

Article

Reversible Data Hiding in Encrypted Images Based on an Adaptive Recognition Strategy for Blocks

Zhi Pang ¹, Han Li ¹, Zhaolin Xiao ¹ and Liansheng Sui ^{1,2,*}¹ School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China² Shaanxi Key Laboratory for Network Computing and Security Technology, Xi'an 710048, China

* Correspondence: suiliansheng@xaut.edu.cn

Abstract: As the rapid development of third-party storage and homomorphic encryption have profoundly stimulated the desire for secure communication, reversible data hiding in encrypted images has received widespread attention, since it allows lossless data conveying and perfect image recovery. In order to obtain secure reversible data hiding with high embedding capacity, a novel block embedding method is proposed, based on an adaptive recognition strategy for combined blocks in the binary image, with which the adjacent identical blocks can be integrated into a combination to reserve more spare bits for data accommodation. Furthermore, a fully reversible data hiding method for grayscale images in the encryption domain is designed. The secret data is hidden into lower bit-planes of the image while the original bits of those embedded lower pixels are recorded into the vacated space of higher bit-planes. The original image can be reconstructed flawlessly as well as the secret data being extracted without errors. To reinforce security, the original image and the secret data are encrypted and scrambled based on sequences generated with the high-dimension chaotic system. Due to its high sensitivity of initial values, the performance such as security and robustness is guaranteed. By comparing the PSNR value of the marked decrypted image and evaluating the quality of the extracted secret image, experimental results demonstrate that the proposed method can obtain higher embedding capacity, achieving 0.2700bpp–0.3924bpp increment over the state-of-the-art methods, and recover the marked decrypted image with high visual symmetry/quality, and efficiently resist against potential attacks, such as the histogram analysis, differential, brute-force, JPEG attacks, and so on.

Citation: Pang, Z.; Li, H.; Xiao, Z.; Sui, L. Reversible Data Hiding in Encrypted Images Based on an Adaptive Recognition Strategy for Blocks. *Symmetry* **2023**, *15*, 524. <https://doi.org/10.3390/sym15020524>

Academic Editors: Dumitru Baleanu and Christos Volos

Received: 11 January 2023

Revised: 27 January 2023

Accepted: 9 February 2023

Published: 15 February 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: reversible data hiding; adaptive recognition strategy; Huffman coding

1. Introduction

Thanks to the wide application of cloud servers and computing, traditional digital media has played an essential role in information transmission in the past few decades. However, some characteristics of the transmission medium, such as high redundancy, public nature of the transmission channel, and low replication cost, may cause it to be vulnerable to malicious attacks. As an efficient means of secure communication, data hiding techniques conceal secret data into digital media in an imperceptible way while most of the original content remains the same and the hidden data can be retrieved flawlessly [1–3]. Early data hiding algorithms have been known to permanently damage the host image and the irreversible distortions are intolerable in the fields with extremely high requirements for the accuracy of images, such as medical diagnosis, military drawings, and judicial appraisal [4–7]. Based on this, reversible data hiding (RDH) is proposed for the reversibility of host image, i.e., the host image must be completely recovered to its original form after data extraction [8]. In previous RDH schemes, lossless compression [9–12] was extensively studied to vacate spare room in the original image, but the compression of redundant space and data embedding often left obvious distortions in the image,

resulting in degraded visual quality. To address this issue, data hiding techniques based on histogram shifting [13–17] and difference expansion [18–22] have been developed and matured as two productive branches of RDH. In general, histogram shifting embeds the secret data into the pixels with peak value of the image histogram, while difference expansion uses the difference in pixel pairs to record additional bits. These RDH methods have achieved satisfactory capacity versus distortion performance, but are only applicable for data hiding in the plaintext domain.

Nowadays, with the tremendous rise in popularity of privacy protection and cloud storage, it is desired that original content is encrypted before uploading when being transmitted to service providers. Therefore, the reversible data hiding in encrypted images (RDHEI) emerges in the public view as a solution for simultaneously protecting the conveyed data and original image [23]. Depending on whether spare space reservation is performed after or before the image encryption, the RDHEI methods are mainly classified into two categories: vacating room after encryption (VRAE) [24–33], and reserving room before encryption (RRBE) [34–39]. In VRAE methods, the content owner only encrypts the original image before transmission, and the secret data embedding is executed in the following stage by the data hider. For instance, a novel RDH method [23] is first proposed for encrypted images by analyzing the local standard deviation of the marked encrypted images to extract the embedded data when decrypting. A redundant space transfer strategy is described to maintain most elements of redundancy in the original image after encryption [29]. Similarly, another redundancy transfer scheme is proposed from the most significant bits to the least significant bits [30]. For the study in [33], the quad-tree partition and the integer wavelet transform techniques are employed to implement high-capacity data hiding. As another typical type of RDHEI, the RRBE-based methods have a relatively high embedding rate, because the image encryption may destroy the structural correlation of the original image and maximize the entropy of the encrypted image. For example, the least significant bits of some pixels are embedded into others before image encryption in the traditional RDH algorithm [34], and therefore, the recorded least significant bits can be used to accommodate secret data and vacate ample spare space in the original image. A novel patch-level sparse representation method is proposed to reserve abundant vacancies as embedding room [35]. A local difference predictor is adopted to determine the label of the block, indicating the embedding capacity, which can also facilitate data extraction and image reconstruction [40].

More recently, block-based data hiding methods have been studied because pixels in a small region commonly have similar values and are easy to compress in order to generate spare room. For example, according to the distribution of the most significant bits in each block, embeddable blocks are determined and the most significant bits are adaptively compressed by using Huffman coding [31]. The original image is encrypted by a block-level encryption to preserve the correlation between pixels in each block, and then compresses the pixel differences for secret data recording [32]. The original binary image is divided into uniform and non-uniform blocks, where both of them are able to carry secret bits [38]. For the method [39], several of the most significant bit-planes are separated into non-overlapping blocks. If all or most pixels in a block are the same, the block is compressed by sparse encoding to reserve room. However, although the block-based methods mentioned above have achieved relatively good performance, the vacancy reservation is only limited to vacating spare room within each block; the correlation between adjacent blocks is totally ignored. As is pointed out in [3], the embedding capacity and visual symmetry/quality are complementary, which means the quality of the recovered image is very poor. Therefore, in this paper, a novel block embedding algorithm is introduced to implement secret data embedding in the binary image, where intra-block and inter-block correlations are both utilized for vacating spare bits. Then, a reversible data hiding method is proposed, to improve the embedding capacity for the grayscale image.

There are three main contributions in the proposed method. First, to make full use of the redundant space in the original image, two new types of block structure information

are designed, with which some non-embeddable basic blocks can be identified as embeddable. Thus, more spare space can be generated from these blocks to obtain a higher embedding capacity. Second, based on the local similarity of the image content, a novel adaptive recognition strategy is proposed to recognize adjacent blocks and then integrate them into combined blocks with large size. Instead of labeling individual basic blocks repeatedly, a group of new block-labeling bits are designed to mark combined blocks, with which fewer embeddable bits are consumed to label the types of basic and combined blocks. Third, the security of the proposed method is significantly enhanced to resist against possible attacks by using the high-dimension chaos system to implement image encryption. Meanwhile, the sensitivity of initial values enhances the security of the proposed method.

The rest of this paper is organized as follows. In Section 2, the embedding algorithm based on an adaptive combination of basic blocks in the binary image is introduced, which includes the embedding and extraction procedures. In Section 3, the fully reversible data hiding for the grayscale image is described in detail. In Section 4, experimental results and analysis are presented to demonstrate the superiority and security of the proposed method. Finally, a brief conclusion is given in Section 5.

2. The Proposed Embedding Algorithm

In this section, an adaptive recognition strategy is designed to implement the combination of basic blocks in a binary image, and then a new binary-block embedding algorithm is proposed, in which more embeddable bits are created for holding secret data.

2.1. Classification of Basic Binary Blocks

For the binary-block embedding method [39], a binary image with the size of $M \times N$ pixels is first divided into a number of non-overlapping basic blocks, and each block has $n = s_1 \times s_2$ pixels, where $s_1, s_2 \geq 3$. Then, these blocks are separated into two groups with respect to their individual characteristics. All blocks in one group are marked as good ones, which have spare space to accommodate secret data bits, and those in another group are marked as bad ones with a complicated structure that cannot be compressed efficiently. Taken overall, these blocks are specifically classified into five basic categories based on the pre-set conditions. As shown in Table 1, a block can be marked as a Bad block, or a Good-I/II/III/IV block. Let n_0 be the number of pixels with the value of 0 within a block and n_1 be the number of pixels with the value of 1; the minimum value between them is obtained as $m = \min\{n_0, n_1\}$. To correctly mark a block, a threshold denoted as n_a is mathematically calculated as:

$$n_a = \arg \max_x \{n - 3 - \max\{\lceil \log_2 x \rceil, 1\} - x \lceil \log_2 n \rceil \geq 0\} \quad (1)$$

where $1 \leq x \leq \lceil 0.16 * n \rceil$. It is an important index to distinguish good blocks from bad ones, because it denotes the maximum number of minority pixels that can be represented by a block itself.

Table 1. Block types in Yi and Zhou's method.

Condition	Block Type	Block Explanation	Labeling Bits
$m > n_a$	Bad	Cannot embed data	00
$m = n_0 = 0$	Good-I	All pixels with value of 1	11
$m = n_1 = 0$	Good-II	All pixels with value of 0	10
$1 \leq m \leq n_a, n_0 < n_1$	Good-III	Most pixels with value of 1	011
$1 \leq m \leq n_a, n_1 < n_0$	Good-IV	Most pixels with value of 0	010

After a block is classified, a set of fixed block-labeling bits is used to mark it in Yi and Zhou's method. For a Good-I (Good-II) block, the first 2 bits are directly substituted with the labeling bits "11" ("10") and the remaining bits are left to embed secret data bits, which means that the corresponding embedding capacity is $n-2$ bits. For a bad block, the first 2 bits are marked as "00" and other bits are kept unchanged. Because 2 extra embeddable bits are consumed to record the original 2 bits, the capacity of the block is -2 bits. For a Good-III (Good-IV) block, besides the first 3 bits being directly replaced with the bits "011" ("010"), the structure information consisting of the number of minority pixels and their locations must be recorded. Therefore, the embedding capacity of the block is calculated as $n-3-p-\sum_{i=1}^m q_i$, where $p = \max\{\lceil \log_2 n_a \rceil, 1\}$ is the number of bits to embed the parameter m , and q_i is the number of bits to record the i^{th} pixel's location. For the first pixel in the block, $q_1 = \lceil \log_2 n \rceil$, otherwise $q_i = \max\{\lceil \log_2 (n - z_{i-1}) \rceil, 1\}$, where z_{i-1} is the location index of the $(i-1)^{\text{th}}$ pixel. As a result, the increase in spare space to accommodate secret data bits largely depends on the number of Good-I/II/III/IV blocks. Specifically, if there are more embeddable blocks in the original binary image, the embedding capacity can be improved greatly.

After careful analysis of the condition that bad blocks satisfied, it is easily found that a block is identified as a bad block only when it cannot create enough space to store its structure information, which consists of the number of minority pixels and their locations. If the bits occupied by the structure information of a bad block are reduced, it is possible to obtain spare space for embedding secret data bits. If so, the block will be marked as a good block so that the embedding capacity of the binary image can be totally improved. For the sake of illustration, two blocks as shown in Figure 1 are considered. Scanning them from left to right and from top to bottom, it can be seen that one block contains a sequence of consecutive black pixels while another contains a sequence of consecutive white pixels. The size of two blocks is 4×4 pixels, and $n_a = 2$ is calculated using Equation (1). Referencing the conditions listed in Table 1, two blocks are obviously marked as bad ones. To generate spare space, new structure information of two blocks can be designed by the combination of the starting location and the length of the sequence, where the $s_s = \lceil \log_2 n \rceil$ bits are utilized to embed the starting location, and $l_s = \max\{\lceil \log_2 (n - s_s + 1) \rceil, 1\}$ bits are used to store the length of the sequence. As long as the sum of block-labeling bits, s_s and l_s bits is smaller than the number of the block pixels, the spare space can be created.

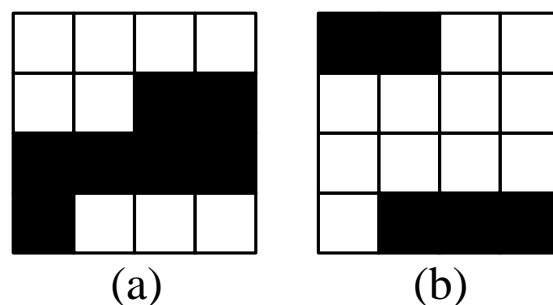


Figure 1. Two bad blocks which can be marked as good ones with new designed structure information. (a) 4 bits "0110" are used to record the starting location of the consecutive black pixels and 4 bits "0111" to the length; (b) 4 bits "0010" are used to record the starting location of the consecutive white pixels and 4 bits "1011" to the length.

To accurately identify such new types of blocks from bad ones, a new threshold denoted as n_b is calculated by the sum of exclusive-or values of every two adjacent pixels,

i.e., each pixel pair, after the block is flattened into a pixel sequence from the first to the last pixel in scan order. The mathematical formula is as follows:

$$n_b = (b_1 \oplus b_2) + (b_2 \oplus b_3) + (b_3 \oplus b_4) + \dots + (b_{n-1} \oplus b_n) \quad (2)$$

where b_1, b_2, \dots, b_n represent the values of pixels in the block. The parameter n_b indicates the number of distinct pixel pairs, i.e., adjacent pixels '10' or '01', in a block. It separates the block sequence into $n_b + 1$ alternating sub-sequences. When $n_b = 1$ or 2, there are 2 or 3 sequences existing within a block. To recover the block during the process of recovery, the starting position and length of one sequence should be recorded. Concretely, the value of n_b is used to determine whether there is a recordable consecutive pixel sequence within a block, and the value of the first pixel in the block, i.e., the value of b_1 , is used to determine the order of alternating white and black pixel sequences. As shown in Figure 2, there are four different situations in a block. Among them, starting position and length of the sequence mark with red rectangles should be recorded. Therefore, based on the threshold n_a, n_b and the parameters m, n_0, n_1, b_1 , blocks in a binary image can be classified into seven basic types in the proposed method, and the corresponding conditions are shown in Table 2 in detail. To avoid confusion, new symbols are adopted to describe the types of different blocks. For example, a block that cannot embed data is identified as "NEB", i.e., a block not suitable for embedding data, while a block with the symbol "EB" can offer spare space for storing secret bits.

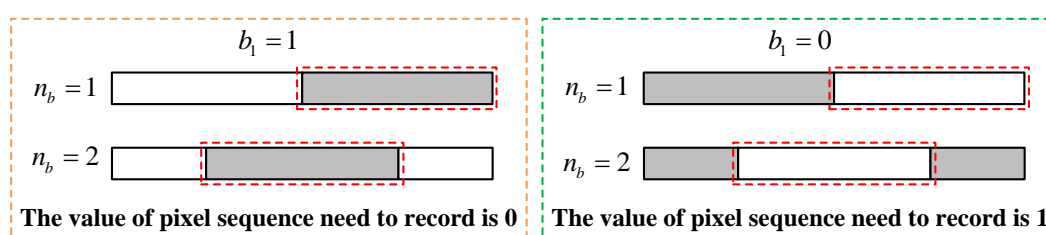


Figure 2. The arrangement of pixel sequences in a block for different n_b and b_1 .

Table 2. Basic block types in the proposed method.

Condition	Block Type	Block Explanation
$m > n_a, n_b > 2$	NEB	Not suitable for embedding data
$m = n_0 = 0$	EB-I	All pixels with value of 1
$m = n_1 = 0$	EB-II	All pixels with value of 0
$1 \leq m \leq n_a, n_0 < n_1$	EB-III	Most pixels with value of 1
$1 \leq m \leq n_a, n_1 < n_0$	EB-IV	Most pixels with value of 0
$m > n_a, n_b \leq 2, b_1 = 1$	EB-V	Including a sequence of consecutive pixels with the value of 0
$m > n_a, n_b \leq 2, b_1 = 0$	EB-VI	Including a sequence of consecutive pixels with the value of 1

2.2. Adaptive Recognition Strategy for Combined Blocks

As described above, different two or three bits are used as block-labeling bits to individually represent five basic blocks in Yi and Zhou's method, and then as much spare space as possible is created in a good block. Obviously, the local similarity of adjacent blocks in a binary image is not utilized, with which the embedding capacity can be further improved. As shown in Figure 3, it is very common that a local region in a binary image consists of contiguous basic blocks with the same type. Figure 3a is the binary image constructed with pixels of the 1th bit-plane, i.e., the most significant bit-plane, which contains the general pixel distribution of original image *Lena*. There are a large number of white and black regions in this binary image, which are mainly composed of basic blocks

marked with EB-I and EB-II, respectively. In addition, Figure 3b is the binary image constructed with pixels of the 5th bit-plane, where more details of the image information are exhibited and most basic blocks are marked with NEB in red regions. All these large and consecutive areas illustrate the reachability of more compressible space in the binary image.

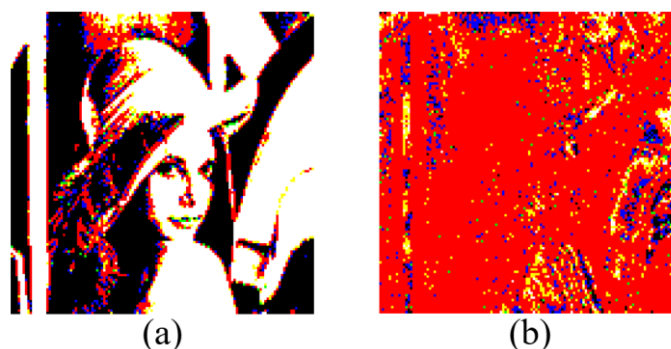


Figure 3. Binary image (a) constructed with pixels of the 1th bit-plane, (b) constructed with pixels of the 5th bit-plane of the image *Lena*, respectively. Note: the size of the basic block is 4×4 pixels, and the EB-I, EB-II, EB-III, EB-IV, EB-V, EB-VI and NEB block are displayed with the white, black, yellow, blue, orange, green and red colors, respectively.

For adjacent blocks with the type of EB-I or EB-II, if the whole region is marked with only one block-labeling bits, instead of each basic block being marked individually, a lot of spare space can be reserved. The first several bits of the first basic block in the region are substituted with the block-labeling bits, and the remaining bits are embeddable. Meanwhile, all bits of other basic blocks in this region can be completely used to accommodate secret bits. For adjacent basic blocks with the type of NEB, only the first several bits of the first block are replaced with the corresponding labeling bits, and all other pixels in the whole region are unchanged. Therefore, embeddable bits are no longer consumed to save original bits of other blocks. Obviously, it is very useful to combine adjacent basic blocks with the same type, such as EB-I, EB-II, or NEB, for improving the embedding capacity of a binary image. Moreover, the larger a region is, the more embeddable bits are reserved.

In order to obtain the region combined with more blocks of the same type, which have the same type, such as EB-I, EB-II, or NEB, an adaptive recognition strategy based on an auxiliary matrix is proposed, which is defined as:

$$G = \begin{bmatrix} (6,6) & (6,5) & (5,6) & (5,5) & (6,4) & (4,6) \\ (5,4) & (4,5) & (6,3) & (3,6) & (4,4) & (5,3) \\ (3,5) & (4,3) & (3,4) & (6,2) & (2,6) & (5,2) \\ (2,5) & (3,3) & (4,2) & (2,4) & (3,2) & (2,3) \\ (6,1) & (1,6) & (5,1) & (1,5) & (2,2) & (4,1) \\ (1,4) & (3,1) & (1,3) & (2,1) & (1,2) & (1,1) \end{bmatrix} \quad (3)$$

Here, each element (x, y) is a tuple and used to judge whether a rectangle region can be constructed with $2^x \times 2^y$ basic blocks. To ensure that larger regions are recognized prior to smaller ones, the tuple order of matrix G , i.e., the size of the combined region, is arranged in descending order and shrinks progressively. After a binary image is divided into a number of non-overlapping basic blocks as listed in Table 2, the combination process is executed as shown in Algorithm 1.

Algorithm 1. Adaptive block combination in binary images.**Input:** Binary image I , image size (M, N) , block size $s_1 \times s_2$, auxiliary matrix G .

```

1 Divide  $I$  into non-overlapping blocks  $B_{i,j}$  sized  $s_1 \times s_2$ ,  $1 \leq i \leq M$ ,  $1 \leq j \leq N$ ;
2 for each block  $B_{i,j}$  do
3   Calculate parameters  $m, n_0, n_1, n_a, n_b$  using Equations (1) and (2);
4   Determine its block type  $T_{i,j}$  according to the conditions listed in Table 2;
5   if  $T_{i,j} \in \{\text{EB-I, EB-II, NEB}\}$  and  $B_{i,j}$  is unprocessed then
6     for each element  $(x, y)$  in matrix  $G$  do
7       Set  $\mathfrak{R}_{i,j}^{x,y}$  as the rectangle region of size  $2^x \times 2^y$  starts with block  $B_{i,j}$ ;
8       Set  $\mathfrak{N}_{i,j}^{x,y}$  as the number of basic blocks with the type of  $T_{i,j}$  in region  $\mathfrak{R}_{i,j}^{x,y}$ ;
9       if  $\mathfrak{N}_{i,j}^{x,y} = 2^x * 2^y$  then
10        Record the block type  $T_{i,j}$ , the block coordinate  $(i, j)$ , and the element  $(x, y)$ ;
11        Mark each identical block in region  $\mathfrak{R}_{i,j}^{x,y}$  as processed;
12      end
13    end
14    if  $B_{i,j}$  is failed to be recognized until elements in matrix  $G$  are exhausted then
15      Set  $B_{i,j}$  as a single block retains its original type;
16      Mark  $B_{i,j}$  as processed;
17    end
18  end
19 end

```

Output: Combined results.

Repeating above combination algorithm to integrate more and more blocks with the same type of EB-I, EB-II or NEB into rectangle regions of different sizes, i.e., different combined blocks. For the region shown in the red box in Figure 4a, the distribution of basic blocks is displayed in Figure 4b. After the combination process, a large number of basic blocks of the types EB-I, EB-II, or NEB are obviously integrated into combined blocks, as shown in Figure 4c.

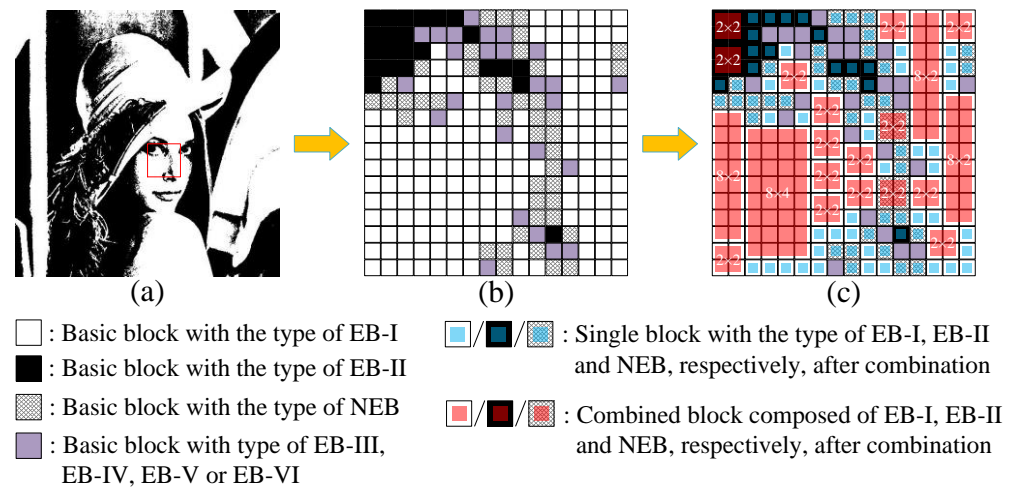


Figure 4. (a) Test binary image, (b) the distribution of basic blocks, and (c) the distribution of all blocks after combination.

2.3. Structure Information of Basic and Combined Blocks

After the adaptive recognition process is executed for a binary image, besides the seven types of basic blocks listed in Table 2, more new types for combined blocks within different rectangle regions are generated. As a result, all blocks including basic and combined blocks are reclassified into 16 types as listed in Table 3, where each type should be assigned a unique block-labeling bits. Obviously, the blocks indexed with 1, 5, 9, 13, 14, 15, and 16 are the same basic blocks as listed in Table 2.

Notably, a square combined block containing 16 white basic blocks should be marked with the type of EB-I₃, while a non-square one is identified as a block with EB-I₄ in the proposed method. Similarly, a square combined block containing 16 black basic blocks is recognized with EB-II₃ rather than EB-II₄, and a square block composed of 16 non-embeddable blocks is considered as NEB₃. In addition, it should be pointed out that the structure information of basic blocks in a combined block with the symbol of “NEB” can be different, and no spare space can be vacated by any of them.

To accurately determine different blocks as well as efficiently reduce the consumption of embeddable bits, a set of block-labeling bits should be properly designed for 16 types of blocks. To achieve this goal, the probability of occurrence of 16 types of blocks from the 1th to the 5th bit-plane are firstly calculated for test images in different image databases, such as USC-SIPI and BOWS2. Then, the corresponding codewords are generated for 16 types of blocks using Huffman coding based on the probability values from statistics. As listed in the fourth column of Table 3, it can be seen that the basic non-embeddable blocks are assigned the shortest codeword due to their huge number, while the combined blocks, such as EB-I₃, EB-I₄, EB-II₃ and EB-II₄, have the longest codewords.

Table 3. Classification of basic and combined blocks.

Index	Block Type	Block Size	Codewords	Block Explanation
1	EB-I ₁	1×1	011	A basic block where all pixels are 1, i.e., a white basic block
2	EB-I ₂	2×2	111111	A square block containing 4 white basic blocks
3	EB-I ₃	4×4	111101111	A square block containing 16 white basic blocks
4	EB-I ₄	2 ^x ×2 ^y	111101110	Containing at least 16 white basic blocks, excluding EB-I ₃ blocks
5	EB-II ₁	1×1	000	A basic block where all pixels are 0, i.e., a black basic block
6	EB-II ₂	2×2	111110	A square block containing 4 black basic blocks
7	EB-II ₃	4×4	111101101	A square block containing 16 black basic blocks
8	EB-II ₄	2 ^x ×2 ^y	111101100	Containing at least 16 black basic blocks, excluding EB-II ₃ blocks
9	NEB ₁	1×1	01	A non-embeddable basic block
10	NEB ₂	2×2	110	A square block containing 4 non-embeddable basic blocks
11	NEB ₃	4×4	1111010	A square block containing 16 non-embeddable basic blocks
12	NEB ₄	2 ^x ×2 ^y	111100	Containing at least 16 non-embeddable basic blocks, excluding NEB ₃ blocks
13	EB-III	1×1	101	A basic block where most pixels are 1
14	EB-IV	1×1	100	A basic block where most pixels are 0
15	EB-V	1×1	11101	A basic block including a sequence of consecutive pixels with the value of 0
16	EB-VI	1×1	11100	A basic block including a sequence of consecutive pixels with the value of 1

In Figure 5, varied block-labeling bits are exhibited for 16 different block types. It is obvious that the codewords of blocks of the type EB-I₁, EB-I₂, EB-I₃, EB-II₁, EB-II₂, EB-II₃, NEB₁, NEB₂, and NEB₃ can be directly considered as their corresponding labeling bits, because no additional information is specially needed to describe their structure. For a block marked with the type of EB-I₄, EB-II₄ or NEB₄, the number of basic blocks is not fixed, i.e.,

the size of rectangle region occupied by these blocks is changeable. It is necessary to record the number of basic blocks in the horizontal and vertical directions in the block-labeling bits besides the Huffman coding codewords. According to the auxiliary matrix G in Equation (3), the maximum number of basic blocks in two directions is 2^6 . Therefore, an additional 6 bits is added to the block-labeling bits following Huffman codeword for these 3 kinds of combined blocks, where the first 3 bits are used to record the first component x of the auxiliary matrix element and the last 3 bits to record the second component y . For a block marked with the type of EB-III or EB-IV, the block-labeling bits comprise the corresponding codeword and its structure information about the number of minority pixels m and their locations, where the length of bits to store these two parts are calculated as Equation (4) and Equation (5), respectively.

$$p = \max \{ \lceil \log_2 n_a \rceil, 1 \} \quad (4)$$

$$q_i = \begin{cases} \lceil \log_2 n \rceil & \text{for } i = 1 \\ \max \{ \lceil \log_2 (n - z_{i-1}) \rceil, 1 \} & \text{for } 2 \leq i \leq m \end{cases} \quad (5)$$

where p is the number of bits to embed the parameter m , q_i is the number of bits to record the location of i^{th} pixel in the block, and z_{i-1} is the location index of the $(i-1)^{\text{th}}$ pixel. Similarly, for a block marked with the type of EB-V or EB-VI, the block-labeling bits record the corresponding codeword and its structure information about the starting location and length of the consecutive sequence, which are s_s and l_s bits needed as expressed below:

$$s_s = \lceil \log_2 n \rceil \quad (6)$$

$$l_s = \max \{ \lceil \log_2 (n - s_s + 1) \rceil, 1 \} \quad (7)$$

Finally, payloads offered by different blocks can be calculated as listed in the last column of Figure 5. Note that the non-embeddable blocks, i.e., the block marked with type of NEB₁, NEB₂, NEB₃ and NEB₄, cannot generate any spare bits but inversely consume extra bits to save original bits occupied by their labeling bits, so the corresponding payloads are negative values. For other blocks, plenty of embeddable bits are emptied for recording secret data.

Block type	Illustration	Codewords	Block-labeling bits	Payload
EB-I ₁ (1×1)		001	0 0 1 codewords	$n - 3$
EB-I ₂ (2×2)		111111	1 1 1 1 1 codewords	$4 * n - 6$
EB-I ₃ (4×4)		111101111	1 1 1 0 1 1 1 1 codewords	$16 * n - 9$
EB-I ₄ (2 ^s ×2 ^t)		111101110	1 1 1 0 1 1 1 0 · 0 1 1 · 1 0 1 codewords binary 'x' binary 'y'	$2^s * 2^t * n - 15$
EB-II ₁ (1×1)		000	0 0 0 codewords	$n - 3$
EB-II ₂ (2×2)		111110	1 1 1 1 1 0 codewords	$4 * n - 6$
EB-II ₃ (4×4)		111101101	1 1 1 0 1 1 1 0 1 codewords	$16 * n - 9$
EB-II ₄ (2 ^s ×2 ^t)		111101100	1 1 1 0 1 1 1 0 0 · 1 1 0 · 1 0 0 codewords binary 'x' binary 'y'	$2^s * 2^t * n - 15$
NEB ₁ (1×1)		01	0 1 codewords	-2
NEB ₂ (2×2)		110	1 1 0 codewords	-3
NEB ₃ (4×4)		1111010	1 1 1 1 0 1 0 codewords	-7
NEB ₄ (2 ^s ×2 ^t)		111100	1 1 1 1 0 0 · 0 1 0 · 1 0 1 codewords binary 'x' binary 'y'	-12
EB-III (1×1)		101	1 0 1 · 101000100 codewords structure information	$n - 3 - p - \sum_{i=1}^m q_i$
EB-IV (1×1)		100	1 0 0 · 100111000 codewords structure information	$n - 3 - p - \sum_{i=1}^m q_i$
EB-V (1×1)		11101	1 1 1 0 1 · 01001000 codewords structure information	$n - 5 - s_s - l_s$
EB-VI (1×1)		11100	1 1 1 0 0 · 00111010 codewords structure information	$n - 5 - s_s - l_s$

Figure 5. Block-labeling bits and payloads for different block types.

2.4. Reversible Data Hiding for Binary Images

Based on the above analysis of the embedding capacity, it can be known that no spare space can be created in blocks with the types of NEB₁, NEB₂, NEB₃ and NEB₄. Even so, if non-embeddable blocks can be combined, it is very helpful for improving the embedding capacity due to reduction of the consumption of embeddable bits for recording the first several bits of all basic blocks. To illustrate the application of basic and combined blocks listed in Table 3, a new binary-block embedding algorithm is proposed for binary images.

Given a binary image, the block size $s_1 \times s_2$, and the auxiliary matrix G , the procedure of embedding secret data can be described in detail as follows:

- (1) Decompose the binary image into a set of non-overlapping basic blocks with the size of $s_1 \times s_2$ pixels. Meanwhile, a null sequence denoted as Seq is created to record additional bits for non-embeddable blocks.
- (2) Calculate the parameters such as n_0, n_1, m, n_a, n_b for each basic block using Equations (1) and (2), and then identify the block type based on the conditions listed in Table 2.
- (3) Combine basic blocks using the adaptive recognition strategy by the aid of the auxiliary matrix G , and then assign new types for finally obtained blocks by referencing Table 3.
- (4) Scan each basic block from left to right and from top to bottom. If the current basic block is marked processed, the next basic block is checked. Otherwise, execute the following steps.
- (5) If the current basic block is an individual one with the type of EB-I₁, EB-II₁, EB-III, EB-IV, EB-V, EB-VI, or NEB₁, it is marked as processed and handled as follows:

If the type is EB-I₁ or EB-II₁, the first 3 bits of the block are substituted with the block-labeling bits, i.e., the Huffman coding codeword, and the last $n - 3$ bits are vacated for data accommodation.

- (a) If the type is EB-III or EB-IV, the first $3 + p + \sum_{i=1}^m q_i$ bits are replaced with the block-labeling bits, and the remaining bits are vacated for data accommodation.
- (b) If the type is EB-V or EB-VI, the first $5 + s_s + l_s$ bits are replaced with the block-labeling bits, and the remaining bits are vacated for data accommodation.
- (c) If the type is NEB₁, the first 2 bits are replaced with the block-labeling bits, and the remaining bits are unchanged. The original first 2 bits of the block considered as additional bits are appended to *Seq*.

As shown in Figure 6a,e–g, the blocks with the type of EB-II₁, EB-IV, EB-VI, and NEB₁ are taken as examples to explain the vacating processes for different basic blocks.

- (6) If the current basic block is a combined one, all basic blocks in this region are marked as processed, and it is handled as follows:
 - (a) If the type is EB-I₂, EB-II₂ or EB-I₃, EB-II₃, the first 6 or 9 bits of the current block are substituted with the block-labeling bits, and the remaining $4 * n - 6$ or $16 * n - 9$ bits in the combined block are vacated for data accommodation.
 - (b) If the type is EB-I₄ or EB-II₄, the first 15 bits of the current block are replaced with the block-labeling bits, and the remaining $2^x * 2^y * n - 15$ bits in the combined block are vacated for data accommodation.
 - (c) If the type is NEB₂ or NEB₃, the first 3 or 7 bits of the current block are replaced with the block-labeling bits, and the remaining bits in the combined block are unchanged. The original first 3 or 7 bits of the current block considered as additional bits are appended to *Seq*.
 - (d) If the type is NEB₄, the first 12 bits of the current block are replaced with the block-labeling bits, and the remaining bits in the combined block are unchanged. The original first 12 bits considered as additional bits are appended to *Seq*.

As shown in Figure 6b–d,h–j, the blocks with the type of EB-II₂, EB-II₃, EB-II₄, NEB₂, NEB₃ and NEB₄ are taken as examples to explain the vacating processes for different combined blocks.

- (7) Until all basic blocks are marked as processed, a number of spare bits can be created. Meanwhile, the whole data to be embedded is constructed with three parts. The first part occupies 20 bits to record the length of secret data, i.e., the number of secret bits, the second part is the secret bits to be hidden, and the third part is the additional bits sequence *Seq*. Notably, the *Seq* is an interim to save the original bits replaced by block-labeling bits in NEB blocks, and is helpful to retrieve these original bits during image recovery.
- (8) Scan each basic block from left to right and from top to bottom; if it or its combined block has spare space, the bits of the whole data are embedded into the spare space sequentially.

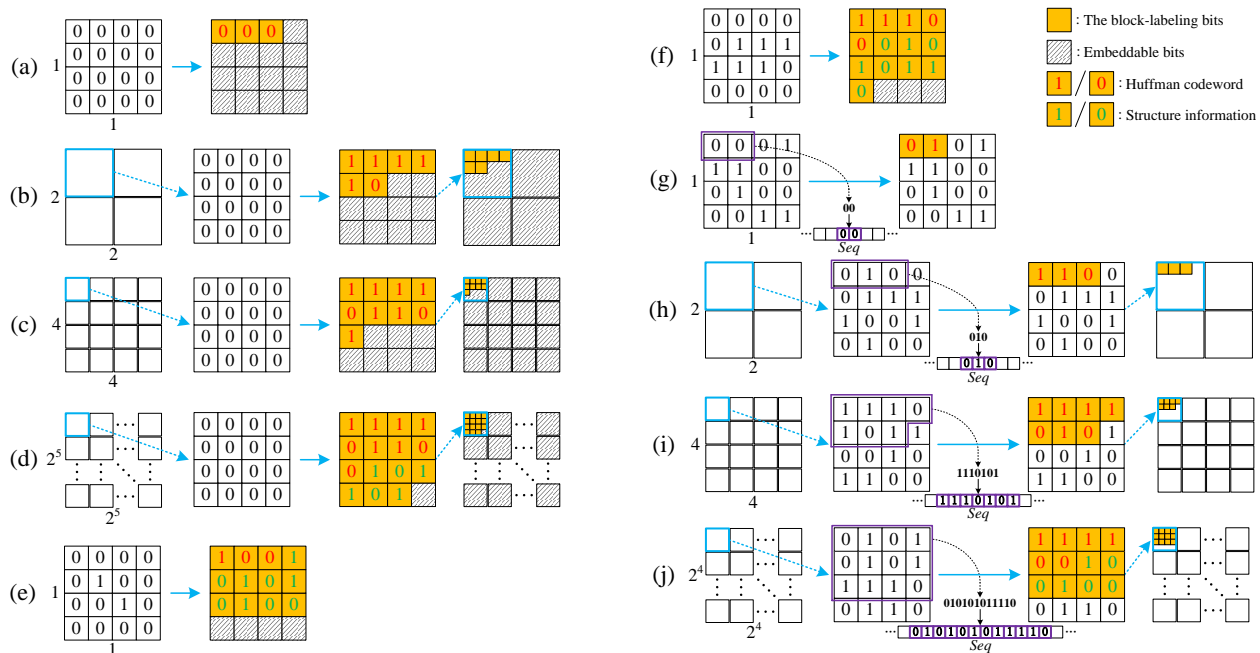


Figure 6. The vacation procedure of spare space for different blocks. (a,e–g), the blocks with the type of EB-II₁, EB-IV, EB-VI, and NEB₁ are taken as examples to explain the vacating processes for different basic blocks. (b–d,h–j), the blocks with the type of EB-II₂, EB-II₃, EB-II₄, NEB₂, NEB₃ and NEB₄ are taken as examples to explain the vacating processes for different combined blocks.

Of course, all above embedding procedures are fully reversible after data extraction and image recovery. That is, for each block in the marked image, the first several bits are first checked to determine which type the block belongs to, based on which, a determination can also be made as to whether there is necessary structure information to be extracted. If the current block is an individual or combined one with the symbol “NEB”, nothing will be done but record the block position in the image and the length of its labeling bits. If the current block is an individual or combined one with the symbol “EB”, the embedded data can be retrieved from remaining pixels, and the block is recoverable according to its block labeling bits. After the whole data is completely extracted, the secret data starts from the 21st bit and its length is calculated by the front 20 bits. Meanwhile, all non-embeddable blocks can be recovered in scan-order by retrieving their original front bits back from the subsequent part in the whole data, i.e., the additional bits sequence *Seq*, which guarantees that the extraction of secret data and the reconstruction of the image are both error-free.

3. The Proposed Hiding Method in Grayscale Images

In this section, the application of the above binary-block embedding algorithm to grayscale images is proposed to implement reversible data hiding with high capacity. As shown in Figure 7, the flowchart of the proposed method includes three processes, i.e., embeddable room reservation and image encryption, data embedding in the encrypted image, and data extraction and recovery of the original image. In the first process, the binary-block embedding is applied to higher bit-planes to create vacant room, and the bits in lower bit-planes of the original image integrated with additional bits are embedded into the vacated space. Then, the modified image is encrypted with the help of an encryption key. In the second process, after the secret data is encrypted using another encryption key, it is embedded into the lower bit-planes of the encrypted image. To further enhance security, the marked encrypted image is scrambled using the third encryption key. In the last process, the secret data and the original image can be recovered without errors, respectively, with respect to different security keys.

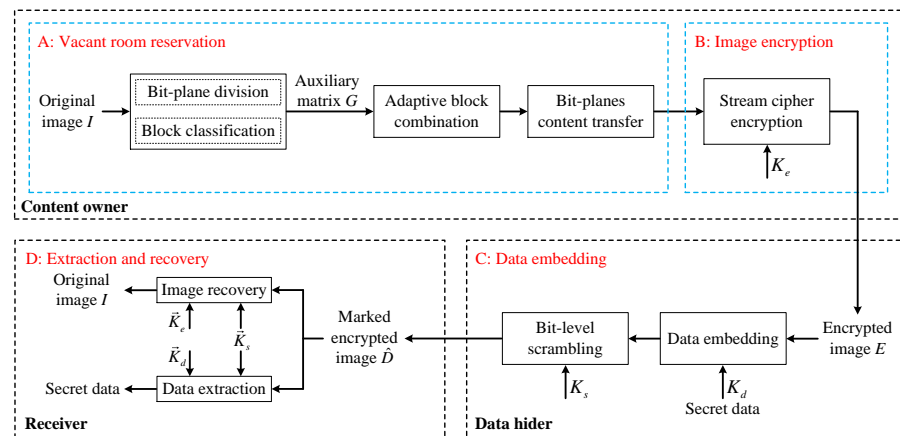


Figure 7. The flowchart of the proposed method.

3.1. Encryption of the original image

For an original grayscale image I with the size of $M \times N$ pixels and the pixel value in $[0, 255]$, it is first decomposed into 8 bit-planes, where I_1 is the most significant bit (MSB) plane and I_8 is the least significant bit (LSB) plane. Then, the proposed binary-block embedding algorithm is applied to each bit-plane for obtaining the corresponding capacity, which is denoted as C_i ($1 \leq i \leq 8$). If $C_i \leq 0$, the bit-plane is unable to accommodate any secret data. The total capacity of the original image, represented as C , can be mathematically calculated as:

$$C = \begin{cases} 0 & \text{if } k=0 \\ \sum_{i=1}^k C_i & \text{otherwise} \end{cases} \quad (8)$$

where the parameter k represents the number of MSB planes that can reserve bit space for data embedding, and is determined as

$$k = \begin{cases} 0 & \text{if } C_1 \leq 0 \\ \arg \max_t \left\{ \frac{\sum_{i=1}^t C_i}{M \times N} \leq 8 - t, t = 1, 2, \dots, 8 \right\} & \text{if } C_i > 0 \end{cases} \quad (9)$$

Next, the original bits of $8 - k$ LSB planes are selected sequentially and embedded into the embeddable blocks in MSB planes. Thus, the vacant space is transferred from MSB to LSB planes, which is helpful for both the embedding and extraction of secret data. Similar to the binary-block embedding algorithm, the whole data to be embedded is composed of two parts. The first part is the bits selected from LSB planes, which has the same length as that of the secret data. The second part is additional bits Seq composed of first several bits of blocks with the type of NEB_1 , NEB_2 , NEB_3 , and NEB_4 , which are replaced with block-labeling bits. Thus, the maximum data embedding rate, denoted as r , is calculated as

$$r = \frac{Q}{M \times N} \quad (10)$$

where Q is the maximum number of secret bits to be embedded, i.e., the maximum length of the first part in the whole data.

Image encryption is an effective way to protect the image content carrying secret data [41–43]. As a high-dimension chaos function, the Chen system has more complicated dynamical properties, which makes it more suitable for applications in the fields of image security. Therefore, a chaotic sequence that is very sensitive to the control parameters and

initial values of the Chen system is generated to accomplish the encryption of the modified original image. The Chen system can be mathematically described as

$$\begin{cases} \dot{x} = a(y - x) \\ \dot{y} = (c - a)x - xz + cy \\ \dot{z} = xy - bz \end{cases} \quad (11)$$

where a , b and c are the control parameters. Given the initial values denoted as $x(0)$, $y(0)$, and $z(0)$, when the control parameters are set to $a = 35$, $b = 3$, and $c \in [20, 28.4]$, three random sequences with non-periodic and non-convergent properties can be engendered by iterating the chaotic system with large iterations. In the process of iteration, the fourth order Runge-Kutta algorithm with a small step value such as 0.001 is repeatedly executed.

The framework of random sequence generation as shown in Figure 8 is adopted to implement image/secret data encryption. First, the encryption key K and the image/secret data are sent to a secure hash function (SHA) to generate two hash sequences with equal length. Then, an XOR operation is performed between them to obtain the mixed sequence, which is equally separated into three parts to form the initial key \hat{K} of the Chen chaotic system. Finally, the encrypted image or embedded cipher is generated by the XOR operation between the image/secret data and the random sequence generated with the Chen system. Both the sensitivity of SHA function and the Chen system jointly guarantee that the proposed method is differential-resistant because a tiny bias occurring to the original image or secret data will result in completely different encrypted results. The generation of the initial key \hat{K} is described in Algorithm 2. Notably, \hat{K} is helpful to decrypt the encrypted image so it needs to be sent to the receiver. Any SHA algorithm can be chosen here, and SHA-512 is selected for the proposed method. The binary sequence \mathcal{H} is the XOR result between the hash sequences of the encryption key and the image/secret data.

Algorithm 2. Initial values generation of the Chen system.

Input: Binary sequence $\mathcal{H} = [h_1, h_2, \dots, h_{512}]$ ($h_i \in \{0, 1\}$, $1 \leq i \leq 512$)

- 1 $\eta_1 \leftarrow \sum_{i=1}^{170} h_i 2^{170-i}$
- 2 $\eta_2 \leftarrow \sum_{i=171}^{340} h_i 2^{340-i}$
- 3 $\eta_3 \leftarrow \sum_{i=341}^{512} h_i 2^{512-i}$
- 4 $x(0) \leftarrow \eta_1 / 2^{170}$
- 5 $y(0) \leftarrow \eta_2 / 2^{170}$
- 6 $z(0) \leftarrow \eta_3 / 2^{172}$

Output: Initial values $\{x(0), y(0), z(0)\}$

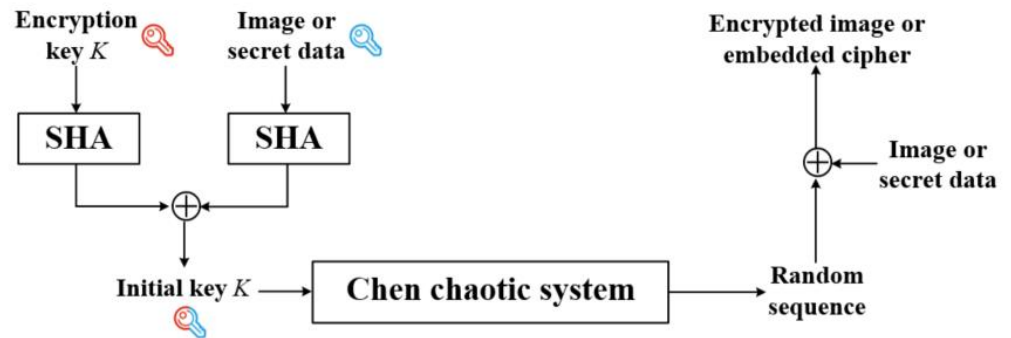


Figure 8. The flowchart of random sequence generation for encryption.

Given the encryption key K_e that generates initial values $x_e(0), y_e(0), z_e(0)$, i.e., the initial key \hat{K}_e , three chaotic sequences denoted as $x_e(i)$, $y_e(i)$, and $z_e(i)$ are generated using Equation (12). Then, using the following formula to produce sequence S_e for image encryption.

$$\begin{cases} S_e(3 \times j - 2) = \text{floor}(\text{abs}(x_e(j)) \times 10^{14}) \bmod 256 \\ S_e(3 \times j - 1) = \text{floor}(\text{abs}(y_e(j)) \times 10^{14}) \bmod 256 \\ S_e(3 \times j) = \text{floor}(\text{abs}(z_e(j)) \times 10^{14}) \bmod 256 \end{cases} \quad (12)$$

where $\text{abs}(\cdot)$ is used to calculate the absolute value, $\text{floor}(\cdot)$ is used to obtain the nearest integer of the argument, and $j = 1, 2, \dots, MN/3$. The sequence S_e is reshaped as an $M \times N$ matrix. After embedding the original bits of LSB planes into MSB planes of the original image, the modified result denoted as R is encrypted using the exclusive-or (XOR) calculation as:

$$E(i, j) = R(i, j) \oplus S_e(i, j) \quad (13)$$

where E is the encrypted image, and (i, j) represents the pixel coordinate.

3.2. Data Embedding in the Encrypted Image

In order to protect the content of secret data, it is also encrypted using the XOR calculation with the aid of data encrypted key K_d by the framework shown in Figure 8. Then, the encrypted secret data is embedded into LSB planes of the encrypted image directly using bit replacement. It should be noted that the first 20 bits in the LSB plane are used to save the length of secret data.

Different from the conventional LSB-based data hiding methods, the LSB planes in the proposed method are only used for accommodating secret data, not for reserving vacant room, where the vacant bits are created in the MSB planes. Meanwhile, because the secret data stored in LSB planes may be easily stolen or destroyed without any protection, the marked encrypted image is permuted using bit-level scrambling. According to the third encryption key K_s , the details of scrambling are described as follows:

- (1) For the marked encrypted image D with $M \times N$ pixels, the binary 8-bit of each pixel is concatenated into a sequence donated as ψ_D with the length of $8MN$ from left to right and from top to bottom.

- (2) Another sequence ψ_s composed of $8MN$ real numbers is generated with the encryption key K_s using the Chen system, and then the one-to-one mapping relationship is established between every two corresponding elements in the sequences ψ_D and ψ_s from beginning to end.
- (3) To permute the bits in ψ_D , the numbers in sequence ψ_s are sorted in ascending or descending order to form the sorted sequence $\hat{\psi}_s$. For each number in $\hat{\psi}_s$ which position has changed, the position alteration is recorded and the position of its mapping bit in ψ_D is altered accordingly until the last bit has finished moving.
- (4) The permuted sequence $\hat{\psi}_D$ is reshaped into 2D form by converting every 8-bit into the pixel value sequentially to construct the scrambled marked encrypted image \hat{D} .

The bit-leveling scrambling of the marked encrypted image D is the key to permute the image bits. In the scrambled result \hat{D} , the secret data is distributed uniformly in the whole image instead of in several LSB planes only, which guarantees that the proposed method has strong robustness and can firmly withstand potential attacks.

3.3. Data Extraction and Image Recovery

If only the initial keys \hat{K}_d and \hat{K}_s are known, the secret data can be completely extracted without knowing the content of the original image. Firstly, the image \hat{D} is scrambled inversely to obtain the marked image D using the sequence generated with the initial key \hat{K}_s . Then, the encrypted secret data is extracted from the LSB planes of the marked image and decrypted using the sequence generated with the encryption key \hat{K}_d . Of course, both the extraction and decryption of secret data are lossless.

When the initial keys \hat{K}_e and \hat{K}_s are accessible, the original image can be recovered approximately or completely. Similar to the extraction of secret data, the marked image D is firstly attained with the help of the third encryption key \hat{K}_s . At the same time, the first 20 bits in the LSB plane are extracted to calculate the length of secret data, from which it is known how many bits in the $8-k$ LSB planes are saved into the spare space in MSB planes. Then, the image D is directly decrypted without removing the secret data in LSB planes to obtain the modified image R using the XOR calculation with the help of key \hat{K}_e .

Finally, the image recovery procedure of the binary-block embedding algorithm is applied to MSB planes of the modified image. The whole data, including the bits selected from LSB planes and additional bits *Seq* composed of first several bits of blocks with the type of NEB₁, NEB₂, NEB₃, and NEB₄, are extracted from embeddable blocks. Meanwhile, all blocks with the symbol “EB” can be recovered based on their block-labeling bits and structure information. For blocks with the symbol “NEB”, the original first several pixels can be retrieved from the additional bits *Seq* of the whole data. If the bits in $8-k$ LSB planes are not substituted with the first part of the whole data, the original image I is recovered approximately. Otherwise, it can be completely restored without any loss of information.

4. Experimental Results and Discussion

4.1. Experimental Environment

In order to demonstrate the superiority and security of the proposed method, a series of experiments were carried out on different test images chosen from the USC-SIPI, BOSS-Base and BOWS2 image databases, which have a size of 512×512 pixels and the pixel value in $[0, 255]$. The environment of experiments is based on a personal computer equipped with a 3.60 GHz AMD 3600 processor, 16.00 GB memory, Windows 10 operating

system, and MATLAB R2018a. This section contains seven aspects of experimental results and analysis: (1) experimental result, (2) maximal embedding rate, (3) distortion of the marked decrypted images, (4) discussion on embedding rate, (5) security analysis, (6) sensitivity analysis of encryption keys, and (7) robustness analysis.

4.2. Experimental Result

The original image *Lena* and its marked encrypted result are shown in Figure 9a,b respectively, where the block size is set to 4×4 pixels and the embedding rate is 2.0384 bpp. The encrypted image is noisy to the extent that both the original image and secret data can be simultaneously protected, and it is difficult to discern any valid information from it. The directly decrypted image without removing the secret data is displayed in Figure 9c, where there is no visual difference between it and the original image. Figure 9d shows the completely recovered image, which is the same as the original image.

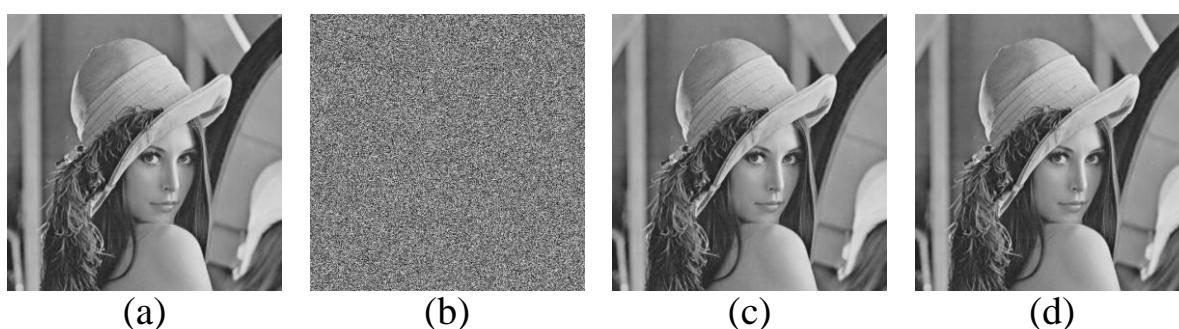


Figure 9. Data hiding and extraction using the proposed method where the embedding rate is 2.0384 bpp. (a) The original image, (b) the marked encrypted image, (c) the approximately decrypted image, and (d) the completely decrypted image.

4.3. Maximal Embedding Rate

To demonstrate that the proposed method has a high maximal embedding rate, several test images as shown in Figure 10 are used to hide secret information. The comparison with 8 state-of-the-art methods [24,30–34,37,39], which are all reversible data hiding methods, is illustrated in Figure 11. It can be seen intuitively that the embedding rate of the proposed method is much larger than that of other methods, regardless of whether the test image is smooth or textural. For the smooth images such as *Airplane*, the maximal embedding rate of other methods is 2.1928 bpp, 2.1899 bpp, 1.8816 bpp, 1.7980 bpp, 1.7002 bpp, 1.0800 bpp, 0.9866 bpp, and 0.0508 bpp, respectively, while the proposed method recorded 2.4683 bpp. By comparison, the proposed method can improve the embedding capacity with an increase of 0.2755 bpp, 0.2784 bpp, 0.5867 bpp, 0.6703 bpp, 0.7681 bpp, 1.3883 bpp, 1.4817 bpp and 2.4175 bpp, respectively. For the textural images such as *Barbara*, the maximal embedding rate of the proposed method can exceed 1.5 bpp, which is much higher than that of other methods. The reason is that, besides intra-block correlation, the correlation between adjacent blocks is effectively exploited to increase the hiding capacity.

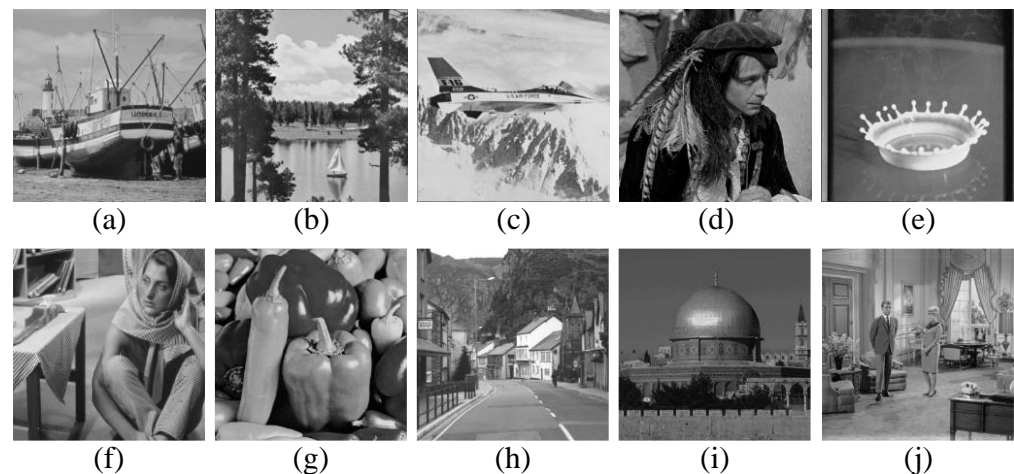


Figure 10. Test images used in experiments. (a) *Boat*, (b) *Lake*, (c) *Airplane*, (d) *Man*, (e) *Splash*, (f) *Barbara*, (g) *Peppers*, (h) *Road*, (i) *Temple*, (j) *Couple*.

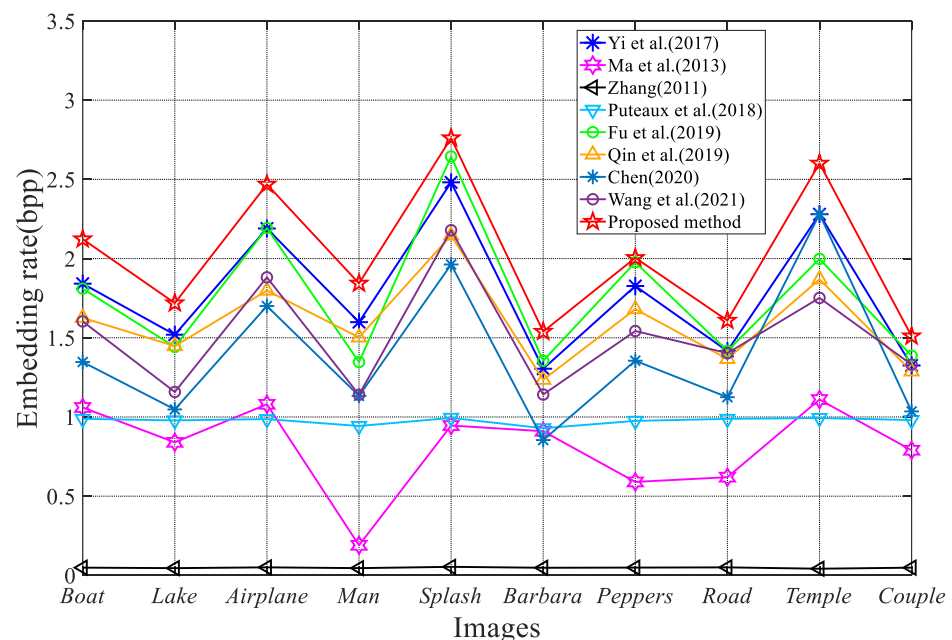


Figure 11. Comparison of maximal embedding rate using different methods, including Yi et al. [39], Ma et al. [34], Zhang [24], Puteaux et al. [37], Fu et al. [31], Qin et al. [30], Chen [32] and Wang et al. [33].

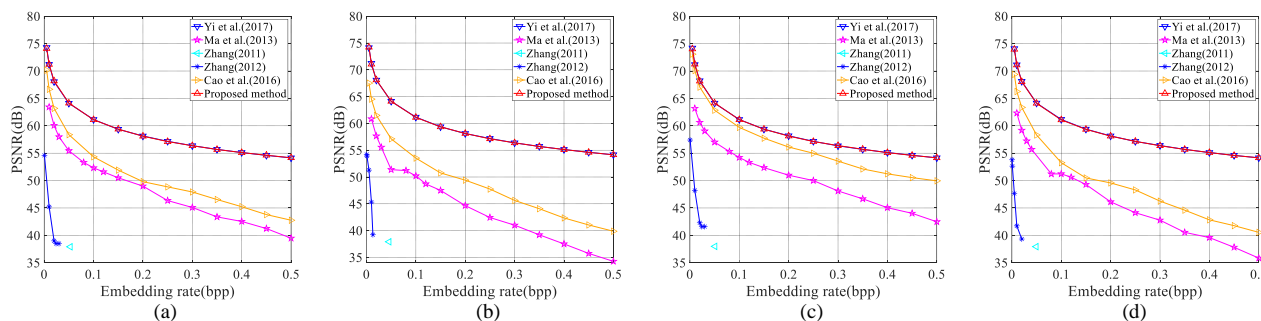
Moreover, in order to further demonstrate the universal improvement of embedding capacity, the extensive experiments are performed on all images in the BOSSBase and BOWS2 databases, which contains 10,000 test images, respectively. The results obtained for the two databases are illustrated in Table 4. Generally, the embedding rate varies diversely for different image textures and the results of smooth test images are higher than those for textural images. From Table 4, it can be seen that the proposed method outperforms other methods regardless of whether the test image is smooth or textural. On average, the proposed method is 0.3564 bpp, 0.3924 bpp, and 0.2700 bpp higher than methods [30,31,39] on the database BOSSBase. Similarly, it is obvious that the proposed method is also 0.3738 bpp, 0.3892 bpp, and 0.2598 bpp higher in average embedding capacity for database BOWS2. In addition, the worst embedding rate of the proposed method on database BOSSBase is much lower than that of BOWS2. This is mainly because the images in BOSSBase are much more complex than those in BOWS2.

Table 4. Maximal embedding rate comparison with three existing methods on two image databases.

Algorithms	BOSSBase			BOWS2		
	Best	Worst	Average	Best	Worst	Average
Qin et al.'s [30]	3.4885	0.1039	2.2093	3.4265	0.1348	2.0608
Fu et al.'s [31]	3.4657	0.0015	2.1733	3.4052	0.0296	2.0454
Yi et al.'s [39]	3.4878	0.0522	2.2957	3.4271	0.2191	2.1748
Proposed	3.9722	0.1367	2.5657	3.8846	0.2840	2.4346

4.4. Distortion of the Marked Decrypted Images

As described in the procedure of secret data extraction and image recovery, the first part of the whole data containing the bits selected from LSB planes of the original image can be extracted from embeddable blocks in MSB planes. If these bits are not replaced to their original positions, the image is recovered approximately, i.e., the marked decrypted image is obtained with distortion. To evaluate the rate versus distortion performance of the decrypted image, the curves between the embedding rate and the PSNR value for the marked decrypted images obtained with different methods are displayed in Figure 12. Obviously, the similar curves between the embedding rate and the PSNR value are generated for the proposed method and Yi and Zhou's method. This is because only LSB planes are used for accommodating secret information in the embedding procedure of two methods, and the original bits in MSB planes can be recovered without errors. For other methods, it can be seen that the visual quality of the marked decrypted image degraded severely with the increase in embedding rate, because the higher bit-planes may be involved in recording secret bits, which causes serious deterioration to the image quality.

**Figure 12.** PSNR comparison for the marked decrypted images using different methods for test images (a) *Lena*, (b) *Lake*, (c) *Airplane*, and (d) *Peppers*, respectively [24,25,34,35,39].

4.5. Discussion on Embedding Rate

As shown in Figure 11, it is known that the proposed method can create more vacant bits for secret data embedding compared with other state-of-the-art methods, especially Yi and Zhou's method. After careful consideration, it can be found that there are two main reasons for this performance. Firstly, by taking basic blocks, including a sequence of consecutive white or black pixels, into consideration, some bad blocks can be recognized as embeddable with one of the new types as listed in Table 2, i.e., EB-V or EB-VI. Therefore, not only are no extra spare bits consumed to mark these blocks as bad ones, but additionally, embeddable bits can be created to record the secret data. The comparison of the number of embeddable basic blocks between the proposed method and Yi and Zhou's method on different test images is displayed in Table 5, where basic blocks in all 8 bit-planes of each image are counted. The proposed method can clearly identify more embeddable basic blocks than Yi and Zhou's method regardless of whether the test image is smooth or textural, which is an increase of 2187 blocks on average. Meanwhile, the number of embeddable bits created in blocks with the type of EB-V and EB-VI using the proposed

method for each test image is listed in Table 6, from which it can be seen that a large number of embeddable bits are created, especially for test images such as *Boat*, *Airplane*, *Road*, *Temple* and *Couple*.

Table 5. Comparison of the number of embeddable basic blocks obtained by the proposed method and Yi and Zhou’s method

Test Images	The Number of Embeddable Basic Blocks		Increment
	Yi and Zhou	Proposed	
<i>Boat</i>	45,429	47,861	2432
<i>Lake</i>	39,089	40,573	1484
<i>Airplane</i>	52,807	55,461	2654
<i>Man</i>	41,884	43,203	1319
<i>Splash</i>	56,929	59,033	2104
<i>Barbara</i>	34,862	36,318	1456
<i>Peppers</i>	44,599	45,899	1300
<i>Road</i>	37,662	40,937	3275
<i>Temple</i>	68,871	71,316	2445
<i>Couple</i>	35,658	39,054	3396
Average	45,779	47,966	2187

Secondly, by making the use of the local similarity of each bit-plane, adjacent monochromatic basic blocks can be integrated into combined blocks with the type of EB-I₂, EB-I₃, EB-I₄, EB-II₂, EB-II₃, or EB-II₄, as listed in Table 3 in the proposed method. Although the block-labeling bits increase by several bits, a large number of embeddable bits can be reserved in the combined blocks in total after integration. Considering the number of monochromatic blocks is constant for test images, the comparison of the number of block-labeling bits used to represent all monochromatic blocks in the proposed method and Yi and Zhou’s method on different test images is displayed in Table 7, where basic and combined blocks in all 8 bit-planes are counted. Obviously, much fewer bits are consumed to identify all monochromatic blocks using the proposed method, thus more bits in the combined blocks are reserved as embeddable bits to record secret information. The average decrement in the length of block-labeling bits is 27,317 among these test images, which means that more 27,317 bits can be saved for data embedding and the embedding capacity is improved.

Table 6. The number of embeddable bits created in blocks with the type of EB-V and EB-VI for different images using the proposed method

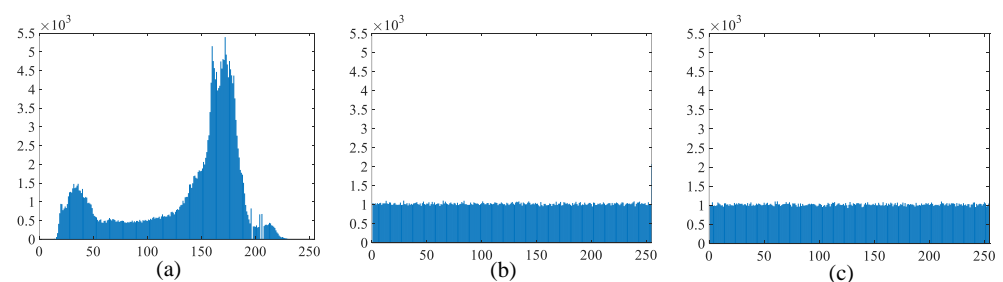
Test Images	The Number of Embeddable Bits		Total
	EB-V	EB-VI	
<i>Boat</i>	5883	6123	12,006
<i>Lake</i>	4460	4693	9153
<i>Airplane</i>	7615	6858	14,473
<i>Man</i>	3805	4629	8434
<i>Splash</i>	6354	6001	12,355
<i>Barbara</i>	4094	3937	8031
<i>Peppers</i>	3534	4191	7725
<i>Road</i>	8903	8351	17,254
<i>Temple</i>	8442	8376	16,818
<i>Couple</i>	8185	8482	16,667
Average	6128	6164	12,292

Table 7. Comparison of the number of block-labeling bits for monochromatic blocks using the proposed method and Yi and Zhou’s method.

Test Images	The Number of Block-labeling Bits		Decrement
	Yi and Zhou	Proposed	
<i>Boat</i>	71,266	42,465	28,801
<i>Lake</i>	57,474	37,209	20,265
<i>Airplane</i>	83,588	46,218	37,370
<i>Man</i>	62,754	40,467	22,287
<i>Splash</i>	92,538	46,407	46,131
<i>Barbara</i>	52,552	35,772	16,780
<i>Peppers</i>	66,980	43,485	23,495
<i>Road</i>	55,778	39,822	15,956
<i>Temple</i>	107,228	56,268	50,960
<i>Couple</i>	50,872	39,744	11,128
Average	70,103	42,786	27,317

4.6. Security Analysis

Since the marked encrypted image is vulnerable to malicious attacks such as histogram, statistical analysis and differential attack when transmitted in a public channel, related security evaluation and analysis of the proposed method should be performed. Firstly, taking the test image *Boat* as an example, the histograms of the original image, the encrypted image, and the marked encrypted image are displayed in Figure 13. It can be seen clearly that the distribution of the pixels in Figure 13a is conspicuous and reveals the statistical characteristics of the original image. In contrast, the histograms of the encrypted images before and after data embedding are both uniformly distributed, as shown in Figure 13b,c respectively, from which any useful statistical features cannot be discerned. Similar results are obtained for other test images, which means that the proposed method can efficiently resist against the histogram analysis attack.

**Figure 13.** Histograms of (a) the test image *Boat*, (b) the encrypted image, and (c) the marked encrypted image.

Secondly, the evaluation of encryption security is extended by measuring the information entropy, the number of pixel changing rate (NPCR), and the unified average changed intensity (UACI) of the images. Suppose that an image I is a grayscale image with $M \times N$ pixels. The relevant calculations are given as follows:

- Information entropy:

$$H(I) = -\sum_{x=0}^{255} \Theta(x) \log_2(\Theta(x)) \quad (14)$$

where x in the range of $[0, 255]$ and $\Theta(x)$ is its probability.

- Number of pixel changing rate (NPCR)

$$\text{NPCR} = \frac{\sum_{i=1}^M \sum_{j=1}^N d(i, j)}{M \times N} \quad (15)$$

where $d(i, j)$ is defined as:

$$d(i, j) = \begin{cases} 1 & , \text{ if } I(i, j) \neq E(i, j) \\ 0 & , \text{ otherwise} \end{cases} \quad (16)$$

- Unified average changed intensity (UACI)

$$\text{UACI} = \frac{1}{M \times N} \sum_{i=1}^M \sum_{j=1}^N \frac{|I(i, j) - E(i, j)|}{255} \quad (17)$$

where $I(i, j)$ and $E(i, j)$ denotes denote the original pixel and the encrypted pixel, respectively. Table 8 presents the value of entropy, NPCR and UACI for images before and after encryption. It can be seen that the encrypted image and the marked encrypted image both have greater entropy value than those of original images, which means that the proposed method is effective at obscuring the original content and protecting the hidden data. Additionally, the values of NPCR and UACI for the encrypted images are close to ideal values, which also proves the significant differences between the original and the encrypted image.

Table 8. The entropy, NPCR, and UACI of different images before and after encryption.

Images	Original Image	Encrypted Image			Marked Encrypted Image		
	Entropy	Entropy	NPCR	UACI	Entropy	NPCR	UACI
<i>Boat</i>	7.1238	7.9994	0.9962	0.2944	7.9993	0.9960	0.2941
<i>Airplane</i>	6.6776	7.9992	0.9961	0.3239	7.9992	0.9963	0.3225
<i>Lake</i>	7.4570	7.9992	0.9962	0.3141	7.9993	0.9962	0.3137
<i>Splash</i>	7.2533	7.9993	0.9961	0.3003	7.9994	0.9959	0.3007
<i>Temple</i>	5.8108	7.9993	0.9960	0.2969	7.9994	0.9960	0.2974
<i>Couple</i>	7.0581	7.9993	0.9960	0.2818	7.9992	0.9961	0.2803

In addition, the performance resisting against differential attack is evaluated from two perspectives, i.e., differential analysis to the original image and secret data, respectively. As shown in Figure 14a,b, two original images are encrypted with the same security keys, K_e and K_s , where the image in Figure 14b is generated from the image in Figure 14a by changing its pixel value 174 in position (197, 318) to 173. Then, the same secret data encrypted with the same security key K_d is embedded into two original images, It can be observed that even one bit difference between two original images yields totally different results, as shown in Figure 14f. Furthermore, the test image *Butterfly* in Figure 15a is considered the original image and encrypted with the same security key. Two pieces of secret information are embedded into the original image, where there is one bit difference between them. The difference between two marked encrypted images is shown in Figure 15d, which is very noisy. Obviously, even one bit difference between two pieces of secret information also yields totally different results. Therefore, the proposed method has high resistance against differential attack because a tiny change in the original image or secret data can result in a significantly different encrypted result.

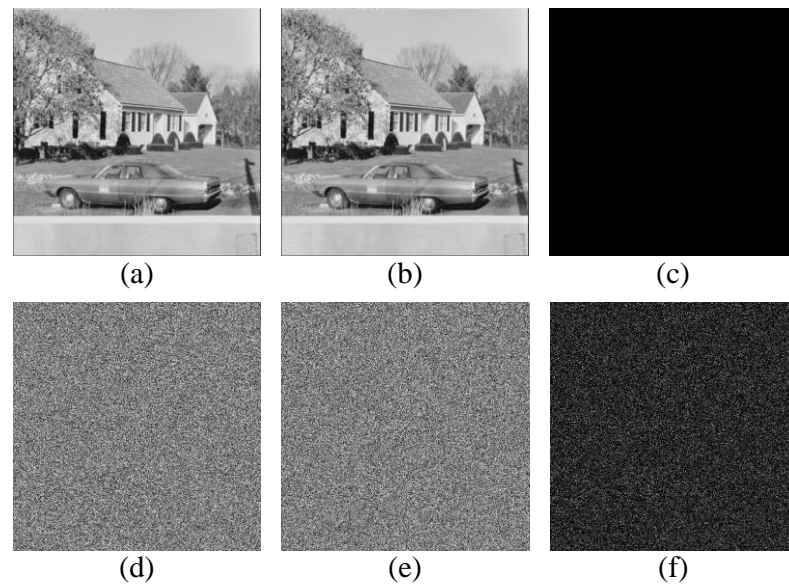


Figure 14. Differential analysis to the original image. (a,b) Two test images, where there is one bit difference between them; (c) the difference between two test images; (d,e) the marked encrypted images embedded with the same secret data; and (f) the difference between two marked encrypted results.

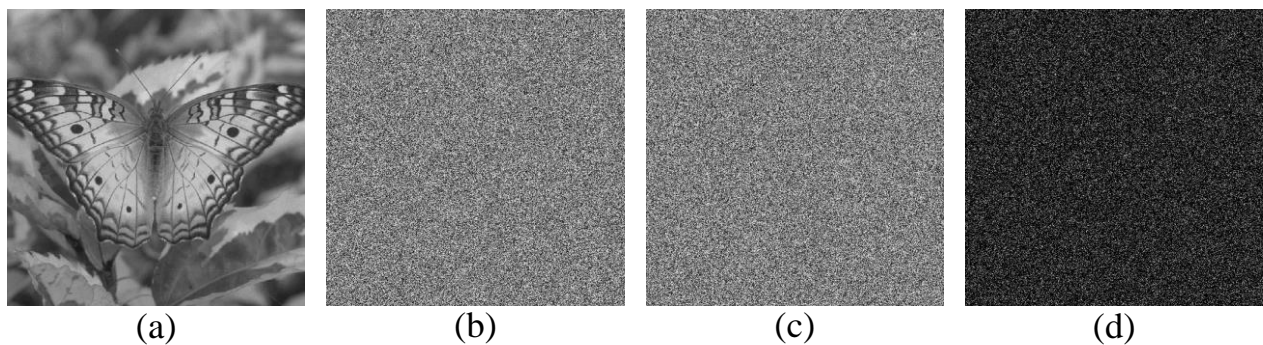


Figure 15. Differential analysis to the secret data. (a) Test image; (b,c) the marked encrypted images embedded with two pieces of secret data, where there is one bit difference between them; and (d) the difference between the two marked encrypted results.

4.7. Sensitivity Analysis of Encryption Keys

As described in the procedure of data embedding, the original image is encrypted and scrambled using random sequences generated by the Chen system with different initial values, i.e., the initial keys \hat{K}_e and \hat{K}_s . Similarly, the secret data is encoded with two encryption keys, i.e., K_d and K_s . Because the generated sequence is very sensitive to the initial values of the Chen system, the size of the key space should be sufficiently large to resist the brute-force attack. That is to say the correct initial keys cannot be obtained by exhaustively retrieving all possible keys. For the sake of simplicity, the mean squared error (MSE) between the original image and the reconstructed image that measures the recovery quality is evaluated when \hat{K}_e and \hat{K}_s are changed in tiny deviations.

Taking test images *Man*, *Peppers*, and *Barbara* as examples, the relationship curves of the MSE value between the original image and the corresponding recovered image versus the deviation of the initial value of the Chen system are plotted in Figures 16 and 17. In Figure 16, the original image cannot be thoroughly recovered when the deviation of $x_e(0)$ is larger than 10^{-15} , the deviation of $y_e(0)$ or $z_e(0)$ is larger than 10^{-14} . Similarly, when the deviation of $x_s(0)$ is larger than 10^{-15} , the deviation of $y_s(0)$ or $z_s(0)$ is larger

than 10^{-14} , the original image still cannot be discerned, as shown in Figure 17. Therefore, the possible combination of the initial keys, i.e., \hat{K}_e and \hat{K}_s , is approximately 10^{86} , which is enormous enough to resist all kinds of brute-force attack.

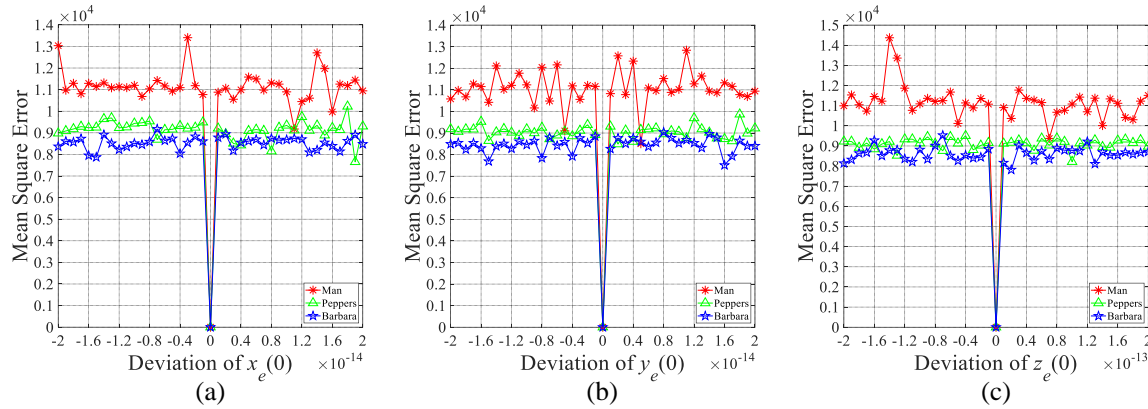


Figure 16. MSE versus the deviation of the initial key \hat{K}_e . (a–c) The mean square error curve versus the deviation of the initial value $x_e(0), y_e(0), z_e(0)$, respectively.

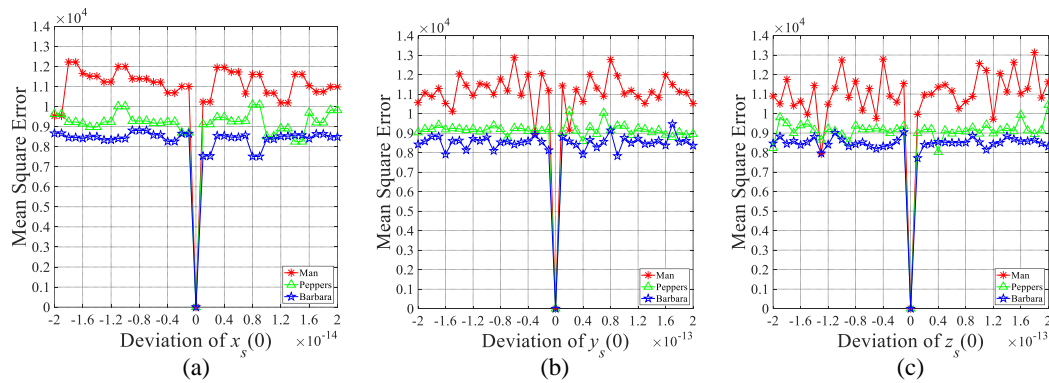


Figure 17. MSE versus the deviation of the initial key \hat{K}_s . (a–c) The mean square error curve versus the deviation of the initial value $x_s(0), y_s(0), z_s(0)$, respectively.

4.8. Robustness Analysis

As an important requirement in the procedure of secret data extraction, the robustness against noise, occlusion and JPEG attacks should be analyzed, because the marked encrypted image inevitably endures the influence of noise and data loss during transmission and storage. To demonstrate the performance of resisting possible attacks, the test images *Man*, *Peppers* and *Barbara* are used as the original images, with which a series of experiments are conducted. The binary image is applied as the secret data, which is the logo of our university. Here, the polarized correlation (PC) is adopted to evaluate the similarity between the original and the extracted secret image, which is mathematically calculated as:

$$PC = \frac{\sum_{i=1}^m \sum_{j=1}^n (w_{ij} - 0.5) * (\hat{w}_{ij} - 0.5)}{\sqrt{\sum_{i=1}^m \sum_{j=1}^n (w_{ij} - 0.5)^2} * \sqrt{\sum_{i=1}^m \sum_{j=1}^n (\hat{w}_{ij} - 0.5)^2}} \quad (18)$$

where w and \hat{w} are the original and the extracted secret image, respectively, and m, n is the image size. Figure 18 shows the secret image extracted from the marked encrypted

image, which is tempered by adding 0.01 salt-and-pepper or 0.005 Gaussian noise, occluding the central 40×40 pixels, discarding the LSB or MSB plane, histogram equalization and JPEG compression attacks, respectively. It can be seen that the secret image can be extracted with high visual quality for most situations. Notably, although the extracted results are degraded seriously for JPEG compression or by adding 0.005 Gaussian noise, the main content of the logo can be intuitively perceived. Therefore, the proposed method is highly robust, which can efficiently identify the privacy of the content owner in real applications.




























Images	Without noise	Adding 0.01 Salt&Peppers noise	Occluding central 40×40 pixels	Discarding LSB plane	Discarding MSB plane	Histogram equalization	JPEG2000: compression ratio=1.1	JPEG compression: quality=90	Adding 0.005 Gaussian noise
<i>Man</i>	 1.0000	 0.9903	 0.9925	 0.8721	 0.8742	 0.7490	 0.7247	 0.4830	 0.2044
<i>Peppers</i>	 1.0000	 0.9896	 0.9927	 0.8734	 0.8776	 0.7508	 0.7232	 0.4847	 0.1961
<i>Barbara</i>	 1.0000	 0.9901	 0.9926	 0.8743	 0.8754	 0.7504	 0.7281	 0.4589	 0.1967

Figure 18. The performance of secret data extraction under potential attacks for different test images.

5. Conclusions

In this paper, a novel block embedding algorithm is proposed based on an adaptive recognition strategy for combined blocks in binary images. First of all, the adjacent white, black, and non-embeddable basic blocks are integrated into the combined blocks. Fewer bits in the combined blocks are used as block-labeling bits, while more bits are reserved to accommodate secret data. Secondly, by introducing new structure information, i.e., blocks including a sequence of consecutive white or black pixels, plenty of non-embeddable blocks are identified as embeddable, which can vacate more spare bits. Based on the proposed binary-block embedding algorithm, a reversible data hiding method for gray-scale images in the encryption domain is proposed by making use of the high-dimension chaotic system. Only LSB planes of the original image are used for accommodating secret data, and the original bits in MSB planes can be recovered without errors; thus, the approximately-recovered image has better visual quality than other reversible data hiding methods. The sensitivity to the initial values of the high-dimension chaotic system guarantees that the proposed method has enhanced security performance. A series of experiments have demonstrated that the proposed method can not only obtain excellent embedding capacity, but also provide enough robustness to resist against various attacks. In the future, the work is expected to realize high-capacity and secure data hiding for audio, video and the 3D cloud, to enable wider and more general practical applications.

Author Contributions: Methodology, L.S. and Z.P.; Software, Z.P.; Validation, Z.X.; Writing—original draft, H.L. All authors have read and agreed to the published version of the manuscript.”

Funding: National Natural Science Foundation of China (NSFC): 62031023.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data will be made available on request.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

- Hong, W.; Chen, T.-S. A novel data embedding method using adaptive pixel pair matching. *IEEE Trans. Inf. Forensics Secur.* **2012**, *7*, 176–184.
- Rai, A.K.; Kumar, N.; Kumar, R.; Om, H.; Chand, S.; Jung, K.H. Implementation of Block-Based Hierarchical Prediction for Developing an Error-Propagation-Free Reversible Data Hiding Scheme. *Symmetry* **2021**, *13*, 1072.
- Lee, C.F.; Shen, J.J.; Wu, Y.J.; Agrawal S PVO-based reversible data hiding exploiting two-layer embedding for enhancing image fidelity. *Symmetry* **2020**, *12*, 1164.
- Coatrieux, G.; LeGuillou, C.; Cauvin, J.-M.; Roux, C. Reversible watermarking for knowledge digest embedding and reliability control in medical images. *IEEE Trans. Inf. Technol. Biomed.* **2009**, *13*, 158–165.
- Huang, L.-C.; Tseng, L.-Y.; Hwang, M.-S. A reversible data hiding method by histogram shifting in high quality medical images. *J. Syst. Softw.* **2013**, *86*, 712–727.
- Randey, R.; Singh, A.; Kumar, B.; Mohan, A. Iris based secure NROI multiple eye image watermarking for teleophthalmology. *Multimed. Tools Appl.* **2016**, *75*, 14381–14397.
- Bhardwaj, R.; Aggarwal, A. An enhanced separable reversible and secure patient data hiding algorithm for telemedicine applications. *Expert Syst. Appl.* **2021**, *186*, 115721.1–115721.14.
- Shi, Y.Q.; Li, X.; Zhang, X.; Wu, H.T.; Ma, B. Reversible data hiding: Advances in the past two decades. *IEEE Access* **2016**, *4*, 3210–3237.
- Celik, M.; Sharma, G.; Tekalp, A.; Saber, E. Lossless generalized-LSB data embedding. *IEEE Trans. Image Process* **2005**, *14*, 253–266.
- Celik, M.; Sharma, G.; Tekalp, A. Lossless watermarking for image authentication: A new framework and an implementation. *IEEE Trans. Image Process* **2006**, *15*, 1042–1049.
- Zhang, W.; Hu, X.; Li, X.; Yu, N. Recursive histogram modification: Establishing equivalency between reversible data hiding and lossless data compression. *IEEE Trans. Image Process* **2013**, *22*, 2775–2785.
- Lin, C.C.; Liu, X.L.; Yuan, S.M. Reversible data hiding for VQ-compressed images based on search-order coding and state-codebook mapping. *Inf. Sci.* **2015**, *293*, 314–326.
- Ni, Z.; Shi, Y.Q.; Ansari, N.; Su, W. Reversible data hiding. *IEEE Trans. Circuits Syst. Video Technol.* **2006**, *16*, 354–362.
- Wang, J.; Ni, J.; Zhang, X.; Shi, Y.-Q. Rate and distortion optimization for reversible data hiding using multiple histogram shifting. *IEEE Trans. Cybern.* **2017**, *47*, 315–326.
- Chen, H.; Ni, J.; Hong, W.; Chen, T.S. Reversible data hiding with contrast enhancement using adaptive histogram shifting and pixel value ordering. *Signal Process Image Commun.* **2016**, *46*, 1–16.
- Jia, Y.; Yin, Z.; Zhang, X.; Luo, Y. Reversible data hiding based on reducing invalid shifting of pixels in histogram shifting. *Signal Process* **2019**, *163*, 238–246.
- Li, W.; Li, X.; Ni, R.; Zhao, Y. PVO-based reversible data hiding using adaptive multiple histogram generation and modification. *Signal Process Image Commun.* **2021**, *99*, 116405–116415.
- Tian, J. Reversible data embedding using a difference expansion. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 890–896.
- Thodi, D.M.; Rodríguez, J.J. Expansion embedding techniques for reversible watermarking. *IEEE Trans. Image Process* **2007**, *16*, 721–730.
- Ou, B.; Li, X.; Zhao, Y.; Ni, R.; Shi, Y.-Q. Pairwise prediction-error expansion for efficient reversible data hiding. *IEEE Trans. Image Process* **2013**, *22*, 5010–5021.
- Hong, W.; Chen, T.S.; Chen, J.N. Reversible data hiding using Delaunay triangulation and selective embedment. *Inf. Sci.* **2015**, *308*, 140–154.
- Kouhi, A.; Sedaaghi, M.H. Prediction error distribution with dynamic asymmetry for reversible data hiding. *Expert Syst. Appl.* **2021**, *184*, 115475.1–115475.13.
- Puech, W.; Chaumont, M.; Strauss, O. A reversible data hiding method for encrypted images. In *Proceedings of SPIE 2008; SPIE: Bellingham, DA, USA, 2008; Volume 6819*.
- Zhang, X. Reversible data hiding in encrypted image. *IEEE Signal Process Lett.* **2011**, *18*, 255–258.
- Zhang, X. Separable reversible data hiding in encrypted image. *IEEE Trans. Inf. Forensics Secur.* **2012**, *7*, 826–832.
- Hong, W.; Chen, T.S.; Wu, H.Y. An improved reversible data hiding in encrypted images using side match. *IEEE Signal Process Lett.* **2012**, *19*, 199–202.
- Wu, X.; Sun, W. High-capacity reversible data hiding in encrypted images by prediction error. *Signal Process* **2014**, *104*, 387–400.
- Qian, Z.; Zhang, X. Reversible data hiding in encrypted images with distributed source encoding. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *26*, 636–646.
- Qian, Z.X.; Wang, S.Z. Reversible data hiding in encrypted JPEG bitstream. *IEEE Trans. Multimed.* **2014**, *16*, 1486–1491.

30. Qin, C.; Qian, X.; Hong, W.; Zhang, X. An efficient coding scheme for reversible data hiding in encrypted image with redundancy transfer. *Inf. Sci.* **2019**, *487*, 176–192.
31. Fu, Y.; Kong, P.; Yao, H.; Tang, Z.; Qin, C. Effective reversible data hiding in encrypted image with adaptive encoding strategy. *Inf. Sci.* **2019**, *494*, 21–36.
32. Chen, K. High capacity reversible data hiding based on the compression of pixel differences. *Mathematics* **2020**, *8*, 1435.
33. Wang, X.; Chang, C.C.; Lin, C.C.; Chang, C.C. Privacy-preserving reversible data hiding based on quad-tree block encoding and integer wavelet transform. *J. Vis. Commun. Image Represent.* **2021**, *79*, 103203.
34. Ma, K.; Zhang, W.; Zhao, X.; Yu, N.; Li, F. Reversible data hiding in encrypted images by reserving room before encryption. *IEEE Trans. Inf. Forensics Secur.* **2013**, *8*, 553–562.
35. Cao, X.; Du, L.; Wei, X.; Meng, D.; Guo, X. High capacity reversible data hiding in encrypted images by patch-level sparse representation. *IEEE Trans. Cybern.* **2016**, *46*, 1132–1143.
36. Yin, Z.; Xiang, Y.; Zhang, X. Reversible data hiding in encrypted images based on multi-MSB prediction and Huffman coding. *IEEE Trans. Multimed.* **2020**, *22*, 874–884.
37. Puteaux, P.; Puech, W. An efficient MSB prediction-based method for high-capacity reversible data hiding in encrypted images. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 1670–1681.
38. Ren, H.; Lu, W.; Chen, B. Reversible data hiding in encrypted binary images by pixel prediction. *Signal Process* **2019**, *165*, 268–277.
39. Yi, S.; Zhou, Y. Binary-block embedding for reversible data hiding in encrypted images. *Signal Process* **2017**, *133*, 40–51.
40. Mohammadi, A.; Nakhkash, M.; Akhaee, M.A. A high-capacity reversible data hiding in encrypted images employing local difference predictor. *IEEE Trans. Circuits Syst. Video Technol.* **2020**, *30*, 2366–2376.
41. Yang, Y.G.; Zhang, T.C.; Chen, X.B.; Zhou, Y.H.; Shi, W.M. Eliminating the texture features in visually meaningful cipher images. *Inf. Sci.* **2018**, *429*, 102–119.
42. Yang, Y.G.; Wang, B.P.; Yang, Y.L.; Zhou, Y.H.; Shi, W.M.; Liao, X. Visually meaningful image encryption based on universal embedding model. *Inf. Sci.* **2021**, *562*, 304–324.
43. Yang, Y.G.; Wang, B.P.; Yang, Y.L.; Zhou, Y.H.; Shi, W.M. Dual embedding model: A new framework for visually meaningful image encryption. *Multimed. Tools Appl.* **2021**, *80*, 9055–9074.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.