On the Cryptanalysis of a Latin Cubes-Based Image Cryptosystem

Rong Huang 1,2,*, Hao Liu 1,2,*, Xiaojuan Liao 3 and Aihua Dong 1,2

1 College of Information Science and Technology, Donghua University, Shanghai 201620, China; dongaihua@dhu.edu.cn
2 Engineering Research Center of Digitized Textile & Apparel Technology, Ministry of Education, Donghua University, Shanghai 201620, China
3 College of Information Science and Technology, Chengdu University of Technology, Chengdu 610059, China; liaoxiaojuan18@cdut.edu.cn
* Correspondence: rong.huang@dhu.edu.cn (R.H.); liuhao@dhu.edu.cn (H.L.)

Abstract: Based on orthogonal Latin cubes, an image cryptosystem with confusion–diffusion–confusion cipher architecture has been proposed recently (Inf. Sci. 2019, 478, 1–14). However, we find that there are four fatal vulnerabilities in this image cryptosystem, which leave open doors for cryptanalysis. In this paper, we propose a reference-validation inference algorithm and design screening-based rules to efficiently break the image cryptosystem. Compared with an existing cryptanalysis algorithm, the proposed method requires fewer pairs of chosen plain-cipher images, and behaves stably since different keys, positions of chosen bits and contents of plain images will not affect the cryptanalysis performance. Experimental results show that our cryptanalysis algorithm only requires \( 3 \sqrt{8 \times H \times W} + 3 \) pairs of chosen plain-cipher images, where \( H \times W \) represents the image’s resolution. Comparative studies demonstrate effectiveness and superiority of the proposed cryptanalysis algorithm.

Keywords: cryptanalysis; chosen-plaintext attack; image cryptosystem; Latin cubes

1. Introduction

One image is worth more than ten thousand words. How to protect image contents from illegal accesses has become a crucial security issue for the practical applications such as virtual meeting, video surveillance or telemedicine, especially when we have entered the era of big data. Cryptography is a cornerstone in the field of information security. Conventional data encryption techniques, e.g., DES (data encryption standard), AES (advanced encryption standard) and IDEA (international data encryption algorithm), are inappropriate for image encryption applications because there are high redundancies and strong correlation among adjacent pixels [1,2]. Permutation-and-diffusion cipher architecture, which alternately shuffles pixel positions and changes pixel values, has the capability to reduce the redundancy and the correlation, thereby playing a central role in many image encryption algorithms [3–6].

Naturally, an image can be encoded as a three-dimensional (3D) bit matrix, in which bits are the smallest elements representing digital information. Some methods [7–14] have extended the permutation-and-diffusion cipher architecture to 3D version, in which the bit-level permutation not only shuffles the bit positions but also changes the pixel values at the same time. Zhu et al. [7] employed Arnold cat map for bit-level permutation and Logistic map for diffusion. Zhang et al. [8] invented a collision-free random bidirectional visiting mechanism by coupling Chen chaotic system with Arnold cat map, and developed a new hybrid 3D permutation rule. In the image cryptosystem [9], multiple chaotic maps were used to control the bit-level row/column-wise permutation. Cai et al. [10] proposed a plaintext-related random-access mechanism that scrambles the 3D bit matrix based on a mixture of three chaotic maps. Gan et al. [11] obtained a random mapping sequence from Chen chaotic system, and designed a tailor-made multilevel quantizer for
generating diffusion matrices. Pak et al. [12] improved the original Logistic map and Sine map by introducing a differential amplification operation, and then built a linear-nonlinear–linear conversion structure geared towards a bit-level image cryptosystem. Xu and Tian [13] constructed a confusion-diffusion-confusion cipher architecture, and grouped three orthogonal Latin cubes together to form a 3D random permutation table. Shahna and Mohamed [14] performed the image permutation at the pixel and bit levels, which combines a key-governed scan pattern with a cyclic shift operation.

Cryptanalysis, which is the opposite of cryptography, focuses on exploring and exposing vulnerabilities in a cryptosystem. Indeed, some of the above works [7–13] have been found to be insecure. Zhang and Wang [15] developed a differential attack that can effectively bypass Zhu’s Arnold cat map [7]. Then, they put forward an improved proposal, in which flipping operations are embedded into the permutation phase, and the connection between plain pixels and keys is established by swapping the order of the permutation and diffusion phases. Later, Wang and Zhao [16] found that Zhang’s improved proposal [15] is still insecure. A reverse differential attack can exactly counteract the order exchange between the permutation and diffusion phases so as to equivalently reconstruct the random mapping sequence. Wu et al. [17] pointed out that the hybrid 3D permutation of Zhang’s image cryptosystem [8] fails to move the bit at the origin, and the diffusion phase merely depends on a weak CBC (cipher block chaining) mode. To break Zhang’s image cryptosystem [8], Li et al. [18] first separated the permutation and diffusion phases by designing a special plain image, and then revealed functionally equivalent keys by using the chosen-plaintext attack. In the same way, Li et al. [19] broke Pak’s work [12] by comparing the differences between several pairs of chosen plain-cipher images. After analyzing the image cryptosystem [9], Liu et al. [20] reported that the correlation information among adjacent plain pixels cannot be erased by the row/column-wise permutation, thereby leaving an open door for the known-plaintext attack. Wen and Yu [21] ascertained that, in the image cryptosystem [9], keys are not associated with the plain images. Exploiting this security flaw, they launched a cracking attack in a divide-and-conquer manner. Recently, Zhang and Yu [22] demonstrated that the process of generating the orthogonal Latin cubes in [13] is independent of the plain images. This allows an adversary to expose the random mapping sequence by using the chosen-plaintext attack.

In this paper, we reinvestigate the security loopholes of Xu’s image cryptosystem [13]. Inspired by the chosen-plaintext attack, we propose an efficient reference-validation inference algorithm and design a screening-based rule to break Xu’s image cryptosystem [13]. Compared with Zhang’s work [22], our cryptanalysis algorithm requires fewer pairs of chosen plain-cipher images, and behaves stably since different keys, positions of chosen bits and contents of plain images will not affect the cryptanalysis performance. Experimental results demonstrate the effectiveness and the superiority of the proposed cryptanalysis algorithm.

The rest of this paper is organized as follows. Section 2 gives a brief introduction to Xu’s image cryptosystem [13], which is the attack target in this study. In Section 3, we first summarize the existing vulnerabilities of [13], and then describe the proposed cryptanalysis algorithm in detail. In Section 4, we conduct extensive experiments and exhibit corresponding results. Last section concludes this paper.

2. Review of Target Image Cryptosystem

In this section, we briefly introduce the TIC (target image cryptosystem) [13]. Overall, it consists of three encryption phases based on Latin cubes, namely pre-permutation, diffusion, and post-permutation.

For better readability, we use bold uppercase symbols (e.g., P) and bold lowercase symbols (e.g., p) to represent cubes and sequences, respectively. Let P(x, y, z) denote the element of P at the coordinate (x, y, z). Let p(n) denote the nth element of p. Non-bold italic font (e.g., n and N) denotes scalars, and Greek letters (e.g., µ) stand for the keys of an
image cryptosystem. Bold calligraphic capital letters (e.g., $\mathcal{N}$) is used to represent sets. Let $\mathbb{Z}$ represent the ring of integers.

Consider an 8-bit plain image with $H \times W$ resolution, where the total number of bits is $8 \times H \times W$. Let $N$ denote the side length of a bit-cube, whose value equals $\sqrt[3]{8} \times H \times W$. For simplicity, Xu and Tian [13] assume that the image size $H$ and $W$ are appropriate values to ensure that the side length $N$ is an integer. Reshape the plain image into the plain bit-cube, denoted by $P$.

In the TIC [13], Logistic map is adopted to generate a chaotic sequence, denoted by $r = [r(0), r(1), \ldots, r(N - 1)]$. The definition of the Logistic map can be expressed as

$$r(n + 1) = \mu \cdot r(n) (1 - r(n)),$$

where $n = 0, 1, \ldots, N - 1$, and $x = r(-1)$ is the initial condition. In Equation (1), $\mu$ is the system parameter. When its value lies in the interval $(3.573815, 4]$, the system exhibits chaotic properties, including ergodicity and high sensitivity to the initial conditions [13]. Figure 1 shows diagrams of bifurcation, Lyapunov exponent, and information entropy of the Logistic map. More details can be found in [23].

![Figure 1](image.png)

**Figure 1.** Chaotic behaviors of the Logistic map: (a) bifurcation diagram; (b) diagram of Lyapunov exponent; (c) diagram of information entropy. Note that when calculating the information entropy, we simply discretize $r(n)$ through the formula $r(n) \cdot 10$.

The chaotic sequence $r$ is first sorted in ascending order, and then the sorted result is used to form a random mapping sequence, denoted by $s = [s(0), s(1), \ldots, s(N - 1)]$. Specifically, $s$ represents the random mapping relations between $r$ and its sorted counterpart. Clearly, $s(n)$ is an integer lying in the interval $[0, N - 1]$. By using the random mapping sequence $s$, three orthogonal Latin cubes, denoted by $L_1$, $L_2$ and $L_3$, are generated as follows

$$\begin{align*}
L_1(x, y, z) &= \alpha^2 \times s(x) + \alpha \times s(y) + s(z), \\
L_2(x, y, z) &= \beta^2 \times s(x) + \beta \times s(y) + s(z), \\
L_3(x, y, z) &= \gamma^2 \times s(x) + \gamma \times s(y) + s(z),
\end{align*}$$

where $L_1(x, y, z)$, $L_2(x, y, z)$ and $L_3(x, y, z)$ are the elements of $L_1$, $L_2$ and $L_3$ at the coordinate $(x, y, z)$, respectively. In Equation (2), the addition operator “$+$” and the multiplication operator “$	imes$” are both defined on the ring $\mathbb{Z}/N\mathbb{Z}$. Note that, in the TIC [13], $N$ is fixed to 128 so that the addition and multiplication operators are originally defined on the finite field $\text{GF}(2^7)$. However, it is too strict to consider a constant side length. In a practical image cryptosystem, $N$ may not be a power of a prime number. Therefore, in this paper, we define the two operators on the ring $\mathbb{Z}/N\mathbb{Z}$ rather than on a finite field.

The three control parameters, namely $\alpha$, $\beta$, and $\gamma$, in Equation (2), must be different nonzero numbers within the ring $\mathbb{Z}/N\mathbb{Z}$. This is a necessary and sufficient condition for the three Latin cubes to be mutually orthogonal. The orthogonality property means that each triple $(L_1(x, y, z), L_2(x, y, z), L_3(x, y, z))$ appears only once after traversing all possible ones, where $x, y$, and $z$ take values from $[0, N - 1]$. The orthogonality property ensures
that all the mapping relations between \((x, y, z)\) and \((L_1(x, y, z), L_2(x, y, z), L_3(x, y, z))\) are one-to-one correspondences without collision, so that the orthogonal Latin cubes can be directly used for permuting a bit-cube. It is worth clarifying here that the initial condition \(\kappa\), the system parameter \(\mu\), and the three control parameters \(\alpha, \beta, \gamma\) collectively serve as keys in the TIC \([13]\).

The first pre-permutation phase shuffles the bit positions of \(P\) based on the three orthogonal Latin cubes. Formally, the pre-permutation phase can be expressed as

\[
U(x, y, z) = P(L_1(x, y, z), L_2(x, y, z), L_3(x, y, z)),
\]

where \(U = [U(x, y, z)|x, y, z = 0, 1, \cdots, N - 1]\) represents the permuted version of \(P\).

The diffusion phase, which is executed after the pre-permutation phase, aims to flip a part of bits in \(U\) under the control of a random bit sequence. In the TIC \([13]\), the random bit sequence is extracted from \(L_1\) through binarization

\[
B(x, y, z) = \begin{cases} 
0, & \text{if } L_1(x, y, z) \geq N/2, \\
1, & \text{otherwise}. 
\end{cases}
\]

In Equation (4), \(N/2\) is the threshold. The binarized bit-cube is denoted by \(\mathbf{B} = [B(x, y, z)|x, y, z = 0, 1, \cdots, N - 1]\). Reshape \(U\) and \(\mathbf{B}\) into two one-dimensional (1D) bit sequences according to the same scanning order. Here, the 3D-to-1D coordinate transformation can be explicitly formulated as

\[
n = x \times N^2 + y \times N + z.
\]

The two resulting 1D bit sequences are denoted by \(\mathbf{u} = [u(0), \cdots, u(n), \cdots, u(N - 1)]\) and \(\mathbf{b} = [b(0), \cdots, b(n), \cdots, b(N - 1)]\), respectively. The diffusion phase is described as

\[
v(n) = u(n) \oplus v(n - 1) \oplus b(n),
\]

where \(v[0, \cdots, v(n), \cdots, v(N - 1)]\) is the diffused bit sequence, and \(v(-1)\) is initialized to 0. In Equation (6), the sign \(\oplus\) represents bit-wise exclusive or (XOR) operator.

In the last post-permutation phase, a CPV (cipher-parity value), denoted by \(t\), is first defined by

\[
t = v(0) \oplus v(1) \oplus \cdots \oplus v(N - 1).
\]

Then, reshape the 1D diffused bit sequence \(v\) into a diffused bit-cube \(\mathbf{V}\). Here, the 1D-to-3D coordinate transformation for each bit position can be explicitly expressed as

\[
\begin{align*}
    x &= n / N^2, \\
y &= n / N \% N, \\
z &= n \% N,
\end{align*}
\]

where the floor sign \(\div\) rounds down to the nearest integer of the number enclosed within the sign, while \(\%\) represents a remainder operator. The three Latin cubes are reused in the following form to permute the diffused bit-cube \(\mathbf{V}\)

\[
\begin{align*}
    C(x, y, z) &= V(L_1(x, y, z), L_2(x, y, z), L_3(x, y, z)), & \text{if } t = 0, \\
    C(x, y, z) &= V(L_3(x, y, z), L_1(x, y, z), L_2(x, y, z)), & \text{otherwise}.
\end{align*}
\]

In Equation (9), \(C = [C(x, y, z)|x, y, z = 0, 1, \cdots, N - 1]\) represents the resulting bit-cube of the post-permutation phase. Finally, reshape \(C\) into an 8-bit cipher image with \(H \times W\) resolution.

Decryption is composed of the inverse encryption operations, which are organized in a reverse order.
3. Cryptanalysis

In this section, we first summarize the existing vulnerabilities of the TIC [13] from four aspects. Then, inspired by the chosen-plaintext attack, we propose an efficient reference-validation inference algorithm to break the random mapping sequence, and design screening-based rules to determine the keys of the orthogonal Latin cubes. In total, our cryptanalysis algorithm requires only $3\sqrt{8} \times H \times W + 3$ pairs of chosen plain-cipher images to break the TIC [13], where $H$ and $W$ represent the image’s height and width, respectively.

3.1. Vulnerability Analysis

Although Xu and Tian [13] conducted various security tests, there still exist four fatal vulnerabilities in their image cryptosystem. First, the process of generating the chaotic sequence $r$ is independent of the plain image, so that an attacker can arbitrarily fabricate desired plain images without influencing $r$. Second, the diffusion phase, as shown in Equation (6), fundamentally inherits from the traditional CBC mode. This means that modifying one bit in the plain image may only affect a small part of the diffused bits. In such circumstance, an attacker can mine useful information, for example the number of unchanged bits, to infer the random mapping sequence $s$. Third, the initial value of the diffusion phase is set to $v(−1) = 0$ without introducing any cryptography mechanism. This flaw somewhat eases the cryptanalysis task. Fourth, the post-permutation phase, as shown in Equation (9), fails to conceal the statistical information of $V$. This enables an attacker to bypass the post-permutation phase and to calculate the equivalent CPV directly from the cipher images.

Kerckhoffs’s principle, which lies at the core of cryptanalysis, sets forth that the security of a cryptosystem relies only on the keys, rather than on the details of the encryption/decryption algorithm. In other words, the encryption/decryption details, e.g., the coordinate transformations between 1D and 3D as shown in Equations (5) and (8), are all open knowledge. In summary, the task of breaking the TIC [13] is equivalent to reconstructing the key-based information, including the random mapping sequence $s$ (which is controlled by the keys $\kappa$ and $\mu$), and the three orthogonal Latin cubes $L_1$, $L_2$, and $L_3$ (which are controlled by the keys $\alpha$, $\beta$, and $\gamma$).

Hereafter, we act as an attacker and use the existing vulnerabilities to break the TIC [13]. In Section 3.2, we ascertain what controls the CPV and propose a reference-validation inference algorithm to reconstruct $s$. In Section 3.3, we turn to design screening-based rules to determine the keys $\alpha$, $\beta$, and $\gamma$.

3.2. Reference-Validation Inference Algorithm

3.2.1. Simplify the Pre-Permutation Phase

As discussed in [22], some special coordinates in the plain bit-cube $P$ can be used to simplify the pre-permutation phase. When substituting $s(x) = s(y) = 0$ into Equation (2), we have $L_1(x, y, z) = L_2(x, y, z) = L_3(x, y, z) = s(z)$. Then, the pre-permutation phase, as shown in Equations (2) and (3), can be rewritten as

$$U(x, y, z) = P(s(z), s(z), s(z)),$$  \hspace{1cm} (10)

in which $s(z)$ must be an integer lying in the interval $[0, N - 1]$. Here, we introduce a new notation $s^{-1}(\cdot)$ to represent the inverse of $s(\cdot)$, and further formulate Equation (10), giving

$$U(s^{-1}(0), s^{-1}(0), s^{-1}(n)) = P(n, n, n),$$  \hspace{1cm} (11)

where the coordinate $(n, n, n)$ is located at the diagonal of $P$. In other words, as long as we visit the diagonal bits of $P$, the original pre-permutation phase can be simplified to Equation (11). This paves the way for inferring the mapping relations between $s^{-1}(n)$ and $n$. As doing so, reconstructing $s$ is trivial since $s^{-1}(\cdot)$ belongs to a bijective mapping.
To this end, the first step is to choose the plain images, which can simplify the pre-permutation phase according to Equation (11). We create \( N \) plain bit-cubes by modifying the diagonal bits of \( P \) in turn. This procedure is described as follows

\[
P'_{n}(x, y, z) = \begin{cases} 1 - P(x, y, z) & \text{if } x = y = z = n, \\ P(x, y, z) & \text{otherwise,} \end{cases}
\]

where \( n = 0, 1, \cdots, N - 1 \). The notation \( P'_{n} = [P'_{n}(x, y, z)|x, y, z = 0, 1, \cdots, N - 1] \) represents the \( n \)th created plain bit-cube. Clearly, the only one different bit between \( P \) and \( P'_{n} \) is located at the coordinate \((n, n, n)\). As discussed above, modifying bit values in \( P \) will not influence the chaotic behaviors of the Logistic map so that all these plain bit-cubes, namely \( P \) and \( P'_{n} \), share the same \( s \). This provides us with the opportunity to break the random mapping sequence \( s \) by the chosen-plaintext attack. For the convenience of presentation, we shall attach a prime superscript on the letter to signify the pre-permutated, the diffused and the post-permutated version of \( P'_{n} \), respectively.

Feeding \( P'_{n} \) into the TIC [13], we can obtain the post-permutated bit-cube \( C'_{n} \). The new CPV, denoted by \( t'_{n} \), can be directly computed from \( C'_{n} \) even without knowing \( V'_{n} \) because the post-permutation phase does not affect any bit values. According to whether \( t'_{n} = t \) or \( t'_{n} \neq t \), the \( N \) pairs of chosen plain-cipher images can be divided into CPV-preserving and CPV-changing groups. The two kinds of groups will be tackled through different strategies, as will be presented below.

### 3.2.2. CPV-Preserving Group

When \( t'_{n} = t \), the bit-cubes \( C \) and \( C'_{n} \) have undergone the same post-permutation phase, as shown in Equation (9), so that we can directly apply the bit-wise XOR operator to them. Calculate a differential bit-cube \( D_{n} = C \oplus C'_{n} \), and count the total number of 1s in \( D_{n} \). Let \( d_{n} \) be the counting result. Since the diffusion phase, as shown in Equation (6), belongs to the weak CBC mode, a modified bit in \( u'_{n} \) will affect the diffused sequence \( v'_{n} \) starting from the landmark position and propagating the influence of the modification along the way forward to the end, as illustrated in Figure 2. Here, the landmark position, denoted by \( l_{p_{n}} \), plays a dual role. On one hand, it corresponds to the flipped bit in \( u'_{n} \) and to the modified bit in \( P'_{n} \) at the coordinate \((n, n, n)\), as indicated by the dotted arrow in Figure 2. Hence, the landmark position \( l_{p_{n}} \) can be represented by

\[
l_{p_{n}} = s^{-1}(0) \times N^{2} + s^{-1}(0) \times N + s^{-1}(n).
\]

Equation (13) is a 3D-to-1D coordinate transformation, which instantiates Equation (5) by using \((s^{-1}(0), s^{-1}(0), s^{-1}(n))\) for \((x, y, z)\). On the other hand, the landmark position can reflect the number of flipped bits in \( v'_{n} \), denoted by \( h_{n} \), taking the following form:

\[
l_{p_{n}} = N^{3} - h_{n},
\]

where \( N^{3} \) is the total number of bits. As illustrated in Figure 2, when \( t'_{n} = t \) holds, the number of flipped bits \( h_{n} \) can be exposed by counting the number of 1s in \( D_{n} \), namely that the equation \( h_{n} = d_{n} \) holds. Substitute for \( l_{p_{n}} \) in Equation (13) using Equation (14) and apply the 1D-to-3D coordinate transformation, giving

\[
\begin{align*}
s^{-1}(0) &= (N^{3} - h_{0}) / N^{2} = (N^{3} - h_{0}) / N\%N, \\
s^{-1}(n) &= (N^{3} - h_{n}) \%N.
\end{align*}
\]

Equation (15) establishes the relationship between \( s^{-1}(n) \) and \( h_{n} \). Based on this relationship, we can readily determine the mapping relations \( s^{-1}(\cdot) \) due to the one-to-one correspondence between \( h_{n} \) and \( n \).
3.2.3. What Controls the CPV

It is worth ascertaining what controls the CPV and how many pairs of chosen plain-
cipher images in the CPV-preserving and CPV-changing groups, respectively. The following
three propositions comprehensively assert that the last term in Equation (13), namely \( s^{-1}(n) \),
controls the CPV.

**Proposition 1.** If the number of flipped bits, namely \( h_n \), is even, then \( t_n' = 1 \). Otherwise \( t_n' \neq 1 \).

**Proof.** Clearly, the flipped bits in \( \mathbf{v}_n' \), as marked by red ink in Figure 2, are the sources of changing the CPV. Hence, in this proof, we only focus on the flipped part that consists of \( h_n \) bits. Suppose that \( h_n = h_n^{(1-0)} + h_n^{(0-1)} \), where \( h_n^{(1-0)} \) (and \( h_n^{(0-1)} \)) denotes the number of 1s (and 0s) being flipped to 0 (and 1) caused by the one bit modification at the landmark position.

If \( h_n \) is even, we will encounter one of the following two cases: (1) both \( h_n^{(1-0)} \) and \( h_n^{(0-1)} \) are even; (2) both \( h_n^{(1-0)} \) and \( h_n^{(0-1)} \) are odd. In both cases, \( h_n^{(1-0)} \) and \( h_n^{(0-1)} \) share the same parity so that flipping \( h_n \) bits will not change the CPV.

On the contrary, if \( h_n \) is odd, the two cases become: (1) \( h_n^{(1-0)} \) is odd while \( h_n^{(0-1)} \) is even; (2) \( h_n^{(1-0)} \) is even while \( h_n^{(0-1)} \) is odd. In both cases, \( h_n^{(1-0)} \) and \( h_n^{(0-1)} \) possess the opposite parity. This means that the number of 1s in the to-be-flipped part of \( \mathbf{v} \) will be changed from an odd integer to an even integer for case (1), and from an even integer to an odd integer for case (2). Consequently, if \( h_n \) is odd, \( t_n' \) is necessarily opposite to \( t \). \( \blacksquare \)

**Proposition 2.** If the landmark position \( l_p_n \) is even, then \( t_n' = 1 \). Otherwise \( t_n' \neq 1 \).

**Proof.** Equation (14) establishes the relationship between \( h_n \) and \( l_p_n' \), in which \( N^3 = W \times H \times 8 \) is even because it is a multiple of 8. Due to this evenness, Equation (14) forces \( h_n \) and \( l_p_n' \) to share the same parity. Consequently, the landmark position \( l_p_n \), just like \( h_n \), controls the CPV. \( \blacksquare \)

**Proposition 3.** If \( s^{-1}(n) \) is even, then \( t_n' = 1 \). Otherwise \( t_n' \neq 1 \).
Proof. Since \( N^3 = W \times H \times 8, N = 2^{(W \times H)}^{1/3} \) and \( N^2 = 4 \cdot (W \times H)^{2/3} \) are multiples of even integers, they must be even. Similarly, the first and second terms on the right-hand side of Equation (13) must be even as well because they take \( N \) and \( N^2 \) as multipliers. The remaining two terms, namely \( l_{p_{n}} \) and \( s^{-1}(n) \), are therefore forced to share the same parity. Consequently, \( s^{-1}(n) \) exactly plays the same role as \( l_{p_{n}} \) and \( h_{n} \) in controlling the CPV. □

Since \( s^{-1}(\cdot) \) must be an integer lying in the interval \([0, N - 1]\), half CPVs \( t'_{n} \) will remain unchanged when \( s^{-1}(n) \) is even, while the remaining half ones are opposite to \( t \) when \( s^{-1}(n) \) is odd. When \( t'_{n} = t \), we determine \( h_{n} \) by counting the number of 1s in \( D_{n} \), and then calculate \( s^{-1}(n) \) by using Equation (15). When \( t'_{n} \neq t \), \( V \) and \( V'_{n} \) will be permuted by different ways, as shown in Equation (9), so that it is meaningless to calculate the bit-wise XOR of \( C \) and \( C'_{n} \). To alleviate this problem, we propose a reference-validation inference algorithm that requires only two additional pairs of chosen plain-cipher images to determine the mapping relations between \( s^{-1}(n) \) and \( n \) for the CPV-changing group. See details in the next three subsections.

For future convenience, we define two index sets \( \mathcal{N}^{=} \) and \( \mathcal{N}^{\neq} \) with the same cardinality \( N/2 \). The former consists precisely of the indices \( n \) for which \( t'_{n} = t \) is true. Members of the latter are the indices \( n \) for which \( t'_{n} \neq t \) is true. Properties of the CPV-preserving and CPV-changing groups are summarized in Table 1 for ease of comparison.

| Table 1. Properties of the CPV (cipher-parity value)-preserving and CPV-changing groups. |
|---------------------------------|------------|-----------|-----------|-------------|
| CPV-preserving group | \( t'_{n} = t \) | even | even | even | \(|\mathcal{N}^{=}| = N/2 \) |
| CPV-changing group | \( t'_{n} \neq t \) | odd | odd | odd | \(|\mathcal{N}^{\neq}| = N/2 \) |

3.2.4. Pair of Reference Plain-Cipher Images

When \( t'_{n} \neq t \), \( D_{n} = C \oplus C'_{n} \), where \( n \in \mathcal{N}^{\neq} \), is no longer informative because the number of 1s in \( D_{n} \) does not reflect the number of flipped bits in \( C'_{n} \). We shall resort to another means to measure \( h_{n} \). The reference-validation inference algorithm fulfills this need through the following three steps:

[step-1]: using a pair of reference plain-cipher images to detect leftmost and rightmost landmark positions for the CPV-changing group (discussed in this subsection);
[step-2]: using a pair of validation plain-cipher images to determine the index \( n \in \mathcal{N}^{\neq} \), whose \( l_{p_{n}} \) corresponds to the leftmost landmark position (discussed in Section 3.2.5);
[step-3]: using the leftmost landmark position to measure \( h_{n} \), where \( n \in \mathcal{N}^{\neq} \) (discussed in Section 3.2.6).

To achieve step-1, we create the reference plain bit-cube by simultaneously modifying three diagonal bits of \( P \)

\[
p_{\text{ref}}(x, y, z) = \begin{cases} 
1 - P(x, y, z) & \text{if } x = y = z = n_{\text{left}}, \\
1 - P(x, y, z) & \text{if } x = y = z = n_{\text{mid}}, \\
1 - P(x, y, z) & \text{if } x = y = z = n_{\text{right}}, \\
P(x, y, z) & \text{otherwise}, 
\end{cases}
\]

where \( P_{\text{ref}} = [p_{\text{ref}}(x, y, z)]x, y, z = 0, 1, \ldots, N - 1 \] denotes the reference plain bit-cube. Hereafter, a letter with the “ref” superscript, such as \( V_{\text{ref}} \) or \( C_{\text{ref}} \), signifies the intermediate encryption result of \( P_{\text{ref}} \). Most importantly, the three indices, namely \( n_{\text{left}}, n_{\text{mid}} \) and \( n_{\text{right}} \), are constrained to satisfy the following two requirements

(i) \( n_{\text{left}} \) and \( n_{\text{right}} \) are selected from \( \mathcal{N}^{\neq} \) while \( n_{\text{mid}} \) comes from \( \mathcal{N}^{=} \);
(ii) the corresponding three landmark positions satisfy the inequality \( l_{p_{n_{\text{left}}}} < l_{p_{n_{\text{mid}}}} < l_{p_{n_{\text{right}}}} \).

The requirement (i) implies that \( P_{\text{ref}} \) provides a bridge between the CPV-preserving and CPV-changing groups. The requirement (ii) ensures that the new CPV, denoted by \( t_{\text{ref}} \),
equals $t$. The reason for this is presented below. Since $l_{p_{\text{left}}} < l_{p_{\text{mid}}} < l_{p_{\text{right}}}$, one bit modification will affect $v_{\text{ref}}$ starting from $l_{p_{\text{left}}}$ and ending at $l_{p_{\text{mid}}}$. The flipped bits appear once again starting from $l_{p_{\text{right}}}$ until the end of the last bit of $v_{\text{ref}}$. As illustrated in Figure 3, the flipped bits (red ink) are separated into two parts in $v_{\text{ref}}$. The number of flipped bits in $v_{\text{ref}}$, denoted by $h_{\text{ref}}$, can be represented by

$$h_{\text{ref}} = (N^3 - l_{p_{\text{left}}}) - (N^3 - l_{p_{\text{mid}}}) + (N^3 - l_{p_{\text{right}}}) = N^3 - l_{p_{\text{left}}} + l_{p_{\text{mid}}} - l_{p_{\text{right}}},$$

(17)

where $N^3$ is an even integer. Moreover, both $l_{p_{\text{left}}}$ and $l_{p_{\text{right}}}$ are odd because the indices $n_{\text{left}}$ and $n_{\text{right}}$ belong to $\mathcal{N}^e$. Conversely, the landmark position $l_{p_{\text{mid}}}$ is even since $n_{\text{mid}} \in \mathcal{N}^e$. As shown in (17), the expression for $h_{\text{ref}}$ contains two even items and two odd items, so that $h_{\text{ref}}$ must be even. Thus, the evenness of $h_{\text{ref}}$ ensures that $t_{\text{ref}} = t$ according to Proposition 1.

![Figure 3. Illustration of the three modified bits in $u_{\text{ref}}$, the flipped bits in $v_{\text{ref}}$, and the three landmark positions. The flipped bit values are expressly marked by red ink.](image-url)

To meet the two requirements, we design the following procedures for selecting the three indices. First, define a new differential bit-cube $D_{m,n} = C_{m} \oplus C_{n}$, where both $m$ and $n$ come from $\mathcal{N}^e$, and we have $t_{m} = t_{n} \neq t$. The number of 1s in $D_{m,n}$, denoted by $d_{m,n}$, is informative in providing the distance between $l_{p_{m}}$ and $l_{p_{n}}$, giving

$$d_{m,n} = |l_{p_{m}} - l_{p_{n}}|.$$  

(18)

 Traverse all possible combinations $\{m, n\}$ and calculate $d_{m,n}$. Search the combination that gives the maximum value in (18). That is

$$m^\#, n^\# = \arg\max_{m,n} d_{m,n},$$

for $m \in \mathcal{N}^e$ and $n \in \mathcal{N}^e$,  

(19)

where we stipulate that $m^\# < n^\#$. Next, traverse all indices $n \in \mathcal{N}^e$ and select the one whose landmark position is the median. That is

$$n^\# = \arg\max_{n} d_{n},$$

for $n \in \mathcal{N}^e$.  

(20)
It is worth mentioning that the traversal operations used above are computationally feasible for $\mathcal{N}^{\pm}$ and $\mathcal{N}^{\neq}$. This is because the cardinality of $\mathcal{N}^{\pm}$ (and $\mathcal{N}^{\neq}$) equals $N/2$, which is a negligible number compared with the size of key space.

Clearly, $n_{\text{mid}}$ is set to $n^{-}$. However, there exist two candidate settings, which, in this paper, are called null and alternative hypotheses, respectively. The null hypothesis is that $n_{\text{left}} = m_{\pm}$ and $n_{\text{right}} = n_{\neq}$. The alternative hypothesis is that $n_{\text{left}} = n_{\pm}$ and $n_{\text{right}} = m_{\neq}$. Which of these two hypotheses is true will be determined by using the pair of validation plain-cipher images, as will be discussed later. At this moment, we break the tie by taking the null hypothesis as a provisional setting.

With these preparations, we detect the leftmost and rightmost landmark positions as follows. First, the equality $t_{\text{ref}} = t$ enables us to define a reference differential bit cube $D^{\text{ref}}$ by applying the bit-wise XOR operator to $C$ and $C^{\text{ref}}$, namely $D^{\text{ref}} = C \oplus C^{\text{ref}}$. The number of 1s in $D^{\text{ref}}$, denoted by $d_{\text{ref}}$, equals $h_{\text{ref}}$. That is

$$d_{\text{ref}} = h_{\text{ref}} = N^{3} - lP_{\text{left}} + lP_{\text{mid}} - lP_{\text{right}}$$

in which the middle landmark position $lP_{\text{mid}}$ is known, whose value has been calculated in the last subsection. Further reformulate Equation (21) as follows

$$lP_{\text{right}} + lP_{\text{left}} = N^{3} + lP_{\text{mid}} - d_{\text{ref}},$$

where all unknowns are gathered on the left-hand side while the terms on the right-hand side are all accessible. Then, the distance between $lP_{\text{right}}$ and $lP_{\text{left}}$ can be calculated according to Equation (18), giving

$$lP_{\text{right}} - lP_{\text{left}} = d'_{\text{left}, \text{right}}.$$  

where the absolute value sign has been removed because $lP_{\text{right}}$ must be greater than $lP_{\text{left}}$. Combining Equations (22) and (23), we can obtain the solutions for $lP_{\text{left}}$ and $lP_{\text{right}}$.

3.2.5. Pair of Validation Plain-Cipher Images

In this subsection, we conduct the hypothesis testing to determine the true values of $n_{\text{left}}$ and $n_{\text{right}}$ from the two candidates $m_{\pm}$ and $n_{\pm}$. The null hypothesis is that $n_{\text{left}} = m_{\pm}$ and $n_{\text{right}} = n_{\neq}$, which has been considered as a provisional setting. The alternative hypothesis is that $n_{\text{left}} = n_{\pm}$ and $n_{\text{right}} = m_{\neq}$. To this end, we create the validation plain bit-cube, denoted by $P^{\text{val}}$, by simultaneously modifying three bits at the diagonal of $P$

$$P^{\text{val}}(x, y, z) = \begin{cases} 
1 - P(x, y, z) & \text{if } x = y = z = \bar{n}_{\text{left}} \\
1 - P(x, y, z) & \text{if } x = y = z = \bar{n}_{\text{mid}} \\
1 - P(x, y, z) & \text{if } x = y = z = \bar{n}_{\text{right}} \\
P(x, y, z) & \text{otherwise}
\end{cases}$$

(24)

where $P^{\text{val}}(x, y, z)$ is the element of $P^{\text{val}}$ at the coordinate $(x, y, z)$. Accordingly, a letter with the “val” superscript, such as $V^{\text{val}}$ or $C^{\text{val}}$, signifies the intermediate encryption result of $P^{\text{val}}$. The indices $\bar{n}_{\text{left}}$ and $\bar{n}_{\text{mid}}$ are set as before, namely that $\bar{n}_{\text{left}} = m_{\pm}$ and $\bar{n}_{\text{mid}} = n_{\pm}$. However, $\bar{n}_{\text{right}}$ is set as follows

$$\bar{n}_{\text{right}} = \bar{n}_{\pm} = \arg\max_{n} d'_{m'_{\pm'}, n'}, \text{ for } n \in \mathcal{N}^{\mp} \backslash \{m_{\pm}, n_{\pm}\}.$$  

(25)

Recall that Equation (19) seeks the two indices $m_{\pm}$ and $n_{\pm}$ that maximize $d'_{m_{\pm}, n_{\pm}}$. Here, Equation (25) leaves $m_{\pm}$ unchanged and seeks a new index $\bar{n}_{\pm}$ so that the distance between $lP_{\bar{n}_{\pm}}$ and $lP_{m_{\pm}}$ reaches the second largest value.

Since the new indices $\bar{n}_{\text{left}}, \bar{n}_{\text{mid}}$ and $\bar{n}_{\text{right}}$ still satisfy the requirements (i) and (ii), $C^{\text{val}}$ and $C^{\text{ref}}$ are constrained to share the same CPV. We have $v_{\text{val}} = t_{\text{ref}} = t$. This allows us to define a differential bit cube $D^{\text{val}} = C \oplus C^{\text{val}}$. Let $d^{\text{val}}$ denote the number of...
1s in $D^{\text{val}}$. As done before, we can get the solutions for $lp_{\text{left}}$ and $lp_{\text{right}}$, where $d^{\text{val}}$ instead of $d^{\text{ref}}$ is used in Equation (22). As illustrated in Figure 4, if $lp_{\text{left}} = lp_{\text{mid}}$ and $lp_{\text{right}} \neq lp_{\text{mid}}$, accept the null hypothesis. Otherwise, accept the alternative hypothesis.

![Figure 4](image.png)

**Figure 4.** Illustration of the hypothesis testing. The solid pink arrows indicate the deterministic settings of $n^{\#}$, while the dotted ones with question marks indicate the hypothesis settings.

### 3.2.6. CPV-Changing Group

From the CPV-changing group, the leftmost landmark position $lp_{\text{left}}$ plays a key role in determining the mapping relations between $s^{-1}(n)$ and $n$. Specifically, we state that the index $n^{\text{left}}$ satisfies the following equation

$$n^{\text{left}} = \arg\min_n lp_n, \text{ for } n \in \mathcal{N}^{-\#}. \quad (26)$$

Equipped with this property, we can remove the absolute value sign in Equation (18), and represent the unknown landmark positions through

$$lp_n = lp_{\text{left}} + d_{n^{\text{left}}, n}^{\#}, \quad (27)$$

where $d_{n^{\text{left}}, n}^{\#}$ representing the number of 1s in $D_{n^{\text{left}}, n}^{\#}$, has been calculated in Equation (19). Traverse each member $n \in \mathcal{N}^{-\#}$, look up $d_{n^{\text{left}}, n}^{\#}$ in turn, and use Equation (27) to get $lp_n$. Doing so exposes all landmark positions for the parity-changing group. Further, exploit Equations (14) and (15) to determine the mapping relations between $s^{-1}(n)$ and $n$, where $n \in \mathcal{N}^{-\#}$. So far, all $s^{-1}(n)$, where $n = 0, 1, \ldots, N - 1$, has been obtained, enabling us to reconstruct the random mapping sequence $s$ even without knowing the keys $\kappa$ and $\mu$.

### 3.3. Screening-Based Rules

The pre-permutation phase is based on the three orthogonal Latin cubes, as shown in Equation (2), which belong essentially to quadratic equations over the ring $\mathbb{Z}/N\mathbb{Z}$. The keys, namely $\alpha$, $\beta$, and $\gamma$, can be regarded as unknown variables from the perspective of cryptanalysis. Under the condition that $s$ has been reconstructed, the three shared factors, namely $s(\alpha)$, $s(\beta)$, and $s(\gamma)$, can be exposed by the chosen-plaintext attack. In other words, $s(\alpha)$, $s(\beta)$, and $s(\gamma)$ are viewed as known variables of the quadratic equations in this subsection. However, there may exist multiple groups of the shared factors due to the uncertainty of the CPV. We design screening-based rules to eliminate the uncertainty, paving the way for breaking $\alpha$, $\beta$, and $\gamma$. 

---

1. **Figure 4** illustrates the hypothesis testing process. The solid pink arrows indicate the deterministic settings of $n^{\#}$, while the dotted ones with question marks indicate the hypothesis settings.
2. The CPV-changing group's leftmost landmark position $lp_{\text{left}}$ is crucial in determining the mapping relations between $s^{-1}(n)$ and $n$.
3. The index $n^{\text{left}}$ satisfies the equation $n^{\text{left}} = \arg\min_n lp_n$, for $n \in \mathcal{N}^{-\#}$.
4. After removing the absolute value sign in Equation (18), the unknown landmark positions can be represented through $lp_n = lp_{\text{left}} + d_{n^{\text{left}}, n}^{\#}$.
5. The screening-based rules are designed to eliminate the uncertainty caused by the CPV.
To this end, we need to create two plain bit-cubes $P''_0$ and $P''_1$ by modifying the bits at the coordinates $(L_1(x_0, y_0, z_0), L_2(x_0, y_0, z_0), L_3(x_0, y_0, z_0))$ and $(L_1(x_1, y_1, z_1), L_2(x_1, y_1, z_1), L_3(x_1, y_1, z_1))$, respectively. The coordinates are substituted into the left-hand side of Equation (2). To effectively solve the quadratic equations, we choose the coordinates obeying the following three conditions [24]. First, $L_1(x_0, y_0, z_0) = L_2(x_0, y_0, z_0) \neq L_3(x_0, y_0, z_0)$ and $L_1(x_1, y_1, z_1) \neq L_2(x_1, y_1, z_1) = L_3(x_1, y_1, z_1)$. Second, $L_2(x_0, y_0, z_0) = L_2(x_1, y_1, z_1)$. Third, all these values are integers taken from the interval $[0, N - 1]$.

Feed $P''_m$ into the TIC [13]. Obtain $C''_m$ and $t''_m$, where $m = 0, 1$. The number of the flipped bits in $C''_m$, denoted by $h''_m$, is the key information to expose $s(x_m), s(y_m)$ and $s(z_m)$. Instantiate Equation (8) by using $h''_m$, giving

$$\begin{align*}
    s^{-1}(x_m) &= h''_m / N^2, \\
    s^{-1}(y_m) &= h''_m / N^{%N}, \\
    s^{-1}(z_m) &= h''_m / N^{%,N},
\end{align*}$$

(28)

all of which can be readily converted into $s(x_m), s(y_m)$ and $s(z_m)$ by using the knowledge of $s$.

However, it is not trivial to obtain $h''_m$. We shall first check whether $t''_m$ equals $t$ or not. If $t''_m = t$, calculate a differential bit-cube $D''_m = C'' \oplus C''_m$, and then count the number of 1s in $D''_m$, denoted by $d''_m$. Similar to the CPV-preserving group, we have $h''_m = d''_m$. If $t''_m \neq t$, we select an index $n$ from $N^\neq$ and have $t''_m = t''_n \neq t$. In this case, $D''_m$ should be defined as $D''_m = C'' \oplus \bigoplus C''_m$, and $d''_m$ represents the distance between $lp_m$ and $lp_n$. Similar to Equation (18), we have

$$d''_m = |lp_m - lp_n|,$$

(29)

where the absolute value sign is necessary. The reason for this is presented as follows. In this subsection, the coordinates no longer lie on the diagonal of $P$, meaning that the pre-permutation phase cannot be simplified. As a result, the statement that $lp_n(\text{or} \; lp_n\text{right})$ is located at the leftmost (rightmost) side is no longer true. For an arbitrarily selected $n \in N^\neq$, its landmark position $lp_n$ may be located at the left or right side of $lp_m$. According to Equation (14), $lp_m$ and $lp_n$ can be expressed as $N^3 - h''_m$ and $N^3 - h_n$, respectively. Based on this expression, we can rewrite Equation (29) in the form

$$h''_m = h_n \pm d''_m,$$

(30)

where both $h_n$ and $d''_m$ are accessible. Equation (30) means that, when $t''_m \neq t$, the calculation of $h''_m$ involves uncertainty, resulting in two possible values. Accordingly, by using Equation (28) and the knowledge of $s$, we may obtain two groups of the shared factors, denoted by $[s^+(x_m), s^+(y_m), s^+(z_m)]$ and $[s^-(x_m), s^-(y_m), s^-(z_m)]$, respectively. Here, the superscripts are intended to signify which of the two operators “+” and “−” is used during the calculation of $h''_m$. Since it is difficult to forecast the CPV at the stage of creating $P''_0$ and $P''_1$, we would have to consider following three cases, and design screening-based rules separately to determine the real solutions of $\alpha, \beta$ and $\gamma$.

The first case, indicated by “case 1” in Figure 5, occurs when $t''_0 = t$ and $t''_1 = t$. In this case, we obtain that $h''_0 = d''_0$ and $h''_1 = d''_1$ without uncertainty, meaning that the shared factors $[s(x_0), s(y_0), s(z_0)]$ and $[s(x_1), s(y_1), s(z_1)]$ are both authentic. Therefore, the quadratic equations can be written as

$$\begin{align*}
    L_1(x_0, y_0, z_0) &= L_2(x_0, y_0, z_0) = \chi_0 \times s(x_0) + \chi_0 \times s(y_0) + s(z_0), \\
    L_2(x_1, y_1, z_1) &= L_3(x_1, y_1, z_1) = \chi_1 \times s(x_1) + \chi_1 \times s(y_1) + s(z_1)
\end{align*}$$

(31)

where $\chi_0$ and $\chi_1$ are used here to represent the unknown variables in Equation (2). Solve Equation (31) over the ring $Z/NZ$ and obtain $\chi_0 = \{\chi_0(0), \chi_0(1)\}$ and $\chi_1 = \{\chi_1(0), \chi_1(1)\}$, each of which must contain two real solutions due to the authenticity of the shared factors [24]. For this case, the screening-based rule is to assign $\beta = \chi_0 \cap \chi_1, \alpha = \chi_0 \setminus \{\beta\}$, and $\gamma = \chi_1 \setminus \{\beta\}$, which directly breaks the keys.
The second case, indicated by “case 2” in Figure 5, occurs when \( t_0'' = t \) but \( t_1'' \neq t \) (or equivalently \( t_0'' \neq t \) but \( t_1'' = t \)). In this case, \( \{s(x_0), s(y_0), s(z_0)\} \) is still authentic but we will obtain \( \{s^+(x_1), s^+(y_1), s^+(z_1)\} \) and \( \{s^-(x_1), s^-(y_1), s^-(z_1)\} \) due to the uncertainty of \( h_1'' \). Hence, the quadratic equations take two possible forms

\[
\begin{align*}
L_1(x_0, y_0, z_0) &= L_2(x_0, y_0, z_0) = \chi_0^2 \times s(x_0) + \chi_0 \times s(y_0) + s(z_0) \\
L_2(x_1, y_1, z_1) &= L_3(x_1, y_1, z_1) = (\chi_1^+)^2 \times s^+(x_1) + \chi_1^+ \times s^+(y_1) + s^+(z_1)
\end{align*}
\]

(32)

\[
\begin{align*}
L_1(x_0, y_0, z_0) &= L_2(x_0, y_0, z_0) = \chi_0^2 \times s(x_0) + \chi_0 \times s(y_0) + s(z_0) \\
L_2(x_1, y_1, z_1) &= L_3(x_1, y_1, z_1) = (\chi_1^-)^2 \times s^-(x_1) + \chi_1^- \times s^-(y_1) + s^-(z_1)
\end{align*}
\]

(33)

Solving (32) yields \( \chi_0 \) and \( \chi_1^+ \). Accordingly, \( \chi_0 \) and \( \chi_1^+ \) are the solutions of Equation (33). Due to its authenticity, \( \chi_0 \) can be used to screen out the real solution \( \chi_1 \) from \( \chi_1^+ \) and \( \chi_1^- \). Specifically, the screening-based rule is that, if \( \chi_0 \cap \chi_1^+ = \emptyset \), then \( \chi_1 = \chi_1^- \); otherwise \( \chi_1 = \chi_1^+ \). In this way, the uncertainty can be eliminated, allowing us to smoothly break the keys as described before.

The third case, indicated by “case 3” in Figure 5, occurs when \( t_0'' \neq t \) and \( t_1'' \neq t \). In this case, both \( h_0'' \) and \( h_1'' \) have two possible values, each of which generates two groups of the shared factors. Hence, the quadratic equations take one of the following four forms

\[
\begin{align*}
L_1(x_0, y_0, z_0) &= L_2(x_0, y_0, z_0) = (\chi_0^+)^2 \times s^+(x_0) + \chi_0^+ \times s^+(y_0) + s^+(z_0) \\
L_2(x_1, y_1, z_1) &= L_3(x_1, y_1, z_1) = (\chi_1^+)^2 \times s^+(x_1) + \chi_1^+ \times s^+(y_1) + s^+(z_1)
\end{align*}
\]

(34)

\[
\begin{align*}
L_1(x_0, y_0, z_0) &= L_2(x_0, y_0, z_0) = (\chi_0^-)^2 \times s^-(x_0) + \chi_0^- \times s^-(y_0) + s^-(z_0) \\
L_2(x_1, y_1, z_1) &= L_3(x_1, y_1, z_1) = (\chi_1^-)^2 \times s^-(x_1) + \chi_1^- \times s^-(y_1) + s^-(z_1)
\end{align*}
\]

(35)

Figure 5. A toy example that illustrates the screening-based rules for the three cases. Suppose that the keys \( \alpha, \beta, \) and \( \gamma \) take values 1, 2, and 3, respectively. The bold dotted boxes in different colors highlight the scope of each case. The gray tick and cross indicate that the group of solutions will be accepted and discarded, respectively.
from which we explicitly obtain four groups of solutions, denoted by \( \{\chi_0^+, \chi_1^-\} \), \( \{\chi_0^-, \chi_1^-\} \), \( \{\chi_0^+, \chi_1^+\} \), and \( \{\chi_0^-, \chi_1^+\} \), respectively. Inspect each group, the screening-based rule makes the following judgement. If the intersection of the two solutions is empty, then the corresponding group will be discarded, otherwise, it must be the authentic one \( \{x_0, \chi_1\} \). For example, if only the second group has a non-empty intersection, namely that \( \chi_0^+ \cap \chi_1^- \neq \emptyset \), then we set \( \chi_0 = \chi_0^+ \) and \( \chi_1 = \chi_1^- \). By doing so, the double uncertainties can be eliminated as well.

Figure 5 shows a toy example that illustrates the screening-based rules for the three cases. Assume that \( n = 5 \) is selected from \( \mathcal{N}^\times \), and the pair of chosen plain-cipher bit-cubes, namely \( P_0' \) and \( C_0' \), plays a role when \( t_m' \neq t \). Note that, since the shared factors may be incorrect, there exists empty solution, such as \( \chi_1^+ \) as shown in this example. The empty solution can be discarded directly.

### 3.4. Performance Analysis

In total, the proposed cryptanalysis algorithm requires \( 2 \times \sqrt[3]{H \times W} + 3 \) pairs of chosen plain-cipher images to break the TIC [13]. Compared with Zhang’s work [22] that requires \( 2.5 \times \sqrt[3]{H \times W} + 6 \) pairs, our method is more efficient especially when the image has a high resolution. This merit is particularly useful when the number of admissible accesses to a TIC is limited.

To reconstruct \( s \), the proposed reference-validation inference algorithm requires \( 2 \times \sqrt[3]{H \times W} + 1 \) pairs of chosen plain-cipher bit-cubes. First, we need to create \( n = 1, 2, \ldots, N - 1 \) plain bit-cubes \( P_n' \) using Equation (12), where \( n = 1, 2, \ldots, N - 1 \). In practice, the plain bit-cube \( P_0' \) is needless because \( s^{-1}(0) \) can be immediately derived from Equation (15) regardless of the value of \( n \). Second, to tackle the CPV-changing group, we create a reference plain bit-cube \( P_{\text{ref}} \) using Equation (16) and a validation counterpart \( P_{\text{val}}' \) using Equation (24).

To break the keys \( a, \beta \), and \( c \), Zhang’s work [22] requires two, four or six pairs of chosen plain-cipher images, respectively, to deal with the three cases described in the last section. Note that when evaluating the performance of a cryptanalysis algorithm, we always consider the worst bound that need the most computational consumptions. From this perspective, we state that six pairs are needed in Zhang’s work [22]. By contrast, the designed screening-based rules can eliminate the uncertainty in \( h_m' \) by fully mining the exclusion and intersection relationships between the group-wise solutions. Therefore, in our work, only two pairs of chosen plain-cipher images are sufficient to break the keys \( a, \beta \), and \( c \), whichever case we encounter in practice.

Furthermore, the number of necessary pairs of chosen plain-cipher images can be treated as a constant with respect to the keys, the positions of chosen bits or the contents of plain images. This merit allows an attacker to accurately estimate the computational consumptions before launching the attacks. The experimental results in Section 4.3 will corroborate the above claims.

### 4. Experimental Results

In this section, we conduct simulation experiments and comparative studies to demonstrate the effectiveness and the superiority of the proposed cryptanalysis algorithm. The first experiment aims to exhibit the cryptanalysis results for five standard grayscale images. The second experiment tests the cryptanalysis performance for a camera-based natural scene image, showing the prospects for practical applications. The third experiment is devoted to the comparative studies, which demonstrates that our cryptanalysis algorithm requires fewer pairs of chosen plain-cipher images than Zhang’s work [22]. Following the
setting in [13], we specify that the default keys are $\kappa = 0.12345678912341$, $\mu = 3.99999$, $\alpha = 1$, $\beta = 2$, and $\gamma = 3$, respectively. Unless explicitly stated, the TIC [13] is governed by the default keys. All experiments are implemented on a desktop computer with a 2.90 GHz Intel i7-10700 central processing unit, 16.00 GB memory. The programming environment for simulations is Matlab R2017a installed on the Window 10 operation system.

4.1. Results of the Cryptanalysis Algorithm

Figure 6 shows the results for the first experiment. The first and second columns display the plain images and the corresponding cipher images obtained from the TIC [13], respectively. Five plain images are “Lena”, “Baboon”, “Testpat”, “Wedge”, and “Black”, all of which have the size of $512 \times 512 \times 1$ (grayscale) and the side length $N$ being equal to $\sqrt{512 \times 512 \times 8} = 128$. In this experiment, we first perform the proposed cryptanalysis algorithm for reconstructing $s$ and breaking the keys $\alpha$, $\beta$, and $\gamma$. Then, for each cipher image, perform the decryption algorithm of [13] governed by the broken information. In the third column of Figure 6, we provide the intermediate results, in which the cipher images are partially decrypted to counteract the post-permutation and diffusion phases (with the exception of the pre-permutation phase). The rightmost column shows the completely decrypted images. We see that these completely decrypted images are exactly the same as the plain images in the first column without any visual loss. For “Black”, however, the partial decryption can entirely reveal the visual information because “Black” is immune to the pre-permutation phase. As expected, the third column of Figure 6e is an all zero-valued image, which is the same as the plain image. This provides us with a new perspective to verify the correctness of our cryptanalysis algorithm on the basis of the intermediate results.

In the upper panel of each image, we append some auxiliary information, intended to provide qualitative and quantitative indicators for monitoring the progress and validating the correctness of our cryptanalysis algorithm. The auxiliary information includes CSBPs (cross-sectional bit-planes), LBP (local binary pattern) histograms, and corresponding entropy values calculated from the LBP histograms.

For qualitative comparisons, we select the front CSBP $P(0, :, :)$, the middle CSBP $P(N/2 - 1, :, :)$, and the back CSBP $P(N - 1, :, :)$ out of a given bit-cube $P$, where the colon operator returns a regularly-spaced vector $[0, 1, \ldots, N - 1]$. Intuitively, the CSBP can reflect the bit correlation in a given bit-cube. Observing the CSBPs of the plain images, we see that there exist regular LBPs, especially for “Testpat” and “Wedge”. In contrast, the cipher images’ CSBPs consist of pseudorandom LBPs, meaning that the regularity has been eliminated by the TIC [13]. Most importantly, by comparative observations, we find that the same regular LBPs reappear for the completely decrypted images, thereby verifying that the proposed cryptanalysis algorithm is capable of breaking the TIC [13].

In order to better characterize the regularity, we define a histogram that reflects the probability distribution of occurrence of eight LBPs in a CSBP. As shown in Figure 7, the horizontal axis of the histogram lists the eight LBPs, where three adjacent bits with different binary combinations are viewed as patterns. The vertical axis corresponds to the probability values. For clarity, we omit the captions of the axes when displaying the LBP histograms in the upper panel. Furthermore, to quantificationally measure the regularity, we calculate the entropy value from each LBP histogram. By comparison, we find that cipher images possess flatter LBP distributions and greater entropy values. This supports the statement that the LBPs for the cipher images have lower regularity. The LBP histograms in the fourth column of Figure 6 share the same shapes as those in the first column. Also, we obtain equal entropy values. These results demonstrate that the TIC [13] has been successfully broken by the proposed cryptanalysis algorithm.
Figure 6. Cont.
Figure 6. Simulation results for the first experiment. From left to right, the four columns display plain images, cipher images, partially decrypted images, and completely decrypted images, respectively. Some auxiliary information is shown in the upper panel of each image. See the texts for details.

Figure 7. An example of the LBP (local binary pattern) histogram. This is an enlarged version derived from the upper panel of the plain image “Lena”.

Moreover, we conduct correlation analysis. The correlation coefficient can be viewed as a numerical indicator reflecting the consistency between the plain image and the completely...
decrypted image. In this experiment, we randomly select 10,000 pairs of adjacent pixels in horizontal, vertical, and diagonal directions from each image, and calculate the correlation coefficient \( r_{pq} \) as follows

\[
 r_{pq} = \frac{\sum_{l=1}^{10^4} (p_l - \mathbb{E}(p)) \times (q_l - \mathbb{E}(q))}{\sqrt{\sum_{l=1}^{10^4} (p_l - \mathbb{E}(p))^2 \sum_{l=1}^{10^4} (q_l - \mathbb{E}(q))^2}},
\]

where \( p_l \) and \( q_l \) consists of the \( l \)th pair of adjacent pixels, and \( \mathbb{E}(p) \) stands for the expectation of \( p = \{p_1, p_2, \cdots, p_{10000}\} \). The correlation coefficient \( r_{pq} \) lies in the interval \([-1, 1]\), and both 1 and \(-1\) indicate the highest correlation while 0 no correlation. Particularly, we stipulate that \( r_{pq} = \text{NaN} \) when \( p_l = q_l = c \), where \( c \) is a constant, holds for all values of \( l \). Table 2 lists the quantitative results. We find that the plain image and the completely decrypted image share the same correlation coefficients. For “Black”, the correlation coefficient of the partially decrypted image equals NaN because counteracting the post-permutation and diffusion phases is sufficient to recover the zero-valued pixels, giving \( p_l = q_l = 0 \). These results also demonstrate the correctness of the proposed cryptanalysis algorithm.

Table 2. The numerical results for the correlation analysis.

| IDs  | Directions | Plain Image | Cipher Image Partially | Decrypted Image Partially | Completely |
|------|------------|-------------|------------------------|---------------------------|------------|
| Lena | horizontal | 0.9740      | −0.0004                | −0.0057                   | 0.9740     |
|      | vertical   | 0.9863      | −0.0025                | 0.0069                    | 0.9863     |
|      | diagonal   | 0.9612      | −0.0041                | −0.0030                   | 0.9612     |
| Baboon | horizontal  | 0.9334      | 0.0027                 | 0.0182                    | 0.9334     |
|      | vertical   | 0.9102      | 0.0145                 | −0.0116                   | 0.9102     |
|      | diagonal   | 0.8635      | 0.0011                 | −0.0114                   | 0.8635     |
| Testpat | horizontal | 0.7395      | 0.0060                 | −0.0152                   | 0.7395     |
|      | vertical   | 0.7654      | −0.0107                | 0.0127                    | 0.7654     |
|      | diagonal   | 0.7320      | 0.0200                 | −0.0200                   | 0.7320     |
| Wedge | horizontal | 0.9973      | 0.0003                 | −0.0144                   | 0.9973     |
|      | vertical   | 0.9998      | −0.0107                | −0.0229                   | 0.9998     |
|      | diagonal   | 0.9971      | −0.0018                | 0.0051                    | 0.9971     |
| Black | horizontal | NaN         | 0.0030                 | NaN                       | NaN        |
|      | vertical   | NaN         | −0.0022                | NaN                       | NaN        |
|      | diagonal   | NaN         | 0.0039                 | NaN                       | NaN        |

4.2. Efficiency of the Cryptanalysis Algorithm

In this experiment, the goal is to demonstrate the effectiveness of our cryptanalysis algorithm under a real-life application scenario. To this end, we take a landscape photograph of our university campus as the plain image. It has three color channels with the spatial resolution of \( 1024 \times 2048 \). Under different settings, the TIC [13] yields three cipher images, as shown in Figure 8b,d,f, respectively. In the first setting, the three channels of the plain image are separately encrypted with the same default keys. In the second setting, we introduce a tiny change into the default keys, and separately encrypt the three channels. In the third setting, the TIC [13] is governed by three different keys, and encrypts the three channels in turn. In our computing environment, the proposed cryptanalysis algorithm takes 86.43 s, 87.15 s and 267.58 s (about 89.19 s for each channel, on average), respectively, to complete the cryptanalysis task under the three settings. Figure 8c,e,g exhibit the cryptanalysis results, in which the decrypted images are the same as the plain image.
efficiency of the proposed cryptanalysis algorithm is relatively stable. The time-consuming
of 1024 \times 2048; (b,c) results of the first setting; (d,e) results of the second setting; (f,g) results of the third setting.

We summarize following two points from the above experimental results. First, the proposed cryptanalysis algorithm is of high efficiency for the camera-based natural scene image, showing the feasibility of deployment in some practical system. Second, the efficiency of the proposed cryptanalysis algorithm is relatively stable. The time-consuming data tell us that different keys have almost no effect on the efficiency.

4.3. Comparative Studies

Zhang’s work [22] also focuses on attacking the same TIC [13]. Compared with this work [22], the proposed cryptanalysis algorithm requires fewer pairs of chosen plain-cipher images. We conduct two comparative studies as follows.

In the first comparative study, we count how many pairs of chosen plain-cipher images are necessary to break the TIC [13]. Plain images are the same as those used in Figure 6. The experimental protocol consists of the following steps. Randomly set two coordinates \((L(x_l, y_l, z_l), L(x_l, y_l, z_l), L(x_l, y_l, z_l))\), where \(k = 0, 1\), according to the conditions mentioned in Section 3.3. Perform Zhang’s cryptanalysis algorithm [22] and the proposed one, respectively. Record the number of necessary pairs of chosen plain-cipher images. For each plain image, we repeat the above steps ten times, and finally calculate the average number of necessary pairs. Moreover, we consider the default keys and a new set of keys, intended to examine whether the performance of the cryptanalysis algorithms is sensitive to keys or not. The new keys are \(\kappa = 0.12345678912343, \mu = 3.99998, \alpha = 2, \beta = 1,\) and \(\gamma = 4\). Numerical results are listed in Table 3.
Table 3. The numerical results for the first comparative study. The better results are highlighted by underlines.

| IDs   | Using Default Keys | Using New Keys |
|-------|--------------------|----------------|
|       | Zhang’s Work [22] | Ours           | Zhang’s Work [22] | Ours           |
| Lena  | 164.0              | 131.0          | 164.0              | 131.0          |
| Baboon| 163.2              | 131.0          | 163.4              | 131.0          |
| Testpat| 163.6              | 131.0          | 163.6              | 131.0          |
| Wedge | 164.8              | 131.0          | 164.6              | 131.0          |
| Black | 163.6              | 131.0          | 164.0              | 131.0          |

For the plain images with the size of $512 \times 512$, the proposed cryptanalysis algorithm only requires 131 pairs of chosen plain-cipher images. By contrast, on average, 163.88 pairs are necessary for Zhang’s work [22]. Furthermore, as we see in Table 3, the numerical results in the columns titled by “Ours” are all the same. This demonstrates that different keys, positions of chosen bits and contents of plain images will not affect the performance of our cryptanalysis algorithm. Thus, only given the images’ resolutions, our cryptanalysis algorithm allows an attacker to pre-estimate the computational consumptions more accurately before launching the attacks.

In the second comparative study, we aim to verify that the proposed cryptanalysis algorithm performs much more efficient when dealing with larger images. The experimental protocol consists of the following steps. Resize the plain images to $1024 \times 2048$, $2048 \times 3456$, and $4096 \times 4096$, respectively, using bicubic interpolation. Accordingly, the enlarged plain bit-cubes have the side length $N = 256, 384, \text{ and } 512$, respectively. For each cipher image, perform Zhang’s cryptanalysis algorithm [22] and the proposed one, respectively. Record the number of necessary pairs of chosen plain-cipher images. For the five plain images in Figure 6, we calculate the average number of necessary pairs. For comparison, the numerical results are plotted in a bar chart, as shown in Figure 9. Regardless of the resolutions, the proposed cryptanalysis algorithm consistently outperforms Zhang’s work [22], and the superiority becomes more obvious for larger images.

![Figure 9. The second comparative study’s results. Lower height of a bar indicates “better”.](image)

5. Conclusions

In this paper, we investigate Xu’s image cryptosystem [13], and summarize security loopholes from four aspects. On this basis, we propose a reference-validation inference algorithm and design screening-based rules to efficiently break Xu’s image cryptosystem [13]. Compared with an existing work [22], our cryptanalysis algorithm requires fewer pairs of chosen plain-cipher images. Only $\sqrt{8 \times H \times W} + 3$ pairs, where $H \times W$ represents the image’s resolution, are sufficient to complete the cryptanalysis task. Moreover, the performance of the proposed cryptanalysis algorithm is highly stable since different keys, positions of chosen bits and contents of the plain images will not influence the number
of necessary pairs. This merit enables an attacker to well pre-estimate and allocate the computational consumptions before launching the attacks.

**Author Contributions:** Methodology, R.H. and X.L.; project administration, R.H. and X.L.; funding acquisition, R.H. and X.L.; software, H.L.; writing, R.H. and H.L.; validation, H.L. and A.D. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded in part by the Fundamental Research Funds for the Central Universities (17D110408), the National Natural Science Foundation of China (62001099, 61806171), the National Key Research and Development Program of China (2019YFC1521300).

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

**References**

1. Lian, S.G. *Multimedia Content Encryption Techniques and Applications*, 1st ed.; CRC Press: Boca Raton, FL, USA, 2008; pp. 1–6.
2. Grangetto, M.; Magli, E.; Olmo, G. Multimedia selective encryption by means of randomized arithmetic coding. *IEEE Trans. Multimed.* 2006, 8, 905–917. [CrossRef]
3. Kaur, M.; Kurmar, V. A comprehensive review on image encryption techniques. *Arch. Comput. Method Eng.* 2020, 27, 15–43. [CrossRef]
4. Chai, X.L.; Gan, Z.H.; Yuan, K.; Chen, Y.R.; Liu, X.X. A novel image encryption scheme based on DNA sequence operations and chaotic systems. *Neural Comput. Appl.* 2019, 31, 219–237. [CrossRef]
5. Song, W.; Zheng, Y.; Fu, C.; Shan, P.F. A novel batch image encryption algorithm using parallel computing. *Inf. Sci.* 2020, 518, 211–224. [CrossRef]
6. Liu, L.D.; Jiang, D.H.; An, T.Y.; Guan, Y.F. A plaintext-related dynamical image encryption algorithm based on permutation-combination-diffusion architecture. *IEEE Access* 2020, 8, 62785–62799. [CrossRef]
7. Zhu, Z.L.; Zhang, W.; Wong, K.W.; Yu, H. A chaos-based symmetric image encryption scheme using a bit-level permutation. *Inf. Sci.* 2011, 181, 1171–1186. [CrossRef]
8. Zhang, W.; Yu, H.; Zhao, Y.L.; Zhu, Z.L. Image encryption based on three-dimensional bit matrix permutation. *Signal Process.* 2016, 118, 36–50. [CrossRef]
9. Shafique, A.; Shahid, J. Novel image encryption cryptosystem based on binary bit planes extraction and multiple chaotic maps. *Eur. Phys. J. Plus* 2018, 133, 331. [CrossRef]
10. Cai, S.T.; Huang, L.Q.; Chen, X.S.; Xiong, X.M. A symmetric plaintext-related color image encryption system based on bit permutation. *Entropy* 2018, 20, 282. [CrossRef] [PubMed]
11. Gan, Z.H.; Chai, X.L.; Han, D.J.; Chen, Y.R. A chaotic image encryption algorithm based on 3-D bit-plane permutation. *Neural Comput. Appl.* 2019, 31, 7111–7130. [CrossRef]
12. Pak, C.; An, K.; Jang, P.; Kim, J.; Kim, S. A novel bit-level color image encryption using improved 1D chaotic map. *Multimed. Tools Appl.* 2019, 78, 12027–12042. [CrossRef]
13. Xu, M.; Tian, Z.H. A novel image cipher based on 3D bit matrix and latin cubes. *Inf. Sci.* 2019, 478, 1–14. [CrossRef]
14. Shahnah, K.U.; Mohamed, A. A novel image encryption scheme using both pixel level and bit level permutation with chaotic map. *Appl. Soft Comput.* 2020, 90, 106162.
15. Zhang, Y.Q.; Wang, X.Y. Analysis and improvement of a chaos-based symmetric image encryption scheme using a bit-level permutation. *Nonlinear Dyn.* 2014, 77, 687–698. [CrossRef]
16. Wang, X.Y.; Zhao, H.Y. Cracking and improvement of an image encryption algorithm based on bit-level permutation and chaotic system. *IEEE Access* 2019, 7, 112836–112847.
17. Wu, J.H.; Liao, X.F.; Yang, B. Cryptanalysis and enhancements of image encryption based on three-dimensional bit matrix permutation. *Signal Process.* 2018, 142, 292–300. [CrossRef]
18. Li, M.; Guo, Y.Z.; Huang, J.; Li, Y. Cryptanalysis of a chaotic image encryption scheme based on permutation-diffusion structure. *Signal Process. Image Commun.* 2018, 62, 164–172. [CrossRef]
19. Li, M.; Wang, P.C.; Liu, Y.F.; Fan, H.J. Cryptanalysis of a novel bit-level color image encryption using improved 1D chaotic map. *IEEE Access* 2019, 7, 145798–145806. [CrossRef]
20. Liu, Y.; Qin, Z.; Wu, J.H. Cryptanalysis and enhancement of an image encryption scheme based on bit-plane extraction and multiple chaotic maps. *IEEE Access* 2019, 7, 74070–74080. [CrossRef]
21. Wen, H.P.; Yu, S.M. Cryptanalysis of an image encryption cryptosystem based on binary bit planes extraction and multiple chaotic maps. *Eur. Phys. J. Plus* 2019, 134, 337. [CrossRef]
22. Zhang, Z.Q.; Yu, S.M. On the security of a Latin-bit cube-based image chaotic encryption algorithm. *Entropy* 2019, 21, 888. [CrossRef]
23. Sprott, J.C. *Chaos and Time-Series Analysis*; Oxford University Press: Oxford, UK, 2003; pp. 17–189.
24. Berlekamp, E.; Rumsey, H.; Solomon, G. On the solution of algebraic equations over finite fields. *Inf. Comput.* 1967, 10, 553–564. [CrossRef]