estimation. Precisely, the key space of IEALM is $2^{2e} \cdot (256MN)^3$. For a $2048 \times 2048$-sized plain image, the key space sizes for IEALM implemented with 32-bit and 64-bit precision are $10^{46}$ and $10^{65}$, respectively. Similarly, for a $256 \times 256$ image, these sizes reduce to $10^{36}$ and $10^{20}$, respectively. In any scenario, the key space size is substantially less than the claimed $10^{120}$. It is noteworthy that this estimation represents the maximum theoretical value.

TABLE 2. The items obtained in an attack corresponding to the data shown in Table 1.

| Item   | 1  | 2  | 4  | 8  | 16 | 32 | 64 | 128 | 256 | 512 | 1024 |
|--------|----|----|----|----|----|----|----|-----|-----|-----|-----|
| $T^{20}(i)$ | 63,654 | 41,166 | 44,389 | 5418 | 60,541 | 8324 | 8394 | 52,758 | 10,693 | 18,236 | 12,940 |
| $T^{13}(i)$ | 62,246 | 12,618 | 22,576 | 424 | 5892 | 47,186 | 18,568 | 14,185 | 4948 | 47,571 | 6740 |
| $T^{12}(i)$ | 8436 | 37,177 | 13,122 | 24,285 | 25,840 | 24,911 | 350 | 52,730 | 12,436 | 30,075 | 132  |
| $T^{33}(i)$ | 1357 | 27,981 | 60186 | 16,982 | 691 | 9877 | 32,352 | 30,284 | 62,723 | 61,986 | 27,694 |
| $V(i)$ | 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  |
Given the nonuniform distribution of keys and the limited randomness of sequences generated by certain control parameters in a chaotic map, the actual size of the effective key space is considerably smaller.

CONCLUSION

This article presents an analysis of the security performance of an image encryption algorithm underpinned by 2-D LCLM, utilizing the lens of modern cryptanalysis. It was observed that the sensitivity mechanism between the controlling pseudorandom sequences and the plain image could be nullified for certain specific plain images. Given that the algorithm comprises a cascade of independent basic encryption components, their equivalents were methodically recovered using selected plain images. Further, the properties of the underlying chaotic map were examined to boost the efficiency of the attack. Both theoretical analysis and experimental results are provided, affirming the efficacy of the proposed attack. The findings underscore that the design of a secure and efficient image encryption algorithm requires careful consideration of multiple factors, such as the unique storage format of image data, the randomness quality of the chosen pseudorandom number generator, the security demands of the application context, and the genuine structure and computational complexity of the algorithm itself.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China (92267102).

REFERENCES

1. K. Tajik et al., “Balancing image privacy and usability with thumbnail-preserving encryption,” in Proc. 26th Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS), 2019, pp. 1–15, doi: 10.14722/ndss.2019.23432.
2. R. Lin, S. Liu, J. Jiang, S. Li, C. Li, and C.-C. J. Kuo, “Recovering sign bits of DCT coefficients in digital images as an optimization problem,” J. Vis. Commun. Image Representation, vol. 98, Feb. 2024, Art. no. 104045, doi: 10.1016/j.jvcir.2023.104045.
3. G. Ye and X. Huang, “An image encryption algorithm based on autoblocking and electrocardiography,” IEEE Multimedia, vol. 23, no. 2, pp. 64–71, Apr./Jun. 2016, doi: 10.1109/MMUL.2015.72.
4. X. Li, L. Zhou, and F. Tan, “An image encryption scheme based on finite-time cluster synchronization of two-layer complex dynamic networks,” Soft Comput., vol. 26, no. 2, pp. 511–525, 2021, doi: 10.1007/s00500-021-06500-y.
5. K. Kong, X. Wu, D. You, and H. Kan, “3D-BCNN-based image encryption with finite computing precision,” IEEE Multimedia, vol. 29, no. 4, pp. 97–110, Oct./Dec. 2022, doi: 10.1109/MMUL.2022.3194066.
6. 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, Jul./Sep. 2018, doi: 10.1109/MMUL.2018.112142439.
7. Y. Ma, C. Li, and B. Ou, “Cryptanalysis of an image block encryption algorithm based on chaotic maps,” J. Inf. Secur. Appl., vol. 54, Oct. 2020, Art. no. 102566, doi: 10.1016/jjis.2020.102566.
8. J. Chen, L. Chen, and Y. Zhou, “Universal chosen-ciphertext attack for a family of image encryption schemes,” IEEE Trans. Multimedia, vol. 23, pp. 2372–2385, 2021, doi: 10.1109/TMM.2020.3011315.
9. L. Qu, F. Chen, S. Zhang, and H. He, “Cryptanalysis of reversible data hiding in encrypted images by block permutation and co-modulation,” IEEE Trans. Multimedia, vol. 24, pp. 2924–2937, 2022, doi: 10.1109/TMM.2021.3090588.
10. 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, Jan./Mar. 2022, doi: 10.1109/MMUL.2021.3114589.
11. L. Chen, C. Li, and C. Li, “Security measurement of a medical communication scheme based on chaos and DNA coding,” J. Vis. Commun. Image Representation, vol. 83, Feb. 2022, Art. no. 103424, doi: 10.1016/j.jvcir.2021.103424.
12. S. Zhou, Y. He, Y. Liu, C. Li, and J. Zhang, “Multi-channel deep networks for block-based image compressive sensing,” IEEE Trans. Multimedia, vol. 23, pp. 2627–2640, 2021, doi: 10.1109/TMM.2020.3014561.
13. S. Zhou, X. Deng, C. Li, Y. Liu, and H. Jiang, “Recognition-oriented image compressive sensing with deep learning,” IEEE Trans. Multimedia, vol. 25, pp. 2022–2032, 2023, doi: 10.1109/TMM.2022.3142952.
14. P. Ping, F. Xu, Y. Mao, and Z. Wang, “Designing permutation-substitution image encryption networks with Henon map,” Neurocomputing, vol. 283, pp. 53–63, Mar. 2018, doi: 10.1016/j.neucom.2017.12.048.
15. Z. Hua, Z. Zhu, Y. Chen, and Y. Li, “Color image encryption using orthogonal Latin squares and a new 2D chaotic system,” Nonlinear Dyn., vol. 104, no. 4, pp. 4505–4522, 2021, doi: 10.1007/s11071-021-06472-6.
16. Y. He, Y.-Q. Zhang, and X.-Y. Wang, “A new image encryption algorithm based on two-dimensional spatiotemporal chaotic system,” Neural Comput. Appl., vol. 32, no. 1, pp. 247–260, 2020, doi: 10.1007/s00521-018-3577-z.
17. X. Lu, E. Y. Xie, and C. Li, “Periodicity analysis of Logistic map over ring Z_n,” Int. J. Bifurcation Chaos,
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, Oct./Dec. 2021, doi: 10.1109/MMUL.2021.3080579.

19. O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, “Matching networks for one shot learning,” in *Proc. 30th Int. Conf. Neural Inf. Process. Syst., (NIPS)*, 2016, vol. 29, pp. 3637–3645, doi: 10.5555/3157382.3157504.

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, Aug. 2017, doi: 10.1109/MMUL.2017.3051512.

**CHENGQING LI** is a professor with the School of Computer Science, Xiangtan University, Xiangtan, 411105, Hunan, China. His research interests include image privacy protection and multimedia cryptanalysis. Li received his Ph.D. degree in electronic engineering from the City University of Hong Kong. He is a fellow of the Institution of Engineering and Technology. He is the corresponding author of this article. Contact him at chengqingg@gmail.com.

**XIANHUI SHEN** is a graduate student in computer science at the School of Computer Science, Xiangtan University, Xiangtan, 411105, Hunan, China. His research interests include multimedia cryptanalysis. Shen received his B.Sc. degree in computer science from Xiangtan University. Contact him at xianhui_shen@163.com.

**SHENG LIU** is with the School of Computer Science and Electronic Engineering, Hunan University, Changsha, 410082, Hunan, China. His research interests include image privacy protection and image forensics. Liu received his master’s degree in computer science from Hunan University. Contact him at shengliu@hnu.edu.cn.