Reversible Data Hiding in Encrypted Images Based on Reversible Integer Transformation and Quadtree-Based Partition

SISHENG CHEN1,2,3 AND CHIN-CHEN CHANG2,4, (Fellow, IEEE)

1School of Electrical and Information Engineering, Fujing Branch of Fujian Normal University, Fuzhou 350300, China
2Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan
3Engineering Research Center for ICH Digitalization and Multi-Source Information Fusion, Fujing Branch of Fujian Normal University, Fuzhou 350300, China
4School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China

Corresponding author: Chin-Chen Chang (alan3c@gmail.com)

This work was supported in part by the Youth Teacher Research Program of Fujian Province under Grant JT180618, and in part by the Natural Science Foundation of Fujian Province, China, under Grant 2018J01788.

ABSTRACT This paper presents an improved secure reversible data hiding scheme in encrypted images based on integer transformation, which does not need using a data hider key to protect the embedded secret data. We first segment the original image into blocks of various sizes based on the quadtree-based image partition. For each block, we reserve m least significant bits (LSBs) of each pixel as embedding room based on the reversible integer transformation. In order to improve the security of the image encryption, we pad the mLSBs of each pixel using the corresponding (8-m) most significant bits (MSBs) information after the transformation, which protects the security of the encryption key. Then, we encrypt the transformed image with a standard stream cipher. After the image encryption, the data hider embeds the secret data in the mLSBs of the encrypted images through an exclusive or operation. On the receiving side, the receiver can extract the secret data after the image decryption and recover the original image without loss of quality. The security analysis shows that the proposed scheme improves the security weakness of the scheme directly using adaptive integer transformation. The experimental results show that the proposed method achieves a higher embedding ratio compared with several relevant methods.

INDEX TERMS Data hiding, encrypted images, integer transformation, quadtree.

I. INTRODUCTION

Data hiding in digital images is an information security technique to hide secret data by modifying the cover images. It is usually used for security data transmission or active image authentication. As we know, the process of data embedding may cause distortion to the original image. However, for some sensitive images, such as medical images or artistic images, permanent distortions of the original images are not allowed. For example, a slight distortion of medical images may cause risk diagnosis errors. Reversible data hiding (RDH) technique is just researched to implement lossless data hiding in digital images, which can recover the original images without loss after extracting the secret data [1]. There are many RDH schemes that have been proposed [2]–[6] that hide the secret data in the plaintext domain. In those schemes, the content of the cover image is visible to the data hider.

Recently, with the development of mobile internet, cloud computing becomes a popular solution for the storage of the images generating from the smart mobile devices. And, for protecting the privacy security, image encryption is usually used in a cloud storage system. That is, the content owners first encrypt the images and then upload the encrypted images to the clouds. For ease of management, the cloud servers generally need to embed some additional information such as user data, authentication data or copyright data into the images. In additional, for some sensitive images, the cloud servers are not allowed to introduce permanent distortion to the original images. For those purposes, the reversible data hiding in encrypted images (RDHEI) is researched and has become a new interesting notion in the data hiding area, which allows embedding additional information directly in encrypted images and recovering the original image after the image decryption and the image recovery. The existing methods to embed additional data in encrypted images are summarized to two categories: reserving room before
encryption (RRBE) and vacating room after encryption (VRAE).

In RRBE methods, the image owners usually pre-process the original images to reserve embedding room before encryption. Schemes [7]–[9] use a self-embedding method to reserve room. They vacate the embedding room in LSBs of some pixels and embed the reversible information of these pixels into other pixels using standard reversible data hiding methods. This method has a lower embedding capacity because only some of pixels are used to carry the secret data. In schemes [10], [11], they reserved the embedding room using a reversible integer transformation. They transformed the adjacent two pixel values to be both even or odd, and the two pixels carry one bit secret data in the data embedding processing. Schemes [12]–[15] reserved the embedding room in the MSBs of the pixel values combined with prediction error methods. Malik et al. [15] used the prediction-error method to reserve the embedding room in the MSBs. They used the prediction-error values to replace the pixel values in the prediction block. When the error is small, the MSBs of the prediction-error values are zeros, and a location map is used to record these pixels. Then, they embed the secret data in the MSBs of these pixels in the encrypted image. Chen et al. [16] used pre-coding method to reserve the embedding room before encryption and embedded the secret data with matrix embedding methods.

In VRAE methods, the data hider vacates the embedding room in the encrypted images. There are two main categories in this method. The first category is to keep some redundant space in the encrypted image, and then the room is vacated directly in the cipher image with different methods [17]–[23]. Scheme [17] transferred the redundant space for data embedding by exchanging the MSBs and LSBs of the pixel values before image encryption. Then, they used a coding method to vacate room in the LSBs of pixels in the cipher image blocks. Scheme [18] used a special encryption algorithm to keep the local characteristics in the encrypted image, then compressed the bits from the cipher image to make room for the data embedding using a different compressing method. Schemes [20], [21] used a parameter binary label to compress room for embedding data. Ge et al. [24] kept the histogram of each block in the encrypted image the same as the original image using block based encryption. Then, they embedded the secret data in the encrypted image using the histogram shifting method. Di et al. [25] proposed a bitplane operation and adaptive embedding method to improve the embedding capacity of the existing schemes. The first method can obtain considerable embedding capacity, but the security of the cipher image is poor because the cipher image entropy is not maximized. The second method vacates the room based on the property of encryption algorithms. Huang and Wang [26] combined the MED prediction and specific encryption algorithm to reserve the embedding room for data hiding in the encryption process. Schemes [27], [28] used the homomorphism of the encryption algorithms to embed the secret data by modifying the cipher pixel values without affecting the decryption. These methods can obtain a higher embedding capacity because they don’t consider the quality of the stego-images. However, they expand the size of the encrypted image.

From the above, we know most RDHEI schemes not matter based on RRBE or VRAE have to pre-process the original image for vacating embedding room before encryption. The pre-processing may lead to security weakness of the image encryption, especially reserving the room in the LSBs or MSBs of the cover pixels. Qiu et al. [29] proposed an RDHEI scheme using adaptive integer transformation, which has a high embedding capacity. They divided the original image into blocks. For each block, they transferred the pixel values using a reversible integer transformation with an adaptive parameter \( m \). After the transformation, the \( m \) bits LSBs of each pixel value in the block are zero. Then they encrypted the transformed image with the stream cipher, and embedded the secret data in the \( m \) bits LSBs of each pixel in the cipher image block. We noticed that if the \( m \) bits LSBs of a pixel value are zero, then the \( m \) bits of the LSBs of pixel values in the cipher image will disclose the \( m \) bits of the encryption key. This means that the attacker can obtain part bits of the encryption key. This is a weakness of the cipher images security. Chen et al. [11] classified the existing RDHEI schemes into two types by the numbers of secret keys that the receiver shares with the image owner and data hider: sharing independent secret key (SIK) and sharing no secret key (SNK). Then, they presented a new type of RDHEI, in which the receiver shares only one secret key (SOK).

This paper presents a new SOK-type RDHEI scheme that improves the security of the cipher image based on the reversible integer transformation proposed by [29], which is a SIK-type scheme. We first segment the original image using a quadtree-based partition algorithm. The pre-condition of the image segmentation is to satisfy the demand of the reversible integer transformation. Then, to prevent disclosure of the encryption key information, we pad the \( m \) bits LSBs of each pixel in the encrypted image, then we encrypt the secret data in the padded LSBs. After the image encryption, the receiver decrypts the marked encrypted image and extracts the secret data by removing the padding information based on the right MSBs of each decrypted pixel value. Finally, we can recover the original image using the inverse transformation.

II. RELATED WORKS
In this section, we first introduce the reversible integer transformation proposed in [29], and analyze the weakness of an RDHEI scheme directly using the integer transformation. Then, we introduce an image partition method based on the quad-tree.
A. REVERSIBLE INTEGER TRANSFORM AND ANALYSIS

Qiu et al. [29] proposed a reversible integer transformation that transforms an integer vector \( p = (p_1, p_2, \cdots, p_n) \in \mathbb{Z}^n \) to another integer vector \( p' = (p'_1, p'_2, \cdots, p'_n) \in \mathbb{Z}^n \) with parameter \( m \). After transformation, the mLSBs of all \( p'_i \) in \( p' \) are zeros. The integer transformation is described as follows:

\[
p'_i = \varphi(p_i) = 2^m \times p_i - 2^m \times f(r(p), 2^m),
\]

where \( f(r(p), 2^m) = \lfloor (2^m - 1) \times r(p)/2^m \rfloor, r(p) = \text{round}(p) \) and \( p = (\sum_{i=1}^{n} p_i)/n \). The inverse transformation is:

\[
p_i = \varphi^{-1}(p'_i) = 2^m \times (h(p'_i, 2^m) + (2^m - 1) \times r(h(p'_i, 2^m)))
\]

\[+ l(r(p), 2^m),
\]

where \( h(p'_i, 2^m) = p'_i/2^m, h(p', 2^m) = p'/2^m, \) and \( l(r(p), 2^m) \) denote the mLSBs of \( r(p) \) that is recorded in the processing of transformation (1).

The mean square error between \( p \) and \( p' \) is defined as:

\[
\text{MSE}(p, p') = \sum_{i=1}^{n} (p_i - p'_i)^2/n,
\]

and the mean square error of \( p \) is defined as:

\[
\text{MSE}(p) = \sum_{i=1}^{n} (p_i - \bar{p})^2/n,
\]

where \( \bar{p} \) denotes the average of \( p \), and \( \bar{p} = \sum_{i=1}^{n} p_i/n \).

This means that we can control the transformation error by adjusting parameter \( m \) or the mean square error of the integer vector.

For example, we consider \( p = (161, 162, 160, 163) \) and \( m = 2 \), compute \( r(p) = 162, f(r(p), 2^2) = \lfloor 3 \times 162/4 \rfloor = 122 \), then \( p'_2 = 4p_2 - 4 \times 122 = (156, 160, 152, 164) \). It is obvious that the 2LSBs of each \( p'_i \) in \( p' \) are zeros. If we know the 2LSBs of \( r(p) \), we can transfer \( p \) to \( p' \). We first compute \( h(p', 2^2) = p'/4 = (39, 40, 38, 41) \) and \( l(r(p), 2^2) = 40 \), then recover \( p = (39, 40, 38, 41) + 3 \times 40 + 2 = (161, 162, 160, 163) \).

The integer transformation is used to reserve embedding room in the scheme [29]. Here, we analyze the weakness of this method. For example, considering the original pixel value is \( (161, 162, 160, 163) \), then they are transformed to \( (156, 160, 152, 164) \) according to (1). The binary representation of the transformed result is \( 10011100, 10100000, 10011000, 10100100 \), where the 2LSBs of each pixel is 0. In the image encryption, assuming that the encryption key of the stream cipher is \( 10010001, 00110001, 01010000, 10001100 \), the encryption result is \( 00111110, 10111001, 11010000, 00101000 \). It is obvious that the 2LSBs of cipher-text disclose the 2LSBs of the encryption key. Therefore, it is important to post-process the reserving embedding room before image encryption, which will be presented in the proposed scheme.

B. IMAGE PARTITION BASED ON QUADTREE

The image partition means that it divides the images into non-overlapped blocks. The usual method is to divide an image into solid sized blocks. With a quad-tree based partition, an image block is divided into a series of non-overlapped blocks with various sizes, and a quad-tree corresponds to the image partition processing which records the size and location of each block. It is widely used in the image processing to compress the image partition information [30]–[32].

The image partition based on the quad-tree can be simply described as follows. For the given image \( I \) sized \( 2^n \times 2^n \), decide whether the whole image block satisfies the pre-defined condition \( C \). If \( I \) satisfies condition \( C \), then the partition processing of the current block is complete. If not, then \( I \) is divided into four sub-image blocks sized \( 2^{n-1} \times 2^{n-1} \). Construct a 1-bit node as the root node, and take the four sub-blocks as the four children nodes of the root node. For the four children blocks, decide whether they satisfy condition \( C \) in a left-to-right, top-to-bottom order. If not satisfy the condition, then the sub-block is further split in four smaller image blocks. Similar processing is done to each smaller sub-image blocks individually. In each node, a value of 1 represents that the corresponding block satisfies the condition \( C \) and a value of 0 represents not satisfying condition \( C \). The partition is repeated until all sub-blocks satisfy the condition or the size of the sub-block is equal to the pre-defined size. Finally, we can obtain a quad-tree in which each node denotes a sub-block of the image \( I \). The position of a node in the tree determines the size and the position of a sub-block, and the value determines whether the sub-block satisfies the pre-condition. Fig. 1 shows an example of an image partition processing and its corresponding quadtree.

III. PROPOSED SCHEME

The designed scheme contains three stages. The first stage contains the embedding room reserve and image encryption. The second stage is data hiding. The third stage consists of the data extraction and image recovery. Fig. 2 shows the framework of the three stages, and the detailed processing is described in the following subsection.

A. RESERVE EMBEDDING ROOM

In our scheme, we use integer transform (1) to reserve the embedding room. Unlike scheme [29] that adaptively chooses block size \( m \), we choose a solid \( m \) for the integer transform and adjust the size of the blocks. We use the quadtree-based image partition to adjust the size of the blocks. When transfer the pixel vector \( p = (p_1, p_2, \cdots, p_n) \) of a image block to \( p' = (p'_1, p'_2, \cdots, p'_n) \), to prevent the underflow/overflow in the transform, we must ensure the transformed values belonging to \([0, 255]\). So, according to (1), the pixel values must satisfy the following condition:

\[
0 \leq p_i - f(r(p), 2^m) \leq 2^{8-m}.
\]

Additionally, according to (3), to preserve the distortion of the transform, the mean square error of the vector \( p \) must satisfy:

\[
s^2(p) \leq T/(2^m - 1)^2,
\]

where \( T \) is a threshold value and \( \text{MSE}(p, p') \leq T \). Thus, for the given \( m \), we must choose a suitable block size to
make the pixel vector \( p = (p_1, p_2, \cdots, p_n) \) satisfy the following condition \( C \):

\[
0 \leq p_i - f(r(p), 2^m) \leq 2^{8-m} (\forall p_i \in p) \quad \text{and} \quad s^2(p) \leq T/(2^m - 1)^2. \tag{4}
\]

So, we use condition \( C \) to operate the quadtree-based image partition in an original image. The process of reserving the embedding room is described as follows.

1. **Step 1**: Generate a 1-bit size label as the root node of the quadtree \( LT \). Determine whether the pixel values vector in the whole image satisfy the condition \( C \). If yes, set the value of the root node to 1, and terminate the partition process. If not, set the value to 0, and go to the next step.

2. **Step 2**: Divide the current image block into four sub-image blocks, determine whether the pixel value vectors of each sub-block satisfy condition \( C \) in a left-to-right, top-to-bottom order. Generate 4 label nodes to record the determined results. If a sub-block satisfies the condition \( C \), then set the corresponding label value to 1. If no, then set it to 0. Take the four label nodes as the child nodes of the current label node.

3. **Step 3**: For each block of the four child blocks, assuming that there are \((j - 1)\) blocks that have been confirmed, then:

   - **Case 1**: If its label value is 1, the block is represented as \( pb_i^{(j)} \), and we take the pixel values to form an integer vector \( pb_i^{(j)} = (pb_i^{(j)}(1), pb_i^{(j)}(2), \cdots, pb_i^{(j)}(n)) \). Then, we transform it to \( pb_i^{(j)}(1), pb_i^{(j)}(2), \cdots, pb_i^{(j)}(n) \) using the integer transform (1), where \( pb_i^{(j)} = \varphi(pb_i^{(j)}(1), \cdots, pb_i^{(j)}(n)) \). Reshape \( pb_i^{(j)} \) to an image block \( pb_i^{(j)} \), and record the \( m \)-LSBs of \( r(pb_i^{(j)}) \) in \( LSB_i \). Record the location and size of this block in the blocks table \( BT(j) \) for the next encryption process and end the partition in this block.

   - **Case 2**: If its label value is 0 and the size of the block is equal to the pre-defined minimum size, then keep the pixel values in this block unchanged and end the partition in this block. Record the location and size of this block in \( BT(j) \).

   - **Case 3**: If its label value is 0 and the size of the block is larger than the pre-defined minimum size, then go to Step 2 to continue dividing this block to four smaller sub-blocks.

After completing the image partition according to the condition \( C \), we can get a label tree \( LT \), like the example shown in Fig. 1. The leaf nodes of \( LT \) denote all the blocks of the image. The original image consisting all the blocks of the image is denoted as \( I = \{pb_1^{(1)}, \cdots, pb_N^{(N)}\} \), and the transformed image is denoted as \( I' = \{pb_1^{(1)}', pb_2^{(2)}', \cdots, pb_N^{(N)}'\} \).
where \( N \) is the amount of blocks. In \( pb_{i}^{(0)} \), the mLSBs of the pixel values in those blocks whose label values are 1 are all zeros, which is the embedding room reserved by integer transformation. Meanwhile, the block table \( BT \) records the location and size information of each block. The data hider and receiver can obtain the block table \( BT \) again by operating the image partition according to the label tree \( LT \). So, we only need to embed \( LT \) information in the encrypted image.

**B. IMAGE ENCRYPTION**

According to the block table \( BT \), the image owner encrypts the transformed image \( I' \) block by block. If we directly encrypt it with a stream cipher, then the mLSBs of the encrypted pixel values of blocks whose label values are 1 will disclose the information of the encryption key. Therefore, to prevent disclosure of the encryption key information, we pad messages in the mLSBs of pixel values before encryption. The encryption processing is presented as follows.

**Step 1 (mLSBs Padding):** For each block \( pb_{i}^{(0)} \) whose label value is 1, the mLSBs of pixel values in those blocks are zeros. The pixel value \( pb_{i}^{(0)} \) in each block \( pb_{i}^{(N)} \) is represented as 8 bits binary \( pb_{i}^{(0)} = b_{8}b_{7} \ldots b_{m+1}0 \ldots 0 \). Compute:

\[
b'_{m} \ldots b'_{2}b'_{1} = b_{8}b_{7} \ldots b_{8-m+1} \oplus b_{m+1}b_{m+2} \ldots b_{2m},
\]

then the padding result is \( pb_{i}^{(0)} = b_{8}b_{7} \ldots b_{m+1}b_{m} \ldots b'_{1} \). After all the blocks have been processed, we obtain the new image \( I'' = (pb_{1}^{(1)}, pb_{2}^{(2)}, \ldots, pb_{N}^{(N)}) \). Fig. 3 shows an example of mLSBs padding: here, the padding messages are computed by \( b'_{2}b'_{1} = b_{8}b_{7} \oplus b_{3}b_{4} \).

**Step 2 (Image Encryption):** Here, we apply the standard stream cipher to encrypt the image. We use the sharing key with the receiver as the seed key of a pseudorandom number generator to generate the encryption key sequence \( Key = \{key_{1}, key_{2}, \ldots, key_{N}\} \), where \( key_{j} \) is a key block with the same size as block \( pb_{j}^{(0)} \) and \( 1 \leq j \leq N \). The pixel value \( pb_{j}^{(0)} \) in block \( pb_{i}^{(0)} \) is encrypted with \( key_{j} \) as follows:

\[
ch_{i}^{(0)}(b) = \text{bitxor}(pb_{i}^{(0)}, key_{i}^{(0)}) = pb_{i}^{(0)}(b) \oplus key_{i}^{(0)}(b),
\]

where \( b = 1, 2, \ldots, 8, pb_{i}^{(0)}(b) \) represents the \( b \)-th bit of \( i \)-th pixel values in the block \( pb_{i}^{(0)} \), and \( key_{i}^{(0)}(b) \) represents the \( b \)-th bit of the \( i \)-th key in the key block \( key_{i}^{(0)} \). When all the blocks have been encrypted, we obtain the encrypted image \( I_{c} = \{cb_{1}, cb_{2}, \ldots, cb_{N}\} \).

**Step 3 (Auxiliary Data Self-Embedding):** In order to implement the data hiding and decryption, the encrypted image must carry the label tree \( LT \). Convert the label tree \( LT \) to binary sequence \( LT_{b} \) in the breadth first order as parts of the auxiliary information. Then, to recover the original image, the mLSBs of the mean values of each transformed block, which are recorded in \( LSB_{1} \), are required. So, we first compress the auxiliary data consisted of \( LT_{b} \) and \( LSB_{1} \) losslessly, then embed it in the fixed square area in upper left of the encrypted image by mLSBs substitution. When the auxiliary data is embedded in blocks whose label is 1, we do not need addition operation because the mLSBs of the pixels in those blocks are all zeros. If the pixels carrying auxiliary data belong to the blocks whose label is 0, we record their original mLSBs in \( LSB_{0} \), then \( LSB_{b} \) is taken as parts of the secret data to be embedded in the encrypted image using the data embedding processing.

**C. SECRET DATA EMBEDDING**

The data hider embeds the secret data in the encrypted image \( I_{c} \) by the XOR operation, which contains three steps. The detailed data embedding process is described as follows.

**Step 1 (Image Partition):** Extract the auxiliary data \( LT_{b} \) from the mLSBs of pixels in the upper left square area of the encrypted image, and convert \( LT_{b} \) into label tree \( LT \). According to the label tree \( LT \), segment the encrypted image \( I_{c} \) again to obtain the block table \( BT \). According to \( BT \), divide \( I_{c} \) into a series of blocks, and denoted as \( I_{c} = \{cb_{1}, cb_{2}, \ldots, cb_{N}\} \). Then, divide it into two parts \( I_{c} = I_{c_{0}} \cup I_{c_{1}} \), where \( I_{c_{1}} \) denotes the blocks whose label values are 1, and \( I_{c_{0}} \) denotes the blocks whose label values are 0.

**Step 2 (Blocks Sorting):** After the image partition, according to \( LT \), obtain the distribution graph of blocks and label values. According to the distribution graph, sort all the embeddable blocks without carrying the auxiliary data starting from the larger blocks first, top to bottom, and left to right, and then obtain the sorted embeddable block set \( I_{c_{1}} = \{cb^{(1)}, cb^{(2)}, \ldots, cb^{(N)}\} \), where \( N_{1} \leq N \). Fig. 4 shows a blocks sorting example in a 16 × 16 sized image, in which
Fig. 4.a shows distribution graph of blocks without carrying the auxiliary data, and Fig. 4.b shows the sorting result, where the numbers in the blocks represent their order.

**Step 3 (Data Embedding):** Convert the binary secret data into a 2^m-ary integer sequence and divide it into a set of blocks \( s = \{s(1), s(2), \ldots, s(L)\} \) according to \( I_{s1} \), where \( L \leq N_1 \). The \( i \)-th secret digit \( s_i \) in block \( s(i) \) will be embedded into the pixel \( \text{cb}_i^{(1,j)} \) in the block \( \text{mc}_i^{(1,j)} \) as follows:

\[
\text{mc}_i^{(1,j)}(b) = \text{bitxor}(\text{cb}_i^{(1,j)}, s_i) = \text{cb}_i^{(1,j)}(b) \oplus s_i(b),
\]

where \( b = 1, 2, \ldots, 8 \). Finally, we obtain the marked encrypted image \( I_{mc} = I_{c0} \cup I_{mc1} = \{\text{mc}_1^{(1,j)}, \text{mc}_2^{(1,j)}, \ldots, \text{mc}_N^{(N)}\} \). Fig. 5.a shows an example of the image encryption, in which the block after the mLSBs padding is \( \text{pb}_i^{(1,j)} \), and \( \text{cb}_i^{(1,j)} = \text{bitxor}(\text{pb}_i^{(1,j)}, \text{key}_i^{(j)}) \). Fig. 5.b shows an example of data embedding, where we choose \( m = 2 \).

### D. DATA EXTRACTION AND IMAGE RECOVERY

When receiving the marked encrypted image \( I_{mc} \), the receiver extracts the mLSBs of the pixels in the upper left fixed square area of the encrypted image to obtain the auxiliary data \( LT \), LSB1. According to the tree \( LT \), divide the encrypted image \( I_{mc} \) to reconstruct the block table \( BT \) again. According to \( BT \), divide \( I_{mc} \) into \( I_{mc} = \{\text{mc}_1^{(1,j)}, \text{mc}_2^{(1,j)}, \ldots, \text{mc}_N^{(N)}\} \).

**Step 1 (Decryption):** Use the sharing key with the content owner as the seed key of shared pseudorandom number generator to generate the decryption key sequence \( \text{Key} = \{\text{key}_1^{(1)}, \text{key}_2^{(2)}, \ldots, \text{key}_N^{(N)}\} \), where \( \text{key}_i^{(j)} \) is a key block with the same size as block \( \text{mc}_i^{(j)} \) and 1 \( \leq j \leq N \). Decrypt the pixel \( \text{mc}_i^{(j)} \) in the block \( \text{mc}_i^{(j)} \) of the marked encrypted image \( I_{mc} \) with the decryption key \( \text{key}_i^{(j)} \) as follows:

\[
\text{sb}_i^{(j)}(b) = \text{mc}_i^{(j)}(b) \oplus \text{key}_i^{(j)}(b), \quad b = 1, 2, \ldots, 8.
\]

After all the pixels have been decrypted, obtain a stego-image \( I_s = \{\text{sb}_1^{(1)}, \text{sb}_2^{(2)}, \ldots, \text{sb}_N^{(N)}\} \). According to \( LT \), divide \( I_s \) into two parts: non-embeddable blocks set \( I_{s0} \) and embeddable block set \( I_{s1} \), that is \( I_s = I_{s0} \cup I_{s1} \). The non-embeddable blocks are not modified in the integer transformation and data embedding process, so \( I_{s0} \) is part of the original image, which is denoted by \( I_0 \). Sort all the embeddable blocks in \( I_{s1} \), and obtain the embeddable block set that is denoted by \( I_{s1} = \{s(1), s(2), \ldots, s(N_1)\} \), where \( N_1 \leq N \).

**Step 2 (Secret Data Extraction):** For the pixel value \( \text{sb}_i^{(j)} \) in the stego-block \( \text{sb}_i^{(j)} \), according to (6), (7), and (8), we can verify that:

\[
\text{sb}_i^{(j)} = \text{bitxor}(\text{pb}_i^{(j)}, \text{key}_i^{(j)}).
\]

The binary representations of \( \text{sb}_i^{(1)}, \text{pb}_i^{(1)}, \text{key}_i^{(j)} \), and \( s_i \) are \( b_1b_2 \ldots b_{m+1}b_m \ldots b_1, b_2b_7 \ldots b_{m+1}b_m' \ldots b_1', s_i^{(m)}(m) \ldots s_i^{(m)}(2)s_i^{(1)}(1) \), respectively, thus:

\[
\text{s}_i^{(m)}(m) \ldots \text{s}_i^{(m)}(2)\text{s}_i^{(1)}(1) = b_1b_2b_1 \oplus b_2b_1' \ldots b_2b_1'.
\]

According to (5), we know

\[
b_1b_2b_1 = b_8b_7 \ldots b_8m+1 \oplus b_7m+1b_m+2 \ldots b_2m.
\]

Therefore,

\[
\text{s}_i^{(m)}(m) \ldots \text{s}_i^{(m)}(2)\text{s}_i^{(1)}(1) = b_1b_2b_1 \oplus b_8b_7 \ldots b_8m+1 \oplus b_7m+1b_m+2 \ldots b_2m.
\]

When all the embedded blocks have been processed, the secret data \( s = \{s(1), s(2), \ldots, s(L)\} \) is extracted. Then, \( \text{LSB}_0 \) is split from \( s \) to recover the mLSBs of the pixel values in a block that carry auxiliary data and whose labels are 0.

**Step 3 (Image Recovery):** After all the secret data have been extracted, set the mLSBs of each pixel in \( I_{s1} \) to zeros and obtain \( I'_i = \{pb_1^{(1)}, pb_2^{(1)}, \ldots, pb_N^{(N)}\} \). For the pixel values vector \( pb_1^{(1)}, pb_2^{(1)}, \ldots, pb_N^{(N)} \), use the inverse integer transformation (2) with the recorded information \( \text{LSB}_1 \) to recover pixel block \( \text{pb}_1^{(1)} = \ldots \text{pb}_N^{(N)} \).
FIGURE 6. Illustration of data extraction and image recovery.

\[ s_h^{(i,j)} \] Binary representation

\[
\begin{array}{cccc}
1010001 & 1010010 & 1010000 & 1011011 \\
1001111 & 1001010 & 1001000 & 1000101 \\
1001110 & 1001001 & 1001000 & 1001001 \\
1001101 & 1001001 & 1001000 & 1001001 \\
\end{array}
\]

Set mLSBs to zeros

\[
\begin{array}{cccc}
1010000 & 1010010 & 1010000 & 1011000 \\
1001111 & 1001010 & 1001000 & 1000101 \\
1001110 & 1001001 & 1001000 & 1001001 \\
1001101 & 1001001 & 1001000 & 1001001 \\
\end{array}
\]

\( p_h^{(i,j)} \)

Decomposition

\[
\begin{array}{cccc}
01 & 01 & 10 & 01 \\
10 & 01 & 01 & 10 \\
01 & 01 & 00 & 01 \\
11 & 01 & 10 & 01 \\
\end{array}
\]

Inverse integer transform (2)

\[
\begin{array}{cccc}
162 & 163 & 162 & 160 \\
161 & 163 & 162 & 161 \\
161 & 163 & 162 & 161 \\
161 & 162 & 163 & 163 \\
\end{array}
\]

\( p_h^{(i,j)} \)

Decimal representation

\[
\begin{array}{cccc}
160 & 164 & 160 & 152 \\
156 & 164 & 160 & 156 \\
156 & 164 & 160 & 160 \\
156 & 160 & 164 & 164 \\
\end{array}
\]

\( s_h^{(i,j)} = (b_1, b_2, b_3, b_4, b_5, b_6) \), where, \( p_h^{(i,j)} = \varphi^{-1}(p_h^{(i,j)′}) \) and \( i = 1, 2, \ldots, n \). After all the blocks have been transformed, we get \( I_1 = \{p_h^{(1,1)}, p_h^{(1,2)}, \ldots, p_h^{(1,n)}\} \). Finally, we can recover the original image \( I = I_0 \cup I_1 \) and \( BT \).

Fig. 6 shows an example to illustrate the data extraction and image recovery. The size of the marked encrypted image block \( s_h^{(1,j)} \) is \( 4 \times 4 \), and \( m = 2 \).

IV. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we will evaluate and analyze the performance of the proposed scheme. We first analyze the security of the mLSBs of the encrypted image. Then, we evaluate the performance of the proposed method on some standard test images and 10000 images from BOWS-2 database [33], and present the comparison results with some relevant works.

A. SECURITY ANALYSIS

The security analysis of the RDHEI scheme contains two aspects. The first is to analyze the security of the encrypted images, where we demonstrate that the process of pre-processing and data embedding don’t affect the statistical security of encrypted images. The second is to analyze the security of the secret data, where we demonstrate that an attacker can’t disclose the secret data without the decryption key.

In the proposed scheme, because of using the stream cipher algorithm to encrypt the image, we can guarantee the randomness of encrypted pixels base on pseudo randomness of encryption key. Thus, we can conclude that the proposed scheme can resist the statistical analysis attack with security encryption key. In Section II-A, we have pointed out the security weakness of directly using the reversible integer transformation (1) to reserve embedding room resulting from the mbits LSBS of pixel values in the transformed image are zeros. If we directly encrypt the image in the transformed image, then the mLSBS of the pixel values in the encrypted image disclose the mbits information of the encryption key.

In our scheme, we use the (8-m)MSBs of the transformed pixels that are not modified in the data embedding process to generate a padding message. Then, we use the padding message to pad the m bits LSBS of the transformed pixel values. Thus, the m LSBS of those pixel values in the encrypted image are the ciphertexts of the padding message. This method allows us to keep the randomness of the cipher image. The attackers can’t disclose the (8-m)MSBs information without the encryption key, so they can’t predict the m bits padding data. Therefore, we can ensure the security of the encryption key.

In the proposed scheme, the receiver only shares one secret key. In the data embedding process, the secret data are embedded in the m LSBS of the embeddable pixel values. This means that the secret data are encrypted with the padding message as encryption key. Here, we must point out a reasonable assumption that an attacker will not capture and save the encrypted image if they can’t detect the intention of data hiding. Therefore, when an attacker obtains the marked encrypted image and obtains the auxiliary data \( LT \), they can know the image partition and their label values, but they can’t extract secret key without image encryption key because they can’t obtain the right padding message. And, a receiver who has the image encryption key can decrypt the right (8-m) bits MSBS of each pixel value and remove the padding message in the data extraction. In all, we use the LSBS padding method to improve the security of the image encryption, and which also can guarantee the security of the embedded message without a data hiding key.

B. ANALYSIS RESULTS FOR SOME EXPERIMENTS

In this section, we implement a sequence of experiments to evaluate the proposed scheme using some test images. The test images contain six standard images and 10000 images from Bows-2 database. All the test images are sized 512 × 512.
Fig. 7 shows the experimental results of different stages of the proposed scheme in three test images. Here, we choose the parameters \( m = 1 \) and \( T = 20 \). Fig. 7.c, Fig. 7.g, and Fig. 7.k are directly decrypted images. Fig. 7.d, Fig. 7.h, and Fig. 7.l are recovery images. As shown in Figure 7, we can obtain the stego-image after image decryption. Comparing with the original images, the PSNR values of the recovery images are near \(+\infty\), which means that our scheme is reversible.

In the proposed scheme, a pixel in the embeddable blocks of \( I_{c1} = \{ cb^{(1,1)}, cb^{(1,2)}, \ldots, cb^{(1,N_1)} \} \) carries \( m \) bits secret data. Assume that the total pixels of block \( cb^{(1,j)} \) is \( L_{cb}^{(1,j)} \). In each block, we must use \( m \) bits to record the \( m \)-bits LSBs of the mean values of the block \( pb^{(1,j)} \) in the integer transformation processing or recovery image. In addition, we also record the label tree for the image partition, and the length of label is denoted as \( L_{LT} \). Therefore, the embedding capacity \((EC)\) and embedding ratio \((ER)\) of proposed scheme are

\[
EC = \sum_{j=1}^{N_1} (mL_{cb}^{(1,j)} - m) - L_{LT} \quad \text{(bit)} ,
\]

\[
ER = \frac{EC}{(W \times H)} \quad \text{(bpp)} ,
\]

where, \( W \times H \) is the size of the original image.

Here, we analyze the embedding capacity of the proposed scheme from two aspects. The first is to evaluate the directly encrypted images quality with different payloads. We conduct the experiment in six test images, and Fig. 8 shows the experimental results. The image quality is maintained well with a low payload, then declines as the payload increase. In the high payload stage, the image quality is not very good but it is acceptable. It also performs better for the smooth images than the complex images. The second aspect is to evaluate the maximum embedding capacity. We first take the six test images to calculate the embedding capacity according to (11) and (12), and Table 1 shows the experimental results. When the parameter \( m = 1 \), which means reserving a 1-bit embedding room in a pixel, the maximum embedding ratio is all over 0.9 bpp in all test images. The first four test images reach the maximum embedding capacity when \( m = 4 \). In addition, the larger \( m \) may lead to a decrease in the amounts of embeddable blocks, and thereby the embedding capacity is lower. Like in Lake and Wine, the embedding capacity is declining when \( m \) increases from 3 to 4. Of course, the directly decrypted image quality will decline as the payload increases. From the experimental results shown in Fig. 7 and Fig. 8, we can conclude that the proposed scheme has a high embedding capacity with an acceptable directly decrypted image quality.

To further evaluate the embedding capacity of the proposed scheme, we perform the scheme on 10000 images from the Bows-2 database [33]. Table 2 shows the experimental results in terms of quartiles (1st quartile, median, and 3rd quartile) and mean value. When \( m = 1 \), the embedding ratio is higher than 0.9004 bpp for 75% of images, and higher than 0.9750 bpp for 25% of images. When \( m = 2 \), the embedding ratio is higher than 1.4731 bpp for 75% of images, and higher than 1.8161 bpp for 25% of images. When \( m = 3 \), the mean embedding ratio is 1.9314 bpp, and the embedding ratio is
higher than 2.4014 bpp for 25% of images. When \( m = 4 \), we can get higher embedding ratio in some smooth images, but lower in some images with complex texture, and the mean value is close to mean value with \( m = 3 \).

### C. COMPARISON EXPERIMENTS AND ANALYSIS

In Section IV-A we have shown that the proposed scheme improves the security of the cipher image compared to the scheme [29]. In addition, different from adjusting the parameter \( m \) in the solid-sized blocks to satisfy the predefined condition, we try to adjust block sizes by a quad-tree based image partition with the solid parameter \( m \) in our scheme. We hope to use this method to reduce the auxiliary data and improve the embedding capacity. Table 3 shows the experimental results. We choose the \( 4 \times 4 \) block size in [29] which performs better than other block models. The experimental results show that our method obtains higher embedding capacity in the six test images when the parameter \( m = 1 \) and \( m = 2 \). To further verify the embedding capacity, we experiment on 10000 images from BOWS-2 database to evaluate the mean embedding capacity. The mean embedding ratio in [29] is 0.8555 bpp when \( \max(m) = 1 \), and 1.5592 bpp when \( \max(m) = 2 \). And, the mean embedding ratio of the proposed scheme is 0.9248 bpp when \( m = 1 \), and 1.6107 bpp when \( m = 2 \).

Next, we compare our scheme with other relevant methods [7], [11], [15], [24]–[26] in embedding capacity. Chen et al. [11] also reserved the embedding room using a reversible...
integer transformation, and it is also a SOK-type scheme. In [11], two adjacent pixels carry a bit secret data, so the max embedding capacity of this scheme is 0.5 bpp. Malik et al. [15] reserved the embedding room in the MSBs using the prediction-error method, and the max embedding capacity is about 0.74 bpp. The experimental results show that the proposed scheme has a higher embedding capacity than these two schemes. Di et al. [25] used the bit-plane operation and adaptive embedding method to improve the embedding capacity of the existing schemes. Scheme [24] divided the encrypted image into blocks and embedded the secret data in each block using the histogram shifting method. Huang and Wang [26] combined the MED prediction and specific encryption algorithm to reserve the embedding room for data embedding in the encryption processing and obtained a higher embedding capacity than the above schemes. The detailed comparison results are shown in the Table 4. The proposed scheme has a higher embedding capacity than other schemes in most test images.

V. CONCLUSION

This paper proposed an SOK-type RDHEI scheme using a reversible integer transformation and quad-tree-based image partition, in which we do not need to use a data hiding key to protect the security of the embedded messages. We used the reversible integer transformation based on the various sized blocks to reserve the embedding room. To prevent the LSBs of the pixel values in the cipher image from disclosing the encryption key, we padded information in the mLSBs of the cover pixels after reserving the embedding room. The data hider can embed the secret data directly in the encrypted image through an XOR operation without a data hiding key. The receiver can extract the secret data and recover the original image with the encryption key. The security analysis shows that the proposed scheme improves the security of the image encryption which reserves the embedding room in the LSBs of original images. In addition, the experimental results in the six test images and 10000 images from the BOWS-2 database show that the proposed scheme can obtain a high embedding capacity. Therefore, this RDHEI scheme is can be used to provide the data hiding and image privacy, in which the receiver only share key with content owner. However, the visual quality of directly decrypted image is also important in some application scenes.

In this paper, we mainly focus on the improvement of the embedding capacity and security. Of course, there are requirements for the visual quality of directly decrypted image in some application scenarios. In future work, we are interested in finding an effective way to improving the visual quality of the directly decrypted image, and constructing SOK-type RDHEI schemes based on the homomorphism of encryption algorithm.

REFERENCES

[1] Y.-Q. Shi, X. Li, X. Zhang, H.-T. Wu, and B. Ma, “Reversible data hiding: Advances in the past two decades,” IEEE Access, vol. 4, pp. 3210–3237, Apr. 2016.
[2] Z. Ni, Y. Q. Shi, N. Ansari, and W. Su, “Reversible data hiding,” IEEE Trans. Circuits Syst. Video Technol., vol. 16, no. 3, pp. 346–354, Mar. 2006.
[3] Y. Hu, H.-K. Lee, K. Chen, and J. Li, “Difference expansion based reversible data hiding using two embedding directions,” IEEE Trans. Multimedia, vol. 10, no. 8, pp. 1500–1512, Dec. 2008.
[4] X. Zhang, “Reversible data hiding with optimal value transfer,” IEEE Trans. Multimedia, vol. 15, no. 2, pp. 316–325, Feb. 2013.
[5] Y. Qiu, Z. Qian, and L. Yu, “Adaptive reversible data hiding by extending the generalized integer transformation,” IEEE Signal Process. Lett., vol. 23, no. 1, pp. 130–134, Jan. 2016.
[6] J. Wang, J. Ni, X. Zhang, and Y. Q. Shi, “Rate and distortion optimization for reversible data hiding using multiple histogram shifting,” IEEE Trans. Cybern., vol. 47, no. 2, pp. 315–326, Feb. 2017.
[7] K. Ma, W. Zhang, X. Zhao, N. Yu, and F. Li, “Reversible data hiding in encrypted images by reserving room before encryption,” IEEE Trans. Inf. Forensics Security, vol. 8, no. 3, pp. 553–562, Mar. 2013.
[8] W. Zhang, K. Ma, and N. Yu, “Reversibility improved data hiding in encrypted images,” Signal Process., vol. 94, pp. 118–127, Jan. 2014.
[9] S. Xiang and X. Luo, “Reversible data hiding in homomorphic encrypted domain by mirroring ciphertext group,” IEEE Trans. Circuits Syst. Video Technol., vol. 28, no. 11, pp. 3099–3110, Nov. 2018.
[10] C.-W. Shiu, Y.-C. Chen, and W. Hong, “Encrypted image-based reversible data hiding with public key cryptography from difference expansion,” Signal Process., Image Commun., vol. 39, pp. 226–233, Nov. 2015.
[11] Y. C. Chen, T. H. Hung, S. H. Hsieh, and C. W. Shiu, “A New Reversible Data Hiding in Encrypted Image Based on Multi-Secret and Lightweight Cryptographic Algorithms,” IEEE Trans. Inf. Forensics Security, vol. 14, no. 12, pp. 3332–3343, Dec. 2019.
[12] X. Cao, L. Du, X. Wei, D. Meng, and X. Guo, “High capacity reversible data hiding in encrypted images by patch-level sparse representation,” IEEE Trans. Cybern., vol. 46, no. 5, pp. 1132–1143, May 2016.
[13] P. Puteaux and W. Puech, “An efficient MSB prediction-based method for high-capacity reversible data hiding in encrypted images,” IEEE Trans. Inf. Forensics Security, vol. 13, no. 7, pp. 1670–1681, Jul. 2018.
[14] Y. Puyang, Z. Yin, and Z. Qian, “Reversible data hiding in encrypted images with two-MSB prediction,” in Proc. IEEE Int. Workshop Inf. Forensics Secur., Hong Kong, Dec. 2018, pp. 1–7.
[15] A. Malik, H. X. Wang, Y. Chen, and A. N. Khan, “A reversible data hiding in encrypted image based on prediction-error estimation and location map,” Multimedia Tools Appl., vol. 79, pp. 1–24, Jan. 2020.
[16] S. Chen, C.-C. Chang, and Q. Liao, “Fidelity preserved data hiding in encrypted images based on homomorphism and matrix embedding,” IEEE Access, vol. 8, pp. 22345–22356, Jan. 2020.
[17] Z.-L. Liu and C.-M. Pun, “Reversible data-hiding in encrypted images by redundant space transfer,” Inf. Sci., vol. 433–434, pp. 188–203, Apr. 2018.
[18] M. Li, L. Wang, J. Fan, Y. Zhang, K. Zhou, and H. Fan, “Fidelity preserved data hiding in encrypted highly auto correlated data based on homomorphism and compressive sensing,” *IEEE Access*, vol. 7, pp. 69810–69825, May 2019.

[19] X. Zhang, “Separable reversible data hiding in encrypted image,” *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 826–832, Apr. 2012.

[20] S. Yi and Y. Zhou, “Separable and reversible data hiding in encrypted images using parametric binary tree labeling,” *IEEE Trans. Multimedia*, vol. 21, no. 1, pp. 51–64, Jan. 2019.

[21] Y. Wu, Y. Xiang, Y. Guo, J. Tang, and Z. Yin, “An improved reversible data hiding in encrypted images using parametric binary tree labeling,” *IEEE Trans. Multimedia*, vol. 22, no. 8, pp. 1929–1938, Aug. 2020, doi: 10.1109/TMM.2019.2952979.

[22] Z. Qian, X. Zhang, Y. Ren, and G. Feng, “Block cipher based separable reversible data hiding in encrypted images,” *Multimedia Tools Appl.*, vol. 75, no. 21, pp. 13749–13763, Nov. 2016.

[23] C. Qin, Z. He, X. Luo, and J. Dong, “Reversible data hiding in encrypted image with separable capability and high embedding capacity,” *Inf. Sci.*, vol. 465, pp. 285–304, Oct. 2018.

[24] H. Ge, Y. Chen, Z. Qian, and J. Wang, “A high capacity multi-level approach for reversible data hiding in encrypted images,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 8, pp. 2285–2295, Aug. 2019.

[25] F. Di, F. Huang, M. Zhang, J. Liu, and X. Yang, “Reversible data hiding in encrypted images with high capacity by bitplane operations and adaptive embedding,” *Multimedia Tools Appl.*, vol. 77, no. 16, pp. 20971–20993, Aug. 2018.

[26] D. Huang and J. Wang, “High-capacity reversible data hiding in encrypted image based on specific encryption process,” *Signal Process., Image Commun.*, vol. 80, Feb. 2020, Art. no. 115632.

[27] X. Zhang, J. Long, Z. Wang, and H. Cheng, “Lossless and reversible data hiding in encrypted images with public-key cryptography,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 9, pp. 1622–1631, Sep. 2016.

[28] S. Zheng, Y. Wang, and D. Hu, “Lossless data hiding based on homomorphic cryptosystem,” *IEEE Trans. Dependable Secure Comput.*, early access, Apr. 26, 2019, doi: 10.1109/TDSC.2019.2913422.

[29] Y. Qiu, Z. Qian, H. Zeng, X. Lin, and X. Zhang, “Reversible data hiding in encrypted images using adaptive reversible integer transformation,” *Signal Process.*, vol. 167, Feb. 2020, Art. no. 107288.

[30] C.-C. Chang, I.-C. Chuang, and C.-Y. Chung, “Quadtree-segmented image compression method using vector quantization and cubic B-spline interpolation,” *Imag. Sci. J.*, vol. 52, no. 2, pp. 106–116, Jun. 2004.

[31] F. Di, M. Zhang, X. Liao, and J. Liu, “High-fidelity reversible data hiding by quadtree-based pixel value ordering,” *Multimedia Tools Appl.*, vol. 78, no. 6, pp. 7125–7141, Aug. 2018.

[32] N. Nasrullah, J. Sang, M. Mateen, M. A. Akbar, H. Xiang, and X. Xia, “Reversible data hiding in compressed and encrypted images by using kd-tree,” *Multimedia Tools Appl.*, vol. 78, no. 13, pp. 17535–17554, Jan. 2019.

[33] P. Bas and T. Furon. *Image Database of BOWS-2*. Accessed: Oct. 28, 2019. [Online]. Available: http://bows2.ec-lille.fr/

---

**SISHENG CHEN** received the B.S. degree from Shandong University, in 2005, and the M.S. degree in applied mathematics from Fujian Normal University, in 2008. He is currently pursuing the Ph.D. degree in information engineering and computer science with Feng Chia University, Taichung, Taiwan. He is also a Lecturer with the Fuqing Branch of Fujian Normal University. His research interests include multimedia security and wireless network security.

---

**CHIN-CHEN CHANG** (Fellow, IEEE) received the B.Sc. degree in applied mathematics and the M.Sc. degree in computer and decision sciences from National Tsing Hua University, and the Ph.D. degree in computer engineering from National Chiao Tung University. He was with National Chung Cheng University, from 1989 to 2005. He has been a Chair Professor with the Department of Information Engineering and Computer Science, Feng Chia University, since February 2005. His current research interests include database design, computer cryptography, image compression, and data structures.