entropy

MDPI

Article

Cryptanalysis and Improvement of a Chaotic Map-Based Image Encryption System Using Both Plaintext Related Permutation and Diffusion

Cheng-Yi Lin and Ja-Ling Wu *

Department of Computer Science and Information Engineering, Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei 106, Taiwan; sincerity@cmlab.csie.ntu.edu.tw

* Correspondence: wjl@cmlab.csie.ntu.edu.tw

Received: 27 April 2020; Accepted: 20 May 2020; Published: 22 May 2020

Abstract: In theory, high key and high plaintext sensitivities are a must for a cryptosystem to resist the chosen/known plaintext and the differential attacks. High plaintext sensitivity can be achieved by ensuring that each encrypted result is plaintext-dependent. In this work, we make detailed cryptanalysis on a published chaotic map-based image encryption system, where the encryption process is plaintext Image dependent. We show that some designing flaws make the published cryptosystem vulnerable to chosen-plaintext attack, and we then proposed an enhanced algorithm to overcome those flaws.

Keywords: image encryption; chaotic map; permutation; diffusion; cryptanalysis

1. Introduction

With the rapid progress in digital technology and mobile devices, people have produced more and more user-generated information in these few years; besides texts, most of them are multimedia data, such as images and videos. With the increasing of information security and privacy protection issues, researchers have proposed lots of encryption algorithms [1] against unauthorized access to those user-generated media data.

As suggested in [2], the main techniques used to develop image encryption algorithms can roughly be divided into the following six categories: chaotic map, DNA computing, cellular automata, wavelet transmission, neural networks, and compressive sensing. The extreme initial value sensitivity and high randomness of the chaotic systems make the chaotic maps the most popular tool in digital image encryption algorithms. This is because the chaotic systems have some useful properties, like being ergodic, highly sensitive to initial conditions, and pseudo-randomness, which fit the essential requirements for building a practical cryptosystem [3]. Fridrich [4] proposed the first image encryption algorithm based on a chaotic map in 1998. After that, a large number of digital image encryption algorithms that were based on chaotic maps were proposed [3], and the references therein. Since Liu et al. addressed lots of the chaotic map-based image encryption algorithms published on signal processing and information technology-related Journals before 2019, our following discussions will mainly focus on related works [5–10] posted on the Entropy Journal, most recently.

In order to prevent an Image exchanging system from brute force and differential attacks, [5] presented a new image encryption mechanism, in which the Enhanced Logistic Map (ELM) and some simple encryption techniques, such as block scrambling, modified zigzag transformation, and chaotic-map based key generation, are used. The results of encryption are evaluated from six different security measures. The corresponding results demonstrate the security, reliability, efficiency, and flexibility of the proposed method. Sine-Tent map (STM) is intended in [7] to widen the chaotic

range and to improve the shambolic performance of one-dimensional (1D) discrete chaotic maps. Based on STM, a novel double S-box based color image encryption algorithm is recommended, which offers better applicability in real-time image encryption. Notice that, in [7], the 256-bit hash value of a randomly sampled noise signal is applied to serve as the one-time initial values of the proposed system. Since there is only one operation, XOR, is used to diffuse the pixels; the encryption process can be executed very fast. [8] presents a chaotic-map based image encryption algorithm, where Logistic and Henon maps are used.

High key and high plaintext sensitivities are a must for a cryptosystem to resist the chosen/known plaintext and the differential attacks. High plaintext sensitivity can be achieved by ensuring that each encrypted result is plaintext-dependent. To reach this goal, [9] suggested that the surrounding of a plaintext image could be surrounded by a sequence generated from the SHA-256 hashed value of the corresponding plaintext. For conquering the same challenges, in [10], both the permutation and the diffusion stages of the proposed color image encryption scheme are related to the original plain image. For keeping high system efficiency, only one round of plaintext tied permutation and diffusion operations are performed for obtaining the cipher image. Moreover, the proposed approach can be applied to real-time image encryption directly since there is no need to send original image dependent security keys to the receiver. Our work is highly inspired by and related to [10]; we will explore it further in the next section.

In general, the analyses of encryption and decryption algorithms show that all of the algorithms, as mentioned above, have a good encryption effect, anti-attack ability, and high security. However, a minor designing flaw may make encryption algorithms vulnerable, even if they are chaotic-based. In the following, we will take the high plaintext sensitivity related work: "A simple Chaotic map-based Image Encryption System Using Both Plaintexts Related Permutation and Diffusion (CIES-UBPRPD)", as proposed by Huang et al. [10], as an example to illustrate our above statement.

In a plain data-dependent cryptosystem, like CIES-UBPRPD, the cryptanalysis complexity is mostly increased. Therefore, the security level of the system will also be enhanced. Even though the experiments given in CIES-UBPRPD [10] showed lots of advantages when compared with conventional approaches, some designing flaws have been found by us. In this work, we first break a simplified version of CIES-UBPRPD with a chosen-plaintext attack, to demonstrate the effect of the discovered flaws. Subsequently, we make a few adjustments on CIES-UBPRPD and show that the modified version does relieve the defect of the original CIES-UBPRPD.

The rest of this paper is organized, as follows. Section 2 briefly describes the process of CIES-UBPRPD. In Section 3, we pointed out some design flaws of CIES-UBPRPD and demonstrated the effects of the weaknesses by issuing a chosen-plaintext attack against a simplified version of it. Afterwards, a modified version of CIES-UBPRPD is provided in Section 4. With the added supplements, the security level and the completeness of CIES-UBPRPD can be enlarged significantly. Some experimental results of the modified CIES-UBPRPD are presented in Section 5, in order to verify our previous claim, where the associated security analysis is also included. Finally, Section 6 concludes this writeup.

2. Related Work

In the original CIES-UBPRPD [10], all the arrays are started with index one, while in this work, we use an equivalent description but change all array indexes starting from zero. For the ease of discussion, except for the array indexes, most of our notations follow the usage that was adopted in [10].

2.1. The Involved Chaotic Maps

2.1.1. Generalized Arnold's Cat Map

Arnold's Cat Map (ACM) is a well-known two-dimensional chaotic system proposed by the Russian mathematician Vladimir I. Arnold [11]. ACM is usually replaced by its generalized form to achieve higher security and higher randomness, as shown in Equation (1):

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & a \\ b & ab+1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \mathrm{mod} \begin{bmatrix} M \\ N \end{bmatrix}, \tag{1}$$

where (x, y) and (x', y') denote the positions of the original pixel and the target pixel, a and b are the system parameters, while M and N are the image's height and width, respectively. After obtaining the target position (x', y'), from Equation (1), the two pixels that are located at (x, y) and (x', y') will swap their pixel values with each other.

2.1.2. Chebyshev Map

Chebyshev map [12] is a one-dimensional chaotic system that can be formulated, as shown in Equation (2):

$$x_{n+1} = T_a(x_n) = \cos(a \times \arccos x_n), \tag{2}$$

where $x_n \in [-1, 1]$ and $a \in \mathbb{N}$ is again one of the system parameters. For $a \geq 2$, chaotic behavior of Equation (2) holds. The initial value x_0 of the above equation is considered as part of the secret key. In CIES-UBPRPD, a is fixed at 4.

2.2. Image Encryption Algorithm

On the bases of ACM and Chebyshev map, the above-mentioned Chaotic map-based Image Encryption System—CIES-UBPRPD—was proposed by Huang et al. [10], where the most eye-catching property of the algorithm is its plaintext data-dependent encryption process. The encryption process of the original CIES-UBPRPD consists of the following two stages:

2.2.1. Permutation Stage

Step 1. Iterate the Chebyshev map defined in Equation (2) $M \times N + n_0 + 9$ times, discard the first n_0 terms to avoid the harmful effect, and obtain the chaotic sequence x_n, which contains $(M \times N + 9)$ elements.

$$x_n = \{x_0, x_1, \cdots, x_{M \times N+8}\}.$$

Step 2. Generate another sequence x_{nq} according to

$$x_{nq}(i) = (k_1 \otimes k_2 \otimes k_3) \otimes x_n(i) \times 10^{15}, \tag{3}$$

where $i \in \{0, 1, \cdots, 8\}$ and $\otimes$ denotes bitwise XOR operator.

Step 3. Calculate sum_r, sum_g, and sum_b based on the following equations:

$$\begin{aligned} sum_r &= \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} P_R(i, j), \\ sum_g &= \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} P_G(i, j), \\ sum_b &= \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} P_B(i, j), \end{aligned} \tag{4}$$

where P_R, P_G, and P_B represent the Red, Green, and Blue channels of the plain image P, respectively.

Step 4. Calculate the system parameters by using the following equations:

$$\begin{cases} b_r = \mathrm{mod}\big(x_{nq}(0) \otimes sum_r + x_{nq}(1) \otimes sum_g + x_{nq}(2) \otimes sum_b,\ 256\big) \\ a_r = \mathrm{mod}((b_r + 1) \times (k_1 \otimes k_2 \otimes k_3),\ 65536) + 1 \\ b_g = \mathrm{mod}\big(x_{nq}(3) \otimes sum_r + x_{nq}(4) \otimes sum_g + x_{nq}(5) \otimes sum_b,\ 256\big) \\ a_g = \mathrm{mod}\big(\big(b_g + 1\big) \times (k_1 \otimes k_2 \otimes k_3),\ 65536\big) + 1 \\ b_b = \mathrm{mod}\big(x_{nq}(6) \otimes sum_r + x_{nq}(7) \otimes sum_g + x_{nq}(8) \otimes sum_b,\ 256\big) \\ a_b = \mathrm{mod}((b_b + 1) \times (k_1 \otimes k_2 \otimes k_3),\ 65536) + 1, \end{cases} \tag{5}$$

where (b_r, a_r), $\big(b_g, a_g\big)$, and (b_b, a_b) are pairs of parameters used to permute P_R, P_G, and P_B, respectively. Moreover, mod (x, m) denotes the calculation of "x mod m".

Step 5. Permute P_R, P_G, and P_B using the following modified Cat Map with the corresponding parameters:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & a \\ b+1 & a(b+1)+1 \end{bmatrix} \begin{bmatrix} x+1 \\ y+1 \end{bmatrix} \mathrm{mod} \begin{bmatrix} M \\ N \end{bmatrix}, \tag{6}$$

where $x \in \{0, 1, \cdots, M-1\}$ and $y \in \{0, 1, \cdots, N-1\}$. The scanning sequence is started from left to right and from top to bottom. After P_R, P_G and P_B are shuffled, we get the permuted image P^*.

2.2.2. Diffusion Stage

Step 1. Transform P^*_R, P^*_G, and P^*_B, into three on-dimensional (1D) arrays $P^*_{R\_P}$, $P^*_{G\_P}$, and $P^*_{B\_P}$, respectively, by row-major ordering.

Step 2. Calculate the diffusion matrix D according to

$$\mathrm{D(i)} = \mathrm{mod}(\lfloor x_n(i+9) \times (k_1 \otimes k_2 \otimes k_3) \rfloor,\ 256), \tag{7}$$

where $i \in \{0, 1, \cdots, M \times N - 1\}$.

Step 3. Calculate C_{R_P}, C_{G_P}, and C_{B_P} by using the following equations:

$$\begin{cases} C_{R\_P}(0) = (b_r + k_1)\mathrm{mod}256 \\ C_{G\_P}(0) = \big(b_g + k_2\big)\mathrm{mod}256 \\ C_{B\_P}(0) = (b_b + k_3)\mathrm{mod}256 \end{cases} \tag{8}$$

$$\begin{cases} C_{R\_P}(i) = \mathrm{mod}\big(P^*_{R_P}(i) \otimes D(i) + num,\ 256\big) \otimes C_R(i-1) \\ C_{G\_P}(i) = \mathrm{mod}\big(P^*_{G_P}(i) \otimes D(i) + num,\ 256\big) \otimes C_G(i-1) \\ C_{B\_P}(i) = \mathrm{mod}\big(P^*_{B_P}(i) \otimes D(i) + num,\ 256\big) \otimes C_B(i-1) \end{cases} \tag{9}$$

where $num = \big(a_r \times b_r + a_g \times b_g + a_b \times b_b\big) \otimes (k_1 + k_2 + k_3)$, and $i \in \{1, 2, \cdots, M \times N - 1\}$.

Step 4. Transform C_{R_P}, C_{G_P}, and C_{B_P} into three grayscale images with size $M \times N$, and then merge them into one color cipher image C, with size $M \times N \times 3$.

Notice that the permutation processes executed in Step 5 involved parameters sum_r, $sum_{g,}$ and sum_b, which are input (plaint) image data-dependent (cf. Equations (4) and (5)). Notice that, as pre-described in Section 1, the encryption process only includes one round permutation stage and diffusion stage. Conceptually and theoretically, if the encryption process of a cryptosystem is plaintext data-dependent, the associated cryptanalysis is much complicated. Therefore, the corresponding security level of the system is much enhanced, as compared with its data-independent counterpart.

2.3. Image Decryption Algorithm

Similarly, also from [10], the decryption process of the original CIES-UBPRPD consists of the following five steps:

Step 1. Transform C_R, C_G, and C_G into three 1D arrays C_{R_P}, C_{G_P}, and C_{B_P}, respectively, by row-major ordering.

Step 2. Calculate the diffusion matrix $D = \{d_0, d_1, \cdots, d_{M\times N-1}\}$ based on Equation (7).

Step 3. Calculate the system parameters by using the following equations:

$$\begin{cases} b_r = (C_{R\_P}(0) - k_1) \bmod 256 \\ b_g = (C_{G\_P}(0) - k_2) \bmod 256 \\ b_b = (C_{B\_P}(0) - k_3) \bmod 256 \end{cases} \tag{10}$$

$$\begin{aligned} a_r &= \bmod((b_r + 1) \times (k_1 \otimes k_2 \otimes k_3),\ 65536) + 1 \\ a_g &= \bmod\Big(\big(b_g + 1\big) \times (k_1 \otimes k_2 \otimes k_3),\ 65536\Big) + 1 \\ a_b &= \bmod((b_b + 1) \times (k_1 \otimes k_2 \otimes k_3),\ 65536) + 1. \end{aligned} \tag{11}$$

Step 4. Reconstruct $P^*_{R\_P}$, $P^*_{G\_P}$, and $P^*_{B\_P}$ according to

$$\begin{cases} P^*_{R_P}(i) = \bmod\Big(\big(C_{R_P}(i) \otimes C_{R_P}(i-1)\big) - num,\ 256\Big) \otimes D(i) \\ P^*_{G_P}(i) = \bmod\Big(\big(C_{G_P}(i) \otimes C_{G_P}(i-1)\big) - num,\ 256\Big) \otimes D(i) \\ P^*_{B_P}(i) = \bmod\Big(\big(C_{B_P}(i) \otimes C_{B_P}(i-1)\big) - num,\ 256\Big) \otimes D(i), \end{cases} \tag{12}$$

where

$$num = \Big(a_r \times b_r + a_g \times b_g + a_b \otimes b_b\Big) \otimes (k_1 + k_2 + k_3), \tag{13}$$

and $i \in \{1, 2, \cdots, M \times N - 1\}$. Then transform these three arrays into 2D arrays P^*_R, P^*_G, and P^*_B, respectively.

Step 5. Reconstruct P_R, P_G, and P_B by using the Cat Maps defined in Equation (1), but now the scanning sequence is started from right to left and from bottom to top.

3. Cryptanalysis

Cryptanalysis is a must procedure for any cryptosystem to be applied to any real applications. We launched a few analyses on CIES-UBPRPD when we learned it from [10]. This section is organized, as follows. Section 3.1 reports the security weaknesses we found and Section 3.2 presents the attack that we used to break a simplified version of the original CIES-UBPRPD.

3.1. Security Weaknesses

3.1.1. Equivalent Classes in Keyspace

After detailed analyses of CIES-UBPRPD, we found that two secret key groups $key_1 = (x_0, k_1, k_2, k_3, n_0)$ and $key_2 = \big(x'_0, k'_1, k'_2, k'_3, n'_0\big)$ could play indistinguishable roles to each other, in the original cryptosystem, if the following conditions are satisfied:

$$\begin{cases} n_0 = n'_0 \\ x_0 = x'_0 \\ k_1 \otimes k_2 \otimes k_3 = k'_1 \otimes k'_2 \otimes k'_3 \\ k_1 \equiv k'_1 (\bmod 256) \\ k_2 \equiv k'_2 (\bmod 256) \\ k_3 \equiv k'_3 (\bmod 256). \end{cases} \tag{14}$$

For demonstration, here we choose $key_1 = (0.7, 784533, 763092, 777777, 1500)$ and $key_2 = (0.7, 353173, 676820, 307761, 1500)$, which satisfy all of the conditions given in Equation (14). First, we use key_1 to encrypt the benchmark image Lena (left, Figure 1) and obtain the corresponding cipher image (middle, Figure 1). Subsequently, we use key_2 to decrypt the cipher image and obtain the recovered image (right, Figure 1).

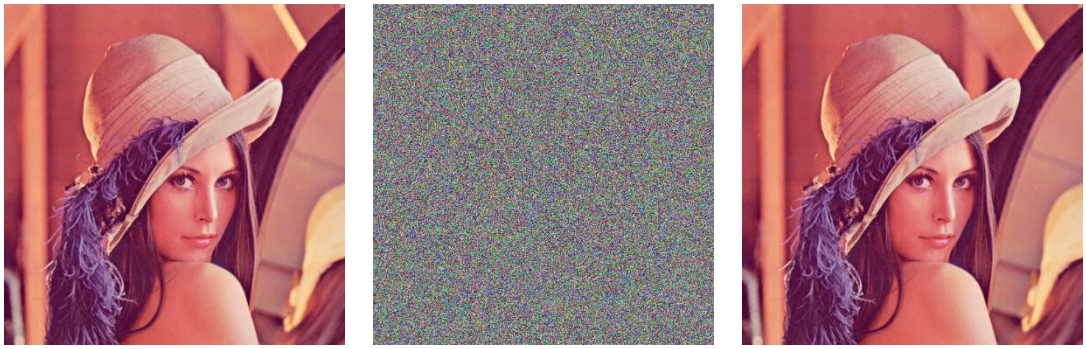


Figure 1. An Example of the Problem of Equivalent Classes in Keyspace.

The above example implies that one can split the set of all keys into equivalence classes that are based on the conditions given in Equation (14), and the effects of keys belonging to the same equivalence classes will be indistinguishable to each other in CIES-UBPRPD. This property shrinks the effective keyspace from $(10^{16} \times (10^{12} - 10^5)^3 \times 1500) \approx 2^{183}$ to $(10^{16} \times 2^8 \times 2^8 \times 2^8 \times 2^{32} \times 1500) \approx 2^{120}$ (since $10^{12} \approx 2^{40}$, we can assume k_1, k_2 and k_3 are of 40-bit long), which is far less than what [10] initially claimed.

3.1.2. Low Sensitivity to the Change of Plaintext

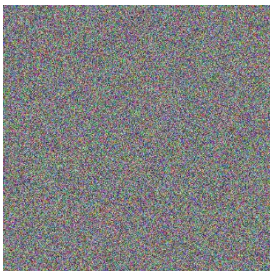
Once the secret key group is chosen, and the summation of pixel values in each channel is given, then the parameters used in the Cat Map are always fixed. Having the same settings means that the permutation mapping will be fixed no matter what the plain image is. Furthermore, there is no cross-channel interaction during permutation and diffusion stages in the original CIES-UBPRPD, which suggests that errors inside one channel will not propagate to the other channels.

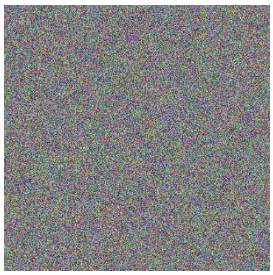
Therefore, we can construct two similar plain images where only their R channels are different, but their sum of R channel remains the same. Additionally the corresponding cipher images of these two images will have no differences in G and B channels. This situation violates the diffusion property that a chaotic-map based cryptosystem is looking for.

For demonstration, we modify the standard Lena image by increasing the first-pixel value by 1 and decreasing the second-pixel value also by 1 in the R channel, and then compare the corresponding cipher image with that of the standard Lena image. The results are shown in Figure 2 and Table 1, the details of two widely used measures, number of pixels change rate (NPCR) and unified average changing intensity (UACI), will be given in Section 5.2.5.

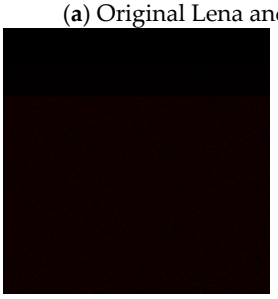
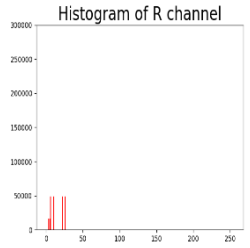
Table 1. Number of pixels change rate (NPCR) and unified average changing intensity (UACI) Values Between the Two Cipher Images Given in Figure 2.

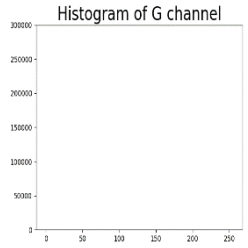
	R	**G**	**B**
NPCR (%)	86.5371	0	0
UACI (%)	4.8464	0	0

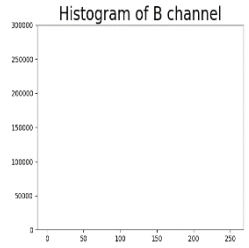





(**a**) Original Lena and Its Cipher Image C_1 (**b**) Modified Lena and Its Cipher Image C_2.





(**c**) The difference Image $|c_1 - c_2|$ and Its Histograms in different Color Channels.

Figure 2. An Illustration Example of the Low Sensitivity to the Change of Plaintext Image.

3.2. Chosen-Plaintext Attack

As stated by Bruce Schneier [13], in academic cryptography, a weakness or a break in a scheme is usually defined quite conservatively: it might require impractical amounts of time, memory, or known plaintexts. It also might require the attacker to be able to do things many real-world attackers cannot. For example, the attacker might need to choose particular plaintexts to be encrypted or even to ask for plaintexts to be encrypted while using several keys related to the secret key. Furthermore, it might only reveal a small amount of information, enough to prove the cryptosystem imperfect, but too little to be useful to real-world attackers. Finally, an attack might only apply to a weakened version of cryptographic tools, like a reduced-round block cipher, as a step towards breaking the full system.

Following the principles of cryptanalysis stated above, we now present a chosen-plaintext attack that works if the size of all images equals 256×256 pixels.

3.2.1. Extraction of the Permutation Matrix

1. Construct a special plain image P, such that

$$P_R = P_G = P_B = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}_{256 \times 256}.$$

2. Encrypt P using CIES-UBPRPD to obtain the corresponding cipher image C. We denote the image after permutation stage as P^* (an intermediate product during the execution of the whole encryption process).

3. Let us use R channel as an example: select two different positions (a, b) and (x, y), where $a, b, x, y \in \{0, 1, \cdots, 255\}$ and construct another special plain image P', such that

$$P'_R = [f(i,j)]_{256\times 256},$$
$$P'_G = P'_B = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}_{256\times 256},$$

where

$$f(i,j) = \begin{cases} 0, \ if \ (i,j) = (a,b) \\ 2, \ if \ (i,j) = (x,y) \\ 1, \ else \end{cases}.$$

4. Encrypt P' to obtain its cipher image C'. From Section 3.1.2, we knew that P and P' share the same parameters used in ACM; thus, they have the same permutation mapping. Let us denote the first position of different values that occurred in C_R and C'_R, following the raster-scan order, as ΔC. It means that $P'^*_R(\Delta C) \neq P^*_R(\Delta C) = 1$, such that $C'_R(\Delta C) \neq C_R(\Delta C)$, which means either $P'_R(a,b) = 0$ or $P'_R(x,y) = 2$ will be moved to the position ΔC after performing the permutation stage. This property reveals some information regarding the permutation behavior of CIES-UBPRPD. We define $((a,b),(x,y),\Delta C)$ as a tuple of constraints.
5. Choose a different (a,b) or (x,y), repeat Step 3 to Step 4 several times and collect all of the produced constraint tuples, and then define the associated collection of constraint tuples as a set S.
6. Construct a 256×256 matrix Z, by setting the chosen positions (a, b) and (x, y), used in Step 2, with different positive integers and all other positions with value 0. For example, assume there are two constraint tuples $((0,0),(0,1),(8,8))$ and $((0,0),(0,2),(6,9))$ in S, we can now set $Z(0,0) = 1$, $Z(0,1) = 2$, $Z(0,2) = 3$, and $Z(i,j) = 0$, $\forall (i,j) \notin \{(0,0),(0,1),(0,2)\}$.
7. Using the brute-force searching algorithm, denoted as Algorithm 1 in the following, to find b_r and $\hat{a_r}$ for all of the above chosen plain images, where $\hat{a_r} = a_r \text{mod} 256$. In the specifically considered case, all of the images are of size $256 \times 256 \times 3$, a_r, and $\hat{a_r}$ are equivalent in the permutation stage. Actually, we do not need to know what a_r exactly is, but only its last eight bits. If Algorithm 1 outputs more than one candidate pair, let us go back to Step 3 to collect more constrained tuples and iterate this step until only one pair is left.
8. Making changes to G and B channels, repeat the procedures listed in Step 3 to Step 7 and obtain b_g, $\hat{a_g}$, b_b, and $\hat{a_b}$, where $\hat{a_g} = a_g \text{mod} 256$ and $\hat{a_b} = a_b \text{mod} 256$.
9. Extract $k_1 \text{mod} 256$, $k_2 \text{mod} 256$, and $k_3 \text{mod} 256$ by using

$$\begin{cases} k_1 \text{mod} 256 = (C_R(0,0) - b_r) \text{mod} 256 \\ k_2 \text{mod} 256 = \left(C_G(0,0) - b_g\right) \text{mod} 256 \\ k_3 \text{mod} 256 = (C_B(0,0) - b_b) \text{mod} 256 \end{cases}.$$

We denote these three values as $\hat{k_1}$, $\hat{k_2}$, and $\hat{k_3}$, respectively, for convenience.

Algorithm 1: Brute-Force Search

Input: matrix Z, set of constrain tuples S
Output: set of parameter pairs r
$r \leftarrow \varnothing$;
for $b_r = 0 \rightarrow 255$ **do**
  for $\hat{a}_r = 0 \rightarrow 255$ **do**
    $Z^* \leftarrow permute(Z, \hat{a}_r, b_r)$;
    if $Check(Z^*, Z, S)$ **then**
      $r \leftarrow r \cup (\hat{a}_r, b_r)$;
return r;

Algorithm 2: $Check(Z^*, Z, S)$

Input: permuted matrix Z^*, original matrix Z, set of constrain tuples S
Output: *True* or *False*
$result \leftarrow true$;
for *every constrain tuple* $((a,b),(x,y),\Delta C) \in S$ **do**
  temp $\leftarrow (Z^*(\Delta C) == Z(a,b)) \vee (Z^*(\Delta C) == Z(x,y))$;
  result $\leftarrow$ result $\wedge$ temp;
return *result*;

3.2.2. Extraction of the Diffusion Matrix

First, we convert above-mentioned C_R into 1D array C_{R_P} by row major ordering. We can now obtain the diffusion matrix D, according to

$$\begin{aligned} D(i) &= \mathrm{mod}((C_{R\_P}(i) \otimes C_{R\_P}(i-1)) - num', 256) \otimes P^*_{R\_P}(i) \\ &= \mathrm{mod}\Big(\big(C_{R_P}(i) \otimes C_{R_P}(i-1)\big) - num', 256\Big) \otimes 1, \end{aligned}$$

where

$$num' = \left(\hat{a_r} \times b_r + \hat{a_g} \times b_g + \hat{a_b} \times b_b\right) \otimes \left(\hat{k_1} + \hat{k_2} + \hat{k_3}\right),$$

$i \in \{1, 2, \cdots, M \times N - 1\}$, and $P^*_{R\_P}$ is a 1D array transformed from P^*_R by row major ordering. Notice that the first element of the diffusion matrix is not used in the cryptosystem, so we can just assign $D(0) = 0$.

So far, we have already extracted $\hat{k_1}$, $\hat{k_2}$, $\hat{k_3}$, and the diffusion matrix D, which are all the required information for decrypting the cipher image.

3.2.3. Recovering the Original Plain Image

Assume that the cipher image is $\overline{C}$ with the size $256 \times 256 \times 3$, and the attacker already extracts $\hat{k_1}$, $\hat{k_2}$, $\hat{k_3}$, and the diffusion matrix D according to the analyses given above. The attacker can decrypt $\overline{C}$ using the following steps:

1. Transform $\overline{C_R}$, $\overline{C_G}$, and $\overline{C_B}$ into three 1D arrays $\overline{C_{R\_P}}$, $\overline{C_{G\_P}}$, and $\overline{C_{B\_P}}$, respectively, in row major ordering.

2. Calculate the required parameters, as follows.

$$\begin{cases} b_r = \left(\overline{C_{R\_P}}(0) - \hat{k_1}\right) \bmod 256 \\ b_g = \left(\overline{C_{G\_P}}(0) - \hat{k_2}\right) \bmod 256 \\ b_b = \left(\overline{C_{B\_P}}(0) - \hat{k_3}\right) \bmod 256 \end{cases}$$

$$\begin{cases} \hat{a_r} = \bmod\left((b_r + 1) \times \left(\hat{k_1} \otimes \hat{k_2} \otimes \hat{k_3}\right), 256\right) + 1 \\ \hat{a_g} = \bmod\left(\left(b_g + 1\right) \times \left(\hat{k_1} \otimes \hat{k_2} \otimes \hat{k_3}\right), 256\right) + 1 \\ \hat{a_b} = \bmod\left((b_b + 1) \times \left(\hat{k_1} \otimes \hat{k_2} \otimes \hat{k_3}\right), 256\right) + 1. \end{cases}$$

3. Reconstruct $\overline{P^*_{R\_P}}$, $\overline{P^*_{G\_P}}$ and $\overline{P^*_{B\_P}}$ according to

$$\begin{cases} \overline{P^*_{R\_P}} = \bmod\left(\left(\overline{C_{R\_P}}(i) \otimes \overline{C_{R\_P}}(i-1)\right) - num', 256\right) \otimes D(i) \\ \overline{P^*_{G\_P}} = \bmod\left(\left(\overline{C_{G\_P}}(i) \otimes \overline{C_{G\_P}}(i-1)\right) - num', 256\right) \otimes D(i) \\ \overline{P^*_{B\_P}} = \bmod\left(\left(\overline{C_{B\_P}}(i) \otimes \overline{C_{B\_P}}(i-1)\right) - num', 256\right) \otimes D(i), \end{cases}$$

where

$$num' = \left(\hat{a_r} \times b_r + \hat{a_g} \times b_g + \hat{a_b} \times b_b\right) \otimes \left(\hat{k_1} + \hat{k_2} + \hat{k_3}\right),$$

and $i \in \{1, 2, \cdots, M \times N - 1\}$. Subsequently, transform these three arrays into two-dimensional (2D) arrays $\overline{P^*_R}$, $\overline{P^*_G}$, and $\overline{P^*_B}$, respectively.

4. Reconstruct $\overline{P_R}$, $\overline{P_G}$, and $\overline{P_B}$ by using ACM, but now the scanning sequence is from right to left and from bottom to top.

We rescale the standard Lena to $256 \times 256 \times 3$ and encrypt it with CIES-UBPRPD, and then crack the cipher image with the proposed chosen-plaintext attack. Figure 3 shows the simulation result.

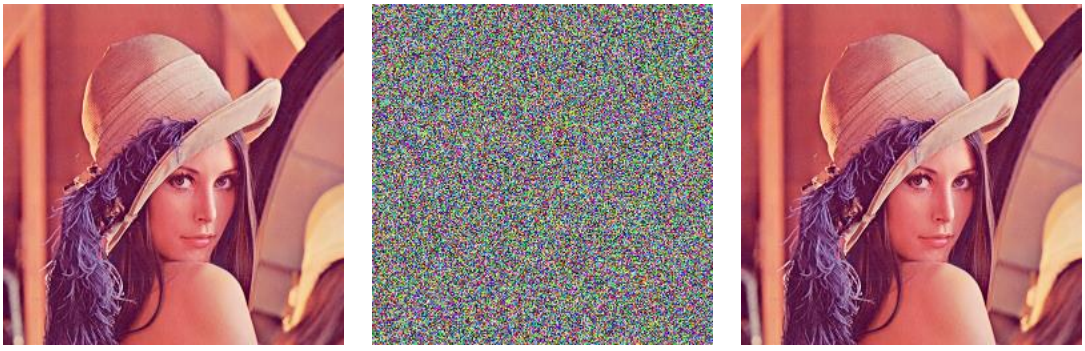


Figure 3. (Left) The Original Plaintext Image; (Middle) the Encrypted (or Ciphertext) Image; (Right) the Recovered Image After Launching the Proposed Chosen-plaintext Attack to the Ciphertext Image.

4. Improved CIES-UBPRPD Algorithm

4.1. The Weaknesses of the Original CIES-UBPRPD

We can crack the original CIES-UBPRPD by the chosen-plaintext attack comes from its following weaknesses:

1. Misuse of the modulo operation. A value's remainder divided by 256 equals to the last eight bits in its binary representation. This operation makes the last eight bits of the value more important than the rest parts. This unequal importance in bits gives us a large number of clues for finding the equivalent classes of parameters in CIES-UBPRPD.

2. The parameters used in ACM are not very sensitive to the initial plain images. As we pointed out in Section 3.1.2, images that have the same sum_r, sum_g, and sum_b share the same system parameters. Thus, it is vulnerable to differential attacks.
3. The diffusion matrix (process) depends only on secret keys, but not the plain images. Once we cracked one cipher image and extracted the diffusion matrix, we can use it to decrypt the other cipher images.

4.2. The Enhanced CIES-UBPRPD

In the enhanced encryption algorithm, instead of the summations of pixel values in each channel, the corresponding SHA-256 hashed values are used as one of the features of a plain image, and this replacement still makes the associated diffusion matrix plaintext-dependent. SHA-256 is a secure cryptographic hash that belongs to SHA-2 families. The hash value served as an external secret key; however, it is dangerous to reuse the same external key when encrypting the same image. Therefore, we add a random number with the precision of 10^{-16} as an additional input to SHA-256 each time that we calculate the hash value. In this way, we can use the hashing output as a one-time key.

4.2.1. Secret Key Formulation

There are six secret keys in the proposed enhanced CIES-UBPRPD algorithm, including the external secret key H that is generated from SHA-256, the initial value x_0 of Chebyshev map, and the four positive integers k_1, k_2, k_3, and n_0, where H is a 256-bit binary number, $x_0 \in (0,1)$, $k_1 \in \left[10^5 \dots 10^{12}\right], k_2 \in [10^5 \dots 10^{12}], k_3 \in [10^5 \dots 10^{12}]$, and $n_0 \in [1000,\ 2500]$. H is then divided into 32 8-bit blocks as $H = h_0, h_1, \cdots, h_{31}$.

4.2.2. Image Encryption Algorithm

Permutation Stage

1. Use x_0 as the initial value and iterate the Chebyshev map $(n_0 + 131)$ times, discard the first n_0 elements to avoid the harmful effect, and obtain the chaotic sequences x_n, which contain 131 elements. That is,
$$x_n = \{x_0, x_1, \cdots, x_{130}\}.$$
2. Generate another sequence x_{nq} by
$$x_{nq}(i) = \lfloor x_i \times k_{i \bmod 3} \times \cos h_{i \bmod 32} \rfloor \bmod 256, \text{ where } i \in \{0, 1, \cdots, 130\}.$$
3. Calculate the parameters by following equations:
$$\begin{aligned} a &= (\sum_{i=0}^{31} h_i \times x_{nq}(i)) \bmod 65536 \\ b &= (\sum_{i=0}^{31} h_i \times x_{nq}(i+32)) \bmod 65536 \\ c &= (\sum_{i=0}^{31} h_i \times x_{nq}(i+64)) \bmod 65536 \\ d &= (\sum_{i=0}^{31} h_i \times x_{nq}(i+96)) \bmod 65536 \end{aligned}$$
4. Permute P using the following three-dimensional (3D) cat map:
$$\begin{bmatrix} i' \\ j' \\ k' \end{bmatrix} = \begin{bmatrix} 1 & a & 0 \\ b & ab+1 & 0 \\ c & d & 1 \end{bmatrix} \begin{bmatrix} i \\ j \\ k \end{bmatrix} \bmod \begin{bmatrix} M \\ N \\ 3 \end{bmatrix}, \quad (15)$$

where $i \in \{0, 1, \cdots, M-1\}$, $j \in \{0, 1, \cdots, N-1\}$, and $k \in \{0, 1, 2\}$ is used to indicate the color channel. The scanning sequence is from R channel to B channel, from left to right and from top to bottom. After this step, we get the permuted image P^*.

Diffusion Stage

1. Transform P^*_R, P^*_G, and P^*_B, into three 1D arrays $P^*_{R\_P}$, $P^*_{G\_P}$ and $P^*_{B\_P}$ by row-major ordering.
2. Use $y_0 = \frac{\cos a + \cos b + \cos c + \cos d + x_0}{5}$ as the new initial value and iterate the Chebyshev map $(3 \times M \times N + n_0)$ times, discard the first n_0 elements and generate another chaotic sequence y_n, which contains $3 \times M \times N$ elements. That is,

$$y_n = \{x_0, x_1, \cdots, x_{3\times M\times N-1}\}.$$

3. Calculate the diffusion matrices D_R, D_G, and D_B according to:

$$\begin{cases} D_R(i) = \lfloor y_i \times (k_2 \otimes k_3) \rfloor \ mod\ 256 \\ D_G(i) = \lfloor y_{i+MN} \times (k_1 \otimes k_3) \rfloor \ mod\ 256 \\ D_B(i) = \lfloor y_{i+2MN} \times (k_1 \otimes k_2) \rfloor \ mod\ 256 \end{cases} \text{where } i \in \{0, 1, \cdots, M \times N - 1\}.$$

4. Calculate C_{R_P}, C_{G_P} and C_{B_P} by using

$$\begin{cases} C_{R\_P}(0) = \mathrm{mod}\Big(P^*_{R\_P}(0) \otimes D_R(0) + num,\ 256\Big) \otimes x_{nq}(128) \\ C_{G\_P}(0) = \mathrm{mod}\Big(P^*_{G\_P}(0) \otimes D_G(0) + num,\ 256\Big) \otimes x_{nq}(129) \\ C_{B\_P}(0) = \mathrm{mod}\Big(P^*_{B\_P}(0) \otimes D_B(0) + num,\ 256\Big) \otimes x_{nq}(130) \end{cases}$$
$$\begin{cases} C_{R\_P}(i) = \mathrm{mod}\Big(P^*_{R\_P}(i) \otimes D_R(i) + num,\ 256\Big) \otimes C_B(i-1) \\ C_{G\_P}(i) = \mathrm{mod}\Big(P^*_{G\_P}(i) \otimes D_G(i) + num,\ 256\Big) \otimes C_R(i) \\ C_{B\_P}(i) = \mathrm{mod}\Big(P^*_{B\_P}(i) \otimes D_B(i) + num,\ 256\Big) \otimes C_G(i), \end{cases}$$

where $num = ((a + b + c + d) \otimes (k_1 + k_2 + k_3))\mathrm{mod}256$ and $i \in \{1, 2, \cdots, M \times N - 1\}$.

5. Transform C_{R_P}, C_{G_P}, and C_{B_P} into three grayscale images with size $M \times N$, and then merge them into color cipher image C with size $M \times N \times 3$.

Image Decryption Algorithm

1. Transform C_R, C_G, and C_G into three 1D arrays C_{R_P}, C_{G_P}, and C_{B_P} respectively, by row-major ordering.
2. Calculate the chaotic sequence x_{nq} in the same way as the encryption process.
3. Calculate the parameters a, b, c, d in the same way as the encryption process.
4. Calculate the diffusion matrices D_R, D_G, and D_B in the same way as the encryption process.
5. Reconstruct $P^*_{R\_P}$, $P^*_{G\_P}$, and $P^*_{B\_P}$ by using

$$\begin{cases} P^*_{R\_P}(0) = \mathrm{mod}\Big(\Big(C_{R\_P}(0) \otimes x_{nq}(128)\Big) - num,\ 256\Big) \otimes D_R(0) \\ P^*_{G\_P}(0) = \mathrm{mod}\Big(\Big(C_{G\_P}(0) \otimes x_{nq}(129)\Big) - num,\ 256\Big) \otimes D_G(0) \\ P^*_{B\_P}(0) = \mathrm{mod}\Big(\Big(C_{B\_P}(0) \otimes x_{nq}(130)\Big) - num,\ 256\Big) \otimes D_B(0) \end{cases}$$
$$\begin{cases} P^*_{R\_P}(i) = \mathrm{mod}((C_{R\_P}(i) \otimes C_{R\_P}(i-1)) - num,\ 256) \otimes D_R(i) \\ P^*_{G\_P}(i) = \mathrm{mod}((C_{G\_P}(i) \otimes C_{G\_P}(i-1)) - num,\ 256) \otimes D_G(i) \\ P^*_{B\_P}(i) = \mathrm{mod}((C_{B\_P}(i) \otimes C_{B\_P}(i-1)) - num,\ 256) \otimes D_B(i) \end{cases},$$

where $num = ((a + b + c + d) \otimes (k_1 + k_2 + k_3))\mathrm{mod}256$ and $i \in \{1, 2, \cdots, M \times N - 1\}$. Subsequently, transform these three arrays into 2D arrays P^*_R, P^*_G and P^*_B, respectively.

6. Reconstruct P_R, P_G, and P_B by using 3D cat map in Equation (15), but now the scanning sequence is from B channel to R channel, from right to left, and from bottom to top.

5. Experimental Results

5.1. Verification of Encryption and Decryption Algorithms

We apply the proposed algorithm to several testing images (all *of size* $512 \times 512 \times 3$) to demonstrate the algorithm's performance. The secret keys are set, as follows: $x_0 = 0.3, k_1 = 111111, k_2 = 222222, k_3 = 333333$, and $n_0 = 1000$. Figure 4 shows the encryption and decryption results. From the results, we can say that the cipher images are noise-like and irrelevant to the plain images.

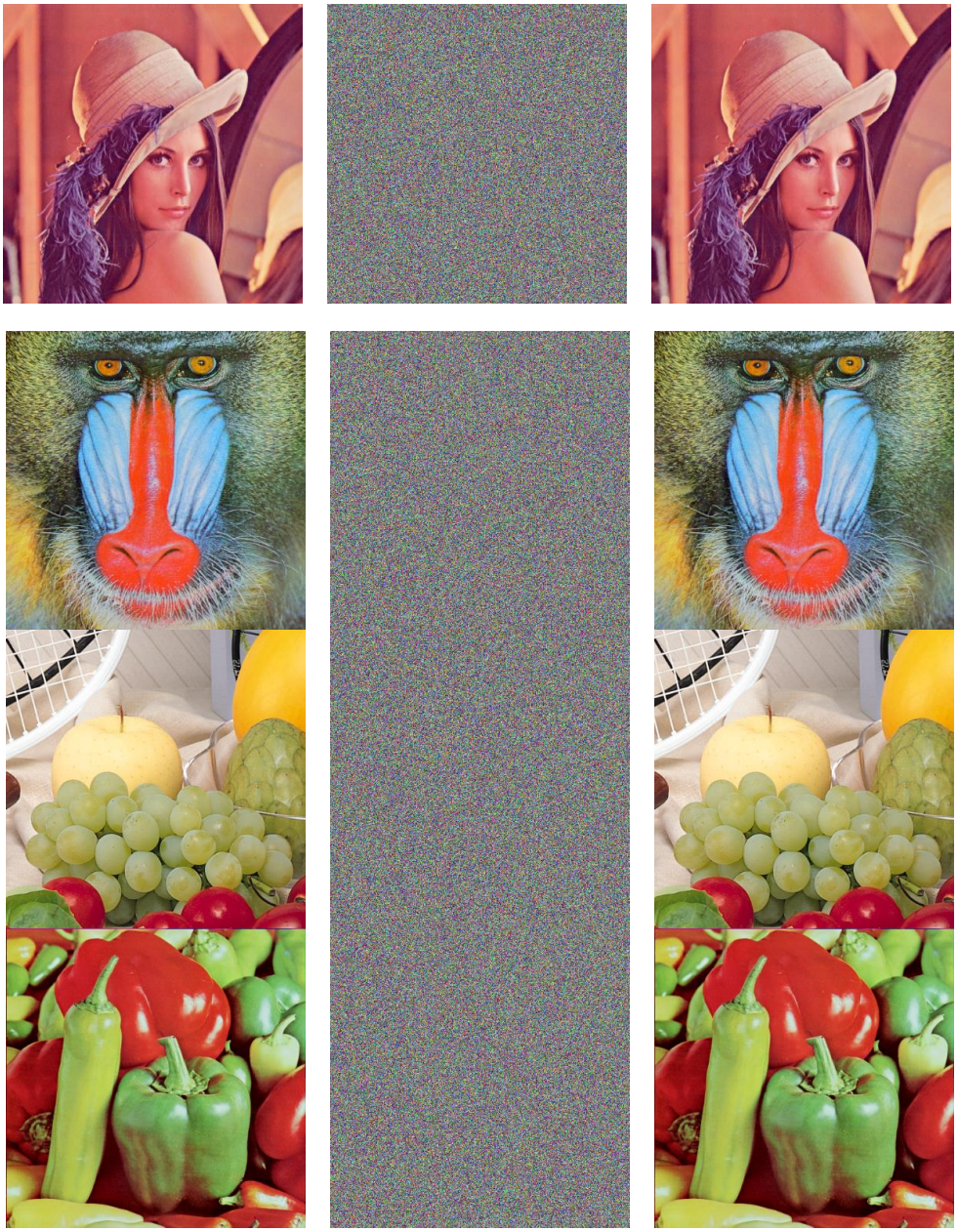


Figure 4. The Testing Results of the Modified Encryption Scheme: (left) the Plaintext Images; (middle) the Ciphertext Images; and, (right) the Recovered Images.

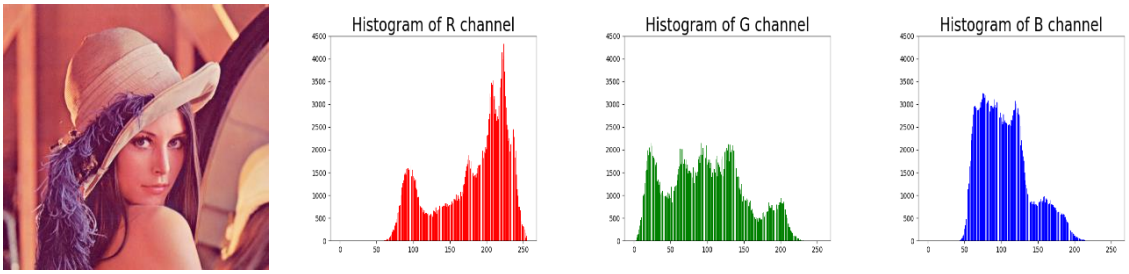
5.2. Security Analyses

5.2.1. KeySpace Analysis

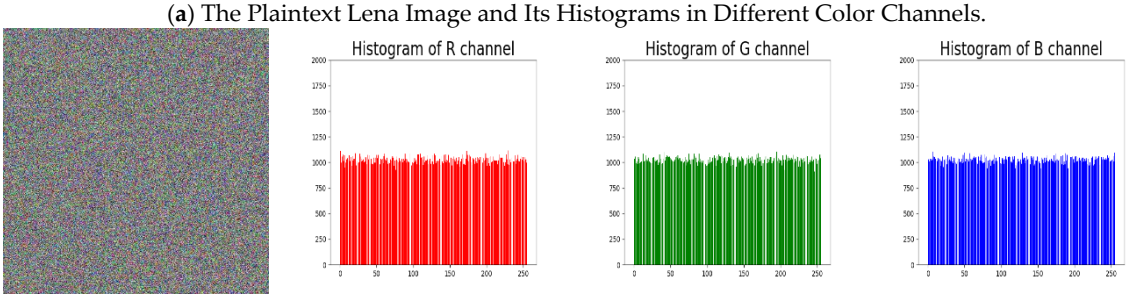
Keyspace is defined as the cardinality of the set of all possible keys. Having a large keyspace is an important factor for ensuring a cryptosystem to resist the brute force attack. The best-known attack complexity of SHA-256 is in the order of 2^{128}. The range of the rest keys are: $x_0 \in (0,1), k_1, k_2, k_3 \in \left[10^5 ... 10^{12}\right]$ and $n_0 \in [1000,\ 2500]$. If x_0 has the precision of 10^{-16}, the keyspace of the proposed scheme can reach to $2^{128} \times 10^{16} \times \left(10^{12} - 10^5\right) \times \left(10^{12} - 10^5\right) \times \left(10^{12} - 10^5\right) \times 1500 \approx 2^{311}$, which is much larger than 2^{100} and enough to make the brute force attack invalid.

5.2.2. Histogram Analysis

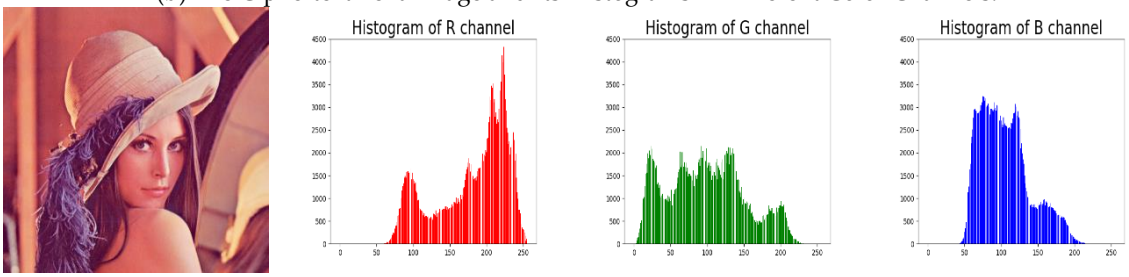
An image's histogram reflects the distribution of its pixels' intensity values, which reveals some statistical information of the image to attackers. To against statistical attacks, the histogram of cipher images generated from a secure encryption system should be flat. The distributions of cipher images' histograms are close to uniformly, indicating that the cipher images are nearly random, and it is rather tough to retrieve any useful statistical information from them, as we can see in Figures 5 and 6.



(**a**) The Plaintext Lena Image and Its Histograms in Different Color Channels.

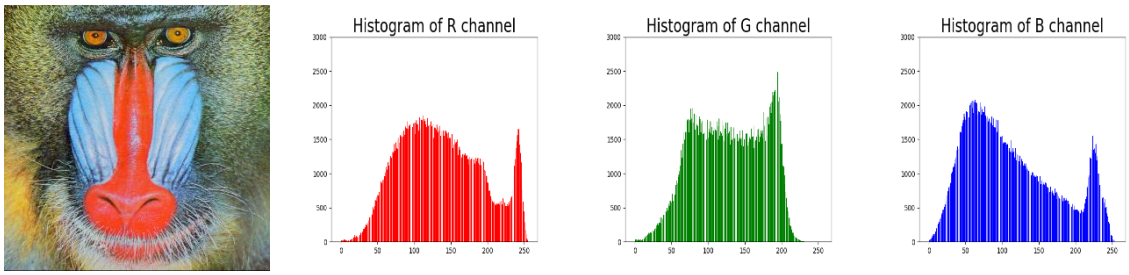


(**b**) The Ciphertext Lena Image and Its Histograms in Different Color Channels.

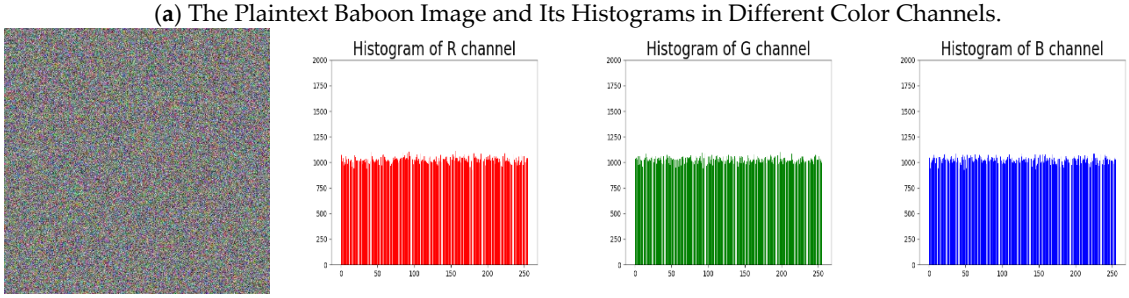


(**c**) The Recovered Lena Image and Its Histograms in Different Color Channels.

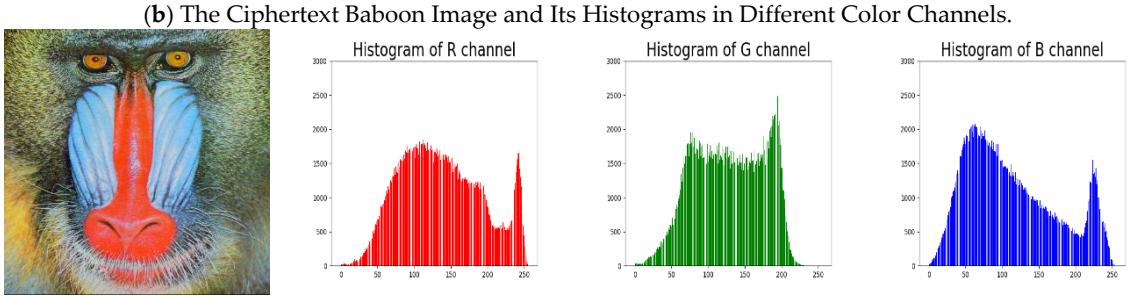
Figure 5. The Original/Encrypted/Recovered Lena Images and the Corresponding Histograms in Different Color Channels.



(**a**) The Plaintext Baboon Image and Its Histograms in Different Color Channels.



(**b**) The Ciphertext Baboon Image and Its Histograms in Different Color Channels.



(**c**) The Recovered Baboon Image and Its Histograms in Different Color Channels.

Figure 6. The Original/Encrypted/Recovered Baboon Images and the Corresponding Histograms in Different Color Channels.

5.2.3. Correlation Analysis

In a natural image (or plain image), two adjacent pixels usually have strong correlation with each other. In contrast, the correlation coefficient of a cipher image should be decreased to zero in order to prevent statistical attacks. We use Equation (16) to measure the correlation of all adjacent pixels at horizontal, vertical, diagonal, and anti-diagonal directions:

$$r_{xy} = \frac{cov(x,y)}{\sqrt{D(x)}\sqrt{D(y)}}, \tag{16}$$

Here,

$$cov(x,y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))(y_i - E(y)),$$
$$D(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))^2,$$
$$E(x) = \frac{1}{N}\sum_{i=1}^{N}x_i,$$

x and y are the two adjacent pixel values and N is the number of pairs of adjacent pixels.

The correlation coefficients of all plain images are close to 1, while those of cipher images are nearly 0, as shown in Table 2. Furthermore, we randomly select 2000 pairs of adjacent pixels at the four specific directions from the R channel of the standard Lena image and its corresponding cipher image, and then plot the scatter diagrams in Figure 7.

Table 2. Correlation Coefficients of the Plain/Cipher Lena Images.

		Plain Image			Cipher Image		
		R	**G**	**B**	**R**	**G**	**B**
Lena	V	0.9893	0.9823	0.9574	0.0015	−0.0017	−0.0023
	H	0.9797	0.9689	0.9325	−0.008	−0.0014	−0.0013
	D	0.9696	0.9554	0.9180	−0.003	−0.0011	−0.0011
	A	0.9777	0.9652	0.9252	−0.006	−0.0005	−0.0002
baboon	V	0.8659	0.7650	0.8808	0.0004	−0.0017	0.006
	H	0.9230	0.8654	0.9073	0.0005	0.0027	0.0019
	D	0.8543	0.7347	0.8398	0.0004	−0.0026	0.0014
	A	0.8518	0.7249	0.8424	−0.0015	−0.0017	0.002

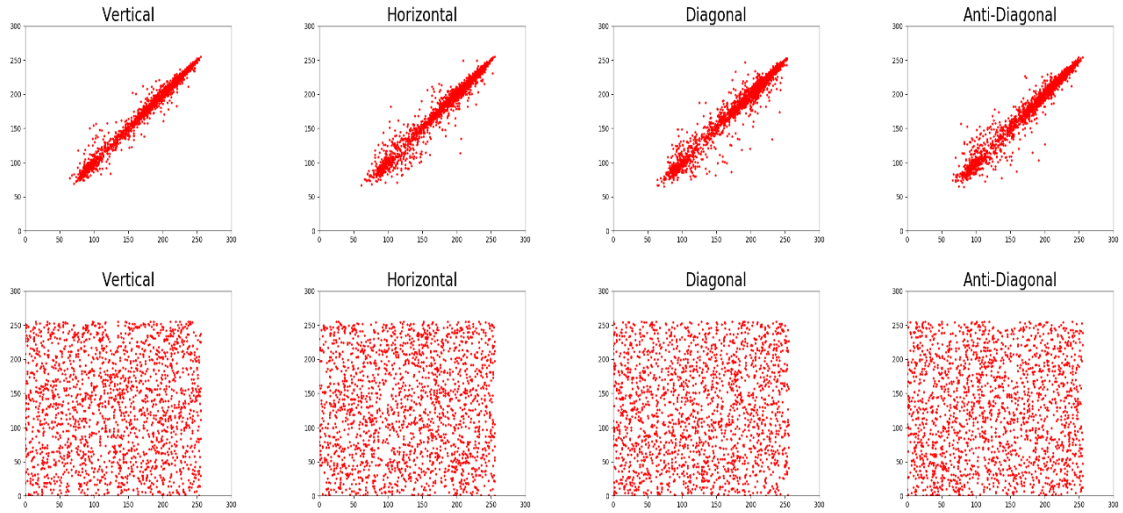
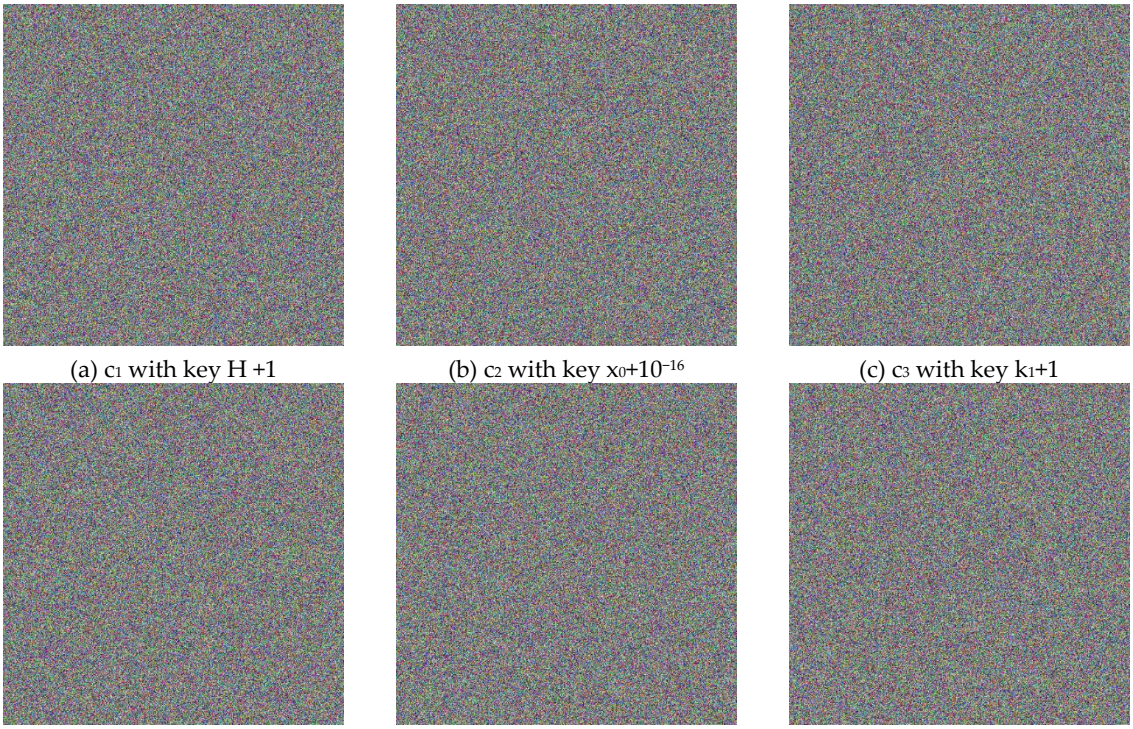


Figure 7. The Scatter Diagrams of Randomly Select 2000 Pairs of Adjacent Pixels at Four Specific Directions from the R Channel of (top) the Plaintext and (bottom) the Ciphertext Lena images.

5.2.4. Key Sensitivity Analysis

A cryptosystem might suffer from differential attacks if cipher images that are generated from different keys are similar. Thus, a secure encryption algorithm should be highly sensitive to all keys, which means even a tiny change in the secret key will lead to a completely different cipher image.

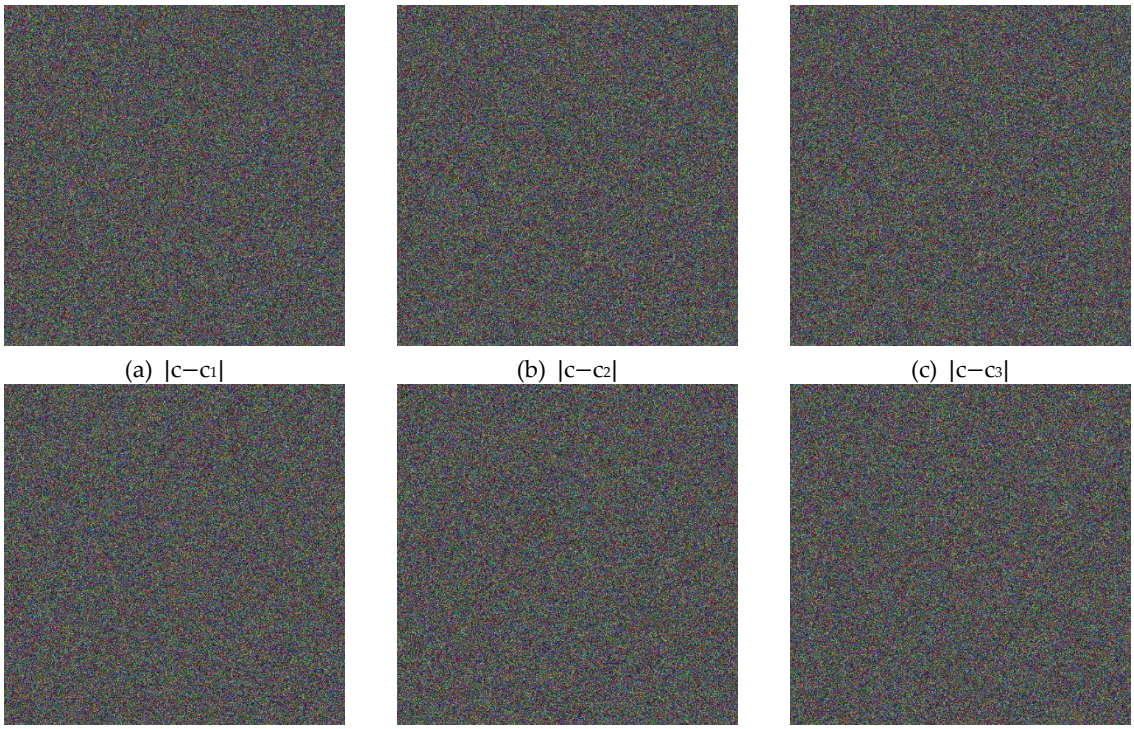
Figure 4 illustrates the plain, cipher, and decrypted Lena images. We change H, x_0, k_1, k_2, k_3, and n_0 by one bit (i.e., 10^{-16} for x_0 and 1 for H, k_1, k_2, k_3 and n_0) to obtain six new cipher images, and the results are shown in Figure 8. Figure 9 shows the differential results between these six cipher images and the cipher image obtained in Figure 4b. Furthermore, we use the six one-bit difference key groups to decrypt Figure 4b, and the decrypting results are shown in Figure 10.



(a) c_1 with key H +1 (b) c_2 with key x_0+10^{-16} (c) c_3 with key k_1+1

(d) c_4 with key k_2+1 (e) c_5 with key k_3+1 (f) c_6 with key n_0+1

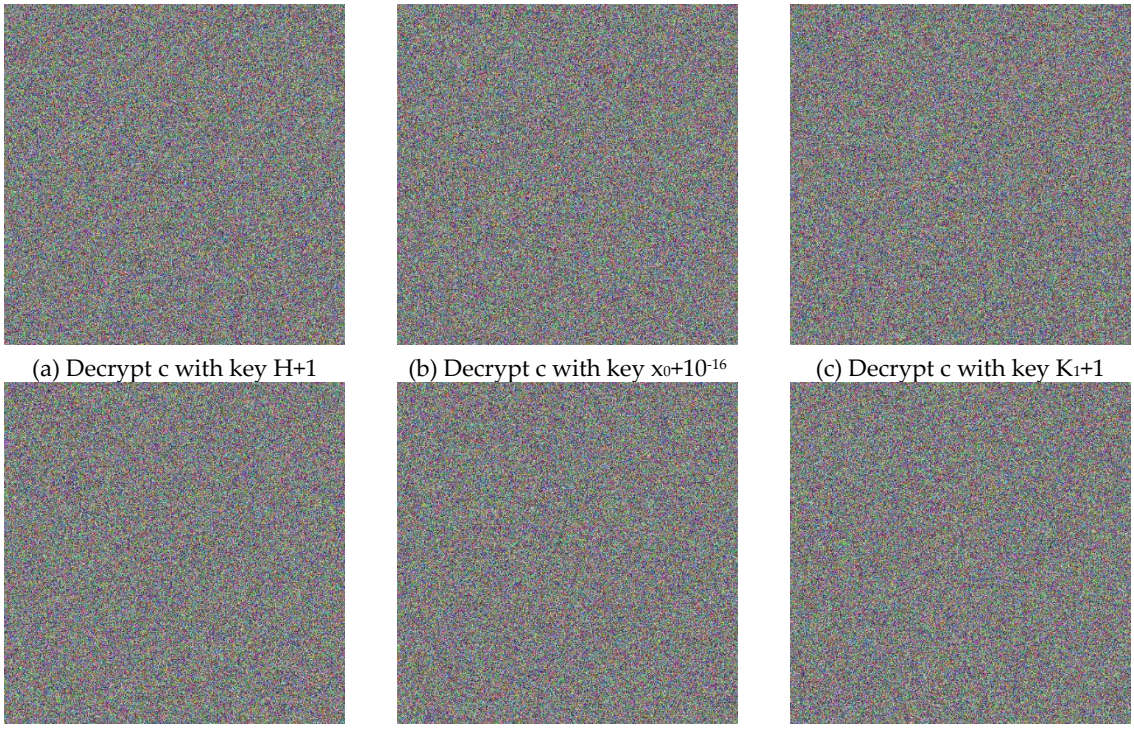
Figure 8. The Six Ciphertext Images Obtained by Changing Single Bit of Different Secret Key Parameters.



(a) $|c-c_1|$ (b) $|c-c_2|$ (c) $|c-c_3|$

(d) $|c-c_4|$ (e) $|c-c_5|$ (f) $|c-c_6|$

Figure 9. The Differential Results Between the Six Cipher Images given in Figure 8 and the Cipher Image has given in Figure 4b. Notice that, without notation confusion, the Original Ciphertext Lena Image is denoted as Image c, here.



(a) Decrypt c with key H+1 (b) Decrypt c with key x_0+10^{-16} (c) Decrypt c with key K_1+1

(d) Decrypt c with key K_2+1 (e) Decrypt c with key K_3+1 (f) Decrypt c with key n_0+1

Figure 10. The Decrypted Images of Figure 4b with the 1-bit Difference Keys, mentioned in Sub-Section 5.2.4.

5.2.5. Plaintext Sensitivity Analysis

Similar to key sensitivity, a secure encryption algorithm should also be very sensitive to plain images. We use NPCR (number of pixels change rate) and UACI (unified average changing intensity) to measure the difference between two cipher images, which are defined, as follows:

$$NPCR = \frac{1}{M \times N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} D(i,j) \times 100\%,$$
$$UACI = \frac{1}{M \times N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \frac{|c_1(i,j) - c_2(i,j)|}{255} \times 100\%,$$

where

$$D(i,j) = \begin{cases} 0, \ if \ c_1(i,j) = c_2(i,j) \\ 1, \ if \ c_1(i,j) \neq c_2(i,j) \end{cases},$$

and c_1 and c_2 are two cipher images. The theoretical values of NPCR and UACI between two different cipher images are 99.6094% and 33.4635%, respectively.

We evaluate plaintext sensitivity, as follows. First, we randomly select x_0, k_1, k_2, k_3, n_0 from the keyspace, encrypt the test image p_1, and with it to get the cipher image c_1. Second, we randomly select a position in p_1, increasing each channel's value by 1 in some places to obtain a modified image p_2. Next, we encrypt p_2 to obtain a cipher image c_2. Subsequently, calculate the NPCR and UACI values between c_1 and c_2. Repeat this process 200 times and list the averaged NPCR and UACI values in Table 3. Both the averaged NPCR and UACI values for all tested images are very close to their theoretical optimal ones, which means the enhanced CIES-UBPRPD algorithm do sensitive to the plain input images, as shown in Table 3.

Table 3. The Plaintext Sensitivity Analyzing Results, Measured in Terms of Averaged NPCR and UACI Values.

	NPCR (%)			UACI (%)		
	R	**G**	**B**	**R**	**G**	**B**
Lena	99.6094	99.6084	99.6096	33.4673	33.4630	33.4662
baboon	99.6075	99.6081	99.6086	33.4606	33.4646	33.4684
fruits	99.6081	99.6103	99.6095	33.4612	33.4620	33.4689
airplane	99.6094	99.6071	99.6101	33.4669	33.4595	33.4564
peppers	99.6099	99.6109	99.6092	33.4649	33.4641	33.4702

5.3. Robustness Analyses

Cipher images can easily be polluted during the transmission through a public channel, as pointed out by one of the anonymous reviewers; therefore, the robustness of a secure image encryption scheme is also an essential performance merit. For testing the robustness of the proposed image encryption scheme, the noise-adding attack and the partial occlusion attack in the ciphertext domain are investigated.

(a) Noise-adding Attack: the ciphered and the de-ciphered images that are presented in Figure 11 are obtained based on the enhanced CIES-UBPRPD, in which the testing ciphertext images have been contaminated by adding with different degrees of salt-and-pepper noises. For performance analysis, we employ the widely used Peak Signal-to-Noise Ratio (PSNR) to evaluate the algorithm's restoring ability. That is

$$\text{PSNR} = 10 \times \log_{10} \frac{255^2}{MSE} (\text{dB}),$$

where,

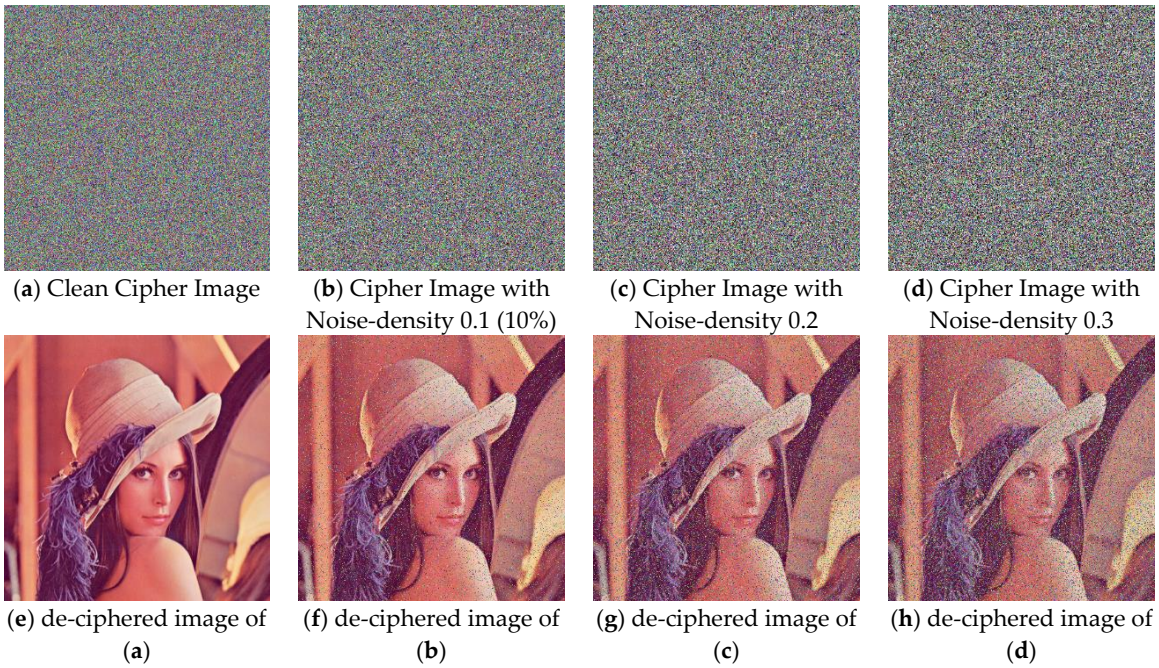
$$\text{MSE} = \frac{1}{3 \times M \times N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \sum_{k=0}^{2} (O(i,j,k) - D(i,j,k))^2,$$

O is the decrypted image obtained from the clean cipher image and D is the decrypted image obtained from the polluted cipher image. Generally, a higher PSNR indicates a better quality or ability of reconstruction. Table 4 shows the PSNRs of the proposed approach against the noise-adding attack.

Table 4. Robustness of the Proposed Approach Against the Noise-adding Attack, in terms of Peak Signal-to-Noise Ratio (PSNR).

Added Noise Density	**PSNR of the De-Ciphered Image**
0.1 (10% of the image frame)	17.6748 (dB)
0.2	14.9988 (dB)
0.3	13.5375 (dB)

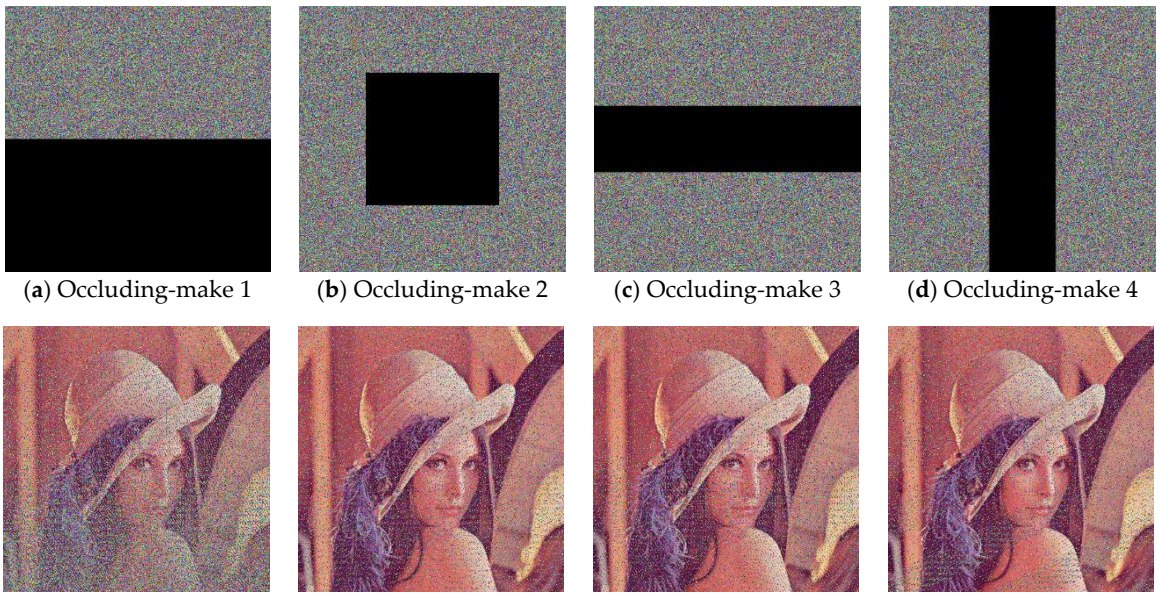
(b) Partial-occlusion Attack. The ciphered and the de-ciphered images that are presented in Figure 12 are obtained based on the enhanced CIES-UBPRPD, in which the testing ciphertext images have been contaminated by occluding some parts of them (i.e., zeroing out those pixel values) that are depicted by the black segments. Similar to the noise-adding attack, PSNR is used to evaluate the proposed approach's restoring ability against this attack, as shown in Table 5.



(**a**) Clean Cipher Image (**b**) Cipher Image with Noise-density 0.1 (10%) (**c**) Cipher Image with Noise-density 0.2 (**d**) Cipher Image with Noise-density 0.3

(**e**) de-ciphered image of (**a**) (**f**) de-ciphered image of (**b**) (**g**) de-ciphered image of (**c**) (**h**) de-ciphered image of (**d**)

Figure 11. Robustness of the Proposed Approach Against the Noise-adding Attack.



(**a**) Occluding-make 1 (**b**) Occluding-make 2 (**c**) Occluding-make 3 (**d**) Occluding-make 4

(**e**) de-ciphered image of (**a**) (**f**) de-ciphered image of (**b**) (**g**) de-ciphered image of (**c**) (**h**) de-ciphered image of (**d**)

Figure 12. Robustness of the Proposed Approach Against the Partial-occlusion Attack.

Table 5. Robustness of the Proposed Approach Against the Partial-occlusion Attack, in terms of PSNR.

Data Occlusion Loss	PSNR of the De-Ciphered Image
Pixel values in the bottom half = 0	11.6397 (dB)
Pixel values in the center square = 0	14.6680 (dB)
Pixel values in center row-rectangle = 0	14.6726 (dB)
Pixel values in center column-rectangle = 0	14.6285 (dB)

From Figures 11 and 12, we can still recognize the de-ciphered images, even if the clean cipher images are contaminated by noise-adding or partial-occlusion attacks. This implies that the robustness of the enhanced CIES-UBPRPD is acceptable to practical image security applications.

6. Discussions and Conclusions

6.1. Discussions

In the past few years, combining both compression and encryption in a single algorithm to reduce the complexity is a new tempting approach for securing data during transmission and storage [14–18]. This new approach aims to extend the functionality of compression algorithms to achieve both compression and encryption simultaneously in a single process without an additional encryption stage. Employing the new combined simultaneous compression-encryption approach highly reduces the required resources for encryption (computational and power resources), according to [15] and [18]. Owing to such an attractive property, lots of works are devoted to this topic [19–23], some of them are also chaotic map based. In [22], we proposed three techniques for enhancing various chaos-based joint compression and encryption (JCAE) schemes. They respectively improved the execution time, compression ratio, and estimation accuracy of three different chaos-based JCAE schemes. However, all of the above-mentioned works are plain image independent. Therefore, for enhancing the security level further, how to design an effective plain Image dependent JCAE scheme is one of our future research directions.

Since steganography can also be utilized to conceal the private information, such as to provide privacy protection of medical images, as pointed out by one of the anonymous reviewers, besides Image Encryption and Decryption, Image Steganography is another worthy of noticing area in the field of Image Security. Interested readers are referred to the following informative writeups, although it is not within the scope of this work [24,25].

6.2. Conclusions

In this paper, we make detailed cryptanalysis on a published chaotic map-based image encryption system, where the encryption process is plaintext Image dependent. We show that some designing weaknesses make the published cryptosystem vulnerable to chosen-plaintext attack, and we then proposed an enhanced algorithm to overcome those weaknesses.

In summary, we use the SHA-256 hash value instead of the sum of each channel as a "plaintext feature", so the enhanced CIES-UBPRPD has higher plaintext sensitivity than its original counterpart. Moreover, the newly proposed encryption process includes the cross-channel interaction, which can resist the Chosen Plaintext Attack that we launched. Since the SHA-256 hash value also serves as an external key, making the improved Keyspace reaches to 2^{311}, which is larger than the effective Keyspace 2^{120} of the original CIES-UBPRPD. Besides the security and the robustness, we also take the execution time into account to provide a full performance baseline for comparing the original and the proposed image encryption algorithms, as suggested by one of the anonymous reviewers. Since the calculation amount of the enhanced CIES-UBPRPD is larger than that of the original CIES-UBPRPD, as shown in Table 6, the execution time of the proposed algorithm will be slightly slower than that of its original counterpart. Our implementation is conducted on Intel Core i7-8700 CPU @ 3.2GHz and 32GB RAM with Windows 10 OS, and written in Python.

Table 6. The Timing Performance Comparison of the Original and the Proposed Approaches.

	Original CIES-UBPRPD	**Enhanced CIES-UBPRPD**
512 × 512 (image size)	3.3557 (s)	3.7944 (s)
256 × 256 (image size)	0.8355 (s)	0.9165 (s)

The value of the proposed algorithm for practical usage in image security is justified, according to the security, the robustness, and the timing performance analyses. Finally, since the currently proposed chosen-plaintext attack can be only effective to all RGB-Colored images of size 256×256 *pixels*, finding an attack that can be applied to more general cases is one of the possible future extensions. We humbly hope that this paper will remind researchers to pay more attention when building their chaotic-based image encryption algorithms in the future.

Author Contributions: Conceptualization, C.-Y.L.; methodology, C.-Y.L.; software, C.-Y.L.; validation, C.-Y.L. and J.-L.W.; formal analysis, C.-Y.L.; investigation, C.-Y.L. and J.-L.W.; data curation, C.-Y.L.; writing—original draft preparation, C.-Y.L. and J.-L.W.; writing—review and editing, C.-Y.L. and J.-L.W.; visualization, C.-Y.L. and J.-L.W.; supervision, J.-L.W.; project administration, J.-L.W.; funding acquisition, J.-L.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Technology, Taiwan, under the grant number: MOST 108-2221-E-002-103-my3.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kumari, M.; Gupta, S.; Sardana, P. A Survey of Image Encryption Algorithms. *3D Res.* **2017**, *8*. [CrossRef]
2. Liu, M.; Zhao, F.; Jiang, X.; Liu, X.; Liu, Y. A Novel Image Encryption Algorithm Based on Plaintext-related Hybrid Modulation Map. *J. Internet Technol.* **2019**, *20*, 2141–2155.
3. Kyamakya, K.; Halang, W.A.; Unger, H.; Chedjou, J.C.; Rulkov, N.C.; Li, Z. (Eds.) Recent Advances in Nonlinear. In *Dynamics and Synchronization: Theory and Applications, Studies in Computational Intelligence 254*; Springer: Berlin, Heidelberg, 2009.
4. Fridrich, J. Symmetric Ciphers Based on Two-Dimensional Chaotic Maps. *Int. J. Bifurc. Chaos* **1998**, *8*, 1259–1284. [CrossRef]
5. Priya, R.; Vidhyapriya, R.; Seifedine, K.; Robertas, D.; Tomas, B. An Image Encryption Scheme Based on Block Scrambling, Modified Zigzag Transformation and Key Generation Using Enhanced Logistic-Tent Map. *Entropy* **2019**, *21*, 656. [CrossRef]
6. Zhu, S.; Wang, G.; Zhu, C. A Secure and Fast Image Encryption Scheme based on Double Chaotic S-Boxes. *Entropy* **2019**, *21*, 790. [CrossRef]
7. Liu, H.; Kadir, A.; Sun, X. Chaos-based fast colour image encryption scheme with true random number keys from environmental noise. *IET Image Process.* **2017**, *11*, 324–332. [CrossRef]
8. Li, Q.; Qian, G. A New Image Encryption Algorithm Based on Chaotic Maps. In Proceedings of the 9th International Conference on Signal Processing Systems, Auckland, New Zealand, 27–30 November 2017; pp. 65–69.
9. Zhu, S.; Zhu, C.; Wang, W. A New Image Encryption Algorithm Based on Chaos and Secure Hash SHA-256. *Entropy* **2018**, *20*, 716. [CrossRef]
10. Huang, L.; Cai, S.; Xiao, M.; Xiong, X. A Simple Chaotic Map-Based Image Encryption System Using Both Plaintext Related Permutation and Diffusion. *Entropy* **2018**, *20*, 535. [CrossRef]
11. Arnold, V.I.; Avez, A. *Problèmes Ergodiques de la Mécanique Classique*; Benjamin: New York, NY, USA, 1967. (In French)
12. He, D.; He, C.; Jiang, L.-G.; Zhu, H.-W.; Hu, G.-R. Chaotic characteristics of a one-dimensional iterative map with infinite collapses. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2001**, *48*, 900–906. [CrossRef]
13. Schneier, B. A self-study course in block-cipher cryptanalysis. *Cryptologia* **2000**, *24*, 18–33. [CrossRef]
14. Wu, C.-P.; Kuo, C.-C. Design of integrated multimedia compression and encryption systems. *IEEE Trans. Multimed.* **2005**, *7*, 828–839. [CrossRef]
15. Grangetto, M.; Magli, E.; Olmo, G. Multimedia Selective Encryption by Means of Randomized Arithmetic Coding. *IEEE Trans. Multimed.* **2006**, *8*, 905–917. [CrossRef]
16. Zhou, J.; Liang, Z.; Chen, Y.; Au, O.C. Security Analysis of Multimedia Encryption Schemes Based on Multiple Huffman Table. *IEEE Signal Process. Lett.* **2007**, *14*, 201–204. [CrossRef]
17. Nagaraj, N.; Vaidya, P.G.; Bhat, K.G. Arithmetic coding as a non-linear dynamical system. *Commun. Nonlinear Sci. Numer. Simul.* **2009**, *14*, 1013–1020. [CrossRef]

18. El-Arsh, H.Y.; Mohasseb, Y.Z. A New Light-Weight JPEG2000 Encryption Technique Based on Arithmetic Coding. In Proceedings of the MILCOM 2013—2013 IEEE Military Communications Conference, Institute of Electrical and Electronics Engineers (IEEE). San Diego, CA, USA, 18–20 November 2013; pp. 1844–1849.
19. Mostafa, M.; Fakhr, M.W. Joint image compression and encryption based on compressed sensing and entropy coding. In Proceedings of the 2017 IEEE 13th International Colloquium on Signal Processing & its Applications (CSPA), Penang, Malaysia, 10–12 March 2017; pp. 129–134. [CrossRef]
20. Guo, L.; Li, J.; Xue, Q. Joint image compression and encryption algorithm based on SPIHT and crossover operator. In Proceedings of the 2017 14th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Institute of Electrical and Electronics Engineers (IEEE), Chengdu, China, 15–17 December 2017; pp. 185–188.
21. Shehata, A.E.R.; El-Arsh, H.Y. Lightweight Joint Compression-Encryption-Authentication-Integrity Framework Based on Arithmetic Coding. *arXiv* **2018**, arXiv:1804.04300.
22. Tsai, C.-J.; Wang, H.-C.; Wu, J.-L. Three Techniques for Enhancing Chaos-Based Joint Compression and Encryption Schemes. *Entropy* **2019**, *21*, 40. [CrossRef]
23. Xie, Y.; Yu, J.; Guo, S.; Ding, Q.; Wang, E. Image Encryption Scheme with Compressed Sensing Based on New Three-Dimensional Chaotic System. *Entropy* **2019**, *21*, 819. [CrossRef]
24. Liao, X.; Guo, S.; Yin, J.; Wang, H.; Li, X.; Sangaiah, A.K. New cubic reference table based image steganography. *Multimed. Tools Appl.* **2017**, *77*, 10033–10050. [CrossRef]
25. Liao, X.; Yin, J.; Guo, S.; Li, X.; Sangaiah, A.K. Medical JPEG image steganography based on preserving inter-block dependencies. *Comput. Electr. Eng.* **2018**, *67*, 320–329. [CrossRef]