IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 28, 2019, accepted December 12, 2019, date of publication December 25, 2019,
date of current version February 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2962230

Two-Layer Reversible Data Hiding Based on AMBTC Image With (7, 4) Hamming Code

JUAN LIN[1,2], SHAOWEI WENG[3,4], (Member, IEEE), TIANCONG ZHANG[3], BO OU[5], AND CHIN-CHEN CHANG[6,7], (Fellow, IEEE)

[1]Engineering Research Center for ICH Digitalization and Multi-Source Information Fusion, (Fuqing Branch of Fujian Normal University), Fujian Province University, Fuzhou 350300, China
[2]School of Electronic and Information Engineering, Fuqing Branch of Fujian Normal University, Fuzhou 350300, China
[3]School of Information Science and Engineering, Fujian University of Technology, Fuzhou 350118, China
[4]Guangdong Key Laboratory of Intelligent Information Processing and Shenzhen Key Laboratory of Media Security, Shenzhen 518060, China
[5]College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China
[6]Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan
[7]School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China

Corresponding authors: Shaowei Weng (wswweiwei@126.com) Tiancong Zhang (kushentian@163.com)

This work was supported in part by the National NSF of China under Grant 61872095, Grant 61571139, Grant 61872128, in part International Scientific and Technological Cooperation of Guangdong Province under Grant 2019A050513012, in part by the Open Project Program of Shenzhen Key Laboratory of Media Security under Grant ML-2018-03, in part by the Natural Science Foundation of Fujian Province under Grant 2018J01788, and in part by the Education and Scientific Research Foundation of Fujian Province under Grant JA15575.

ABSTRACT In Malik *et al.*'s method, each pixel can be embedded with $log_2 3$ bits by being modified at most 1. Thus, their method achieves significant hiding capacity while maintaining good visual quality. However, in their method, the first lower and the first upper quantization levels must be excluded from data embedding in order to ensure reversibility. To this end, we propose a two-layer reversible data hiding (RDH) scheme in combination with (7,4) Hamming code. In the 1st-layer embedding, each block can be embedded with 16 bits. In the 2nd-layer embedding, each already-modified block can carry 6 bits or 12 bits by taking advantage of (7,4) Hamming code that hides three bits by modifying only one bit. At most 2-bit additional information is needed to help decoders to correctly extract the original lower and upper quantization levels. By means of two-layer embedding, our method achieves higher embedding capacity while maintaining almost the same visual quality, compared with Malik *et al.*'s method. Experimental results also demonstrate our effectivity.

INDEX TERMS Reversible data hiding, AMBTC (absolute moment block truncation coding), (7, 4) hamming code, two-layer.

I. INTRODUCTION

Data hiding is a technique capable of embedding hidden information into a cover media imperceptibly for secret communication [1]. The digital images are frequently utilized for the cover objects because they are easily processed and provide rich redundancies for data embedding. The images used for data embedding are named as cover images. In contrast, the embedded images are named as stego images. Data hiding can be divided into reversible data hiding (RDH) and irreversible data hiding, depending on whether the cover image can be fully recovered or not after the hidden data is extracted [2]. In addition to reversibility, the other two important performance measures are stego image quality and hiding capacity for an RDH method.

Data hiding methods can be applied to images in various domains, like spatial domain, frequency domain, or compression domain. The spatial domain data hiding schemes can hide data into images by simply modifying the pixel values. The well-known least-significant-bit (LSB) substitution method is one of the most representative spatial domain methods [3]. The frequency domain data hiding schemes embed data into frequency coefficients obtained by performing discrete wavelet transform (DWT) [4], or discrete cosine transform (DCT) [5] on images. In the compression domain data hiding schemes, secret data are usually embedded into the compressed code of a cover image, which is created by a lossy compression technique such as vector quantization

The associate editor coordinating the review of this manuscript and approving it for publication was Yongjie Li.

(VQ) [6], side match vector quantization (SMVQ) [7], joint photographic experts group (JPEG) [8] and block truncation coding (BTC) [9]. Compared with VQ, SMVQ and JPEG, BTC is an effective and simple compression technique. Absolute moment block truncation coding (AMBTC) [10] is an extension of BTC, in which, an image block is compressed into its compressed code, i.e., a trio consisting two quantization levels and a bitmap.

Considering that AMBTC is a lossy compression technique, reversibility of the AMBTC-based methods means that a stego image/code is restored to its AMBTC-compressed state rather than its original state after the hidden data are extracted. Generally speaking, the AMBTC-based RDH methods are classified into the following four categories [11], [12]: namely 1) bitmap replacement [13]–[17], 2) histogram shifting (HS) [18]–[22], 3) prediction error expansion (PEE) [23]–[29] and 4) compressed image based data hiding [30]–[32]. The bitmap replacement was proposed firstly by Huang *et al.* [15], in which each block is classified into two classes according to the difference between two quantization levels: smooth and complex, and then, the bitmap of one smooth block is replaced by secret data. Afterwards, many researchers extend bitmap replacement to design various RDH algorithms so that the hiding capacity is increased as much as possible on the basis of maintaining the image quality. HS was introduced by Lo *et al.* into an AMBTC-compressed image [33]. Usually, their method can maintain good image quality but achieve limited hiding capacity. To further improve hiding capacity, the prediction is applied to two quantization levels, and thus, the prediction-errors are expanded to embed data. These three kinds of methods are carried out on the compressed code of an AMBTC image, and therefore, they can maintain good image quality but cannot provide high hiding capacity.

In 2015, Lin *et al.* proposed an RDH scheme based on AMBTC-compressed images. For one embeddable block of an AMBTC-compressed image, each pixel valued the upper quantization level is increased by 1 or kept unaltered to embed 1-bit data. In contrast, each pixel equal to the lower quantization level is decreased by 1 or kept unchanged to embed 1-bit data [30]. Thus, their method can achieve the hiding capacity of almost 1 bit per pixel (bpp). Afterwards, Pan *et al.* proposed an RDH method that embeds data into two quantization levels of each AMBTC compressed image based on a reference matrix [31]. Compared with Lin *et al.*'s method, it does not increase the hiding capacity but improves the image quality. In 2018, Malik *et al.* proposed an AMBTC-based RDH scheme using pixel value adjusting strategy [32]. In their scheme, for a block, when the difference between two quantization levels is large, except the first pixel valued the upper quantization and the first pixel valued the lower quantization, each of the remaining pixels is defined as an embeddable pixel. During data embedding, each embeddable pixel can be changed into three different values: plus or minus 1, or remaining unaltered, and thus it can be embedded with $log_2 3$ bits. Their scheme can achieve significant visual quality by modifying pixels at most by 1. Additionally, their method is also able to obtain high hiding capacity of approximately 1.5 bpp by hiding $log_2 3$-bit secret data into every embeddable pixel.

However, for ensuring reversibility, Malik *et al.*'s method needs to treat the first lower and upper quantization levels in each block as reference pixels. These reference pixels cannot be modified during data embedding. In this paper, in order to overcome this drawback and further increase the embedding performance, a two-layer RDH scheme in combination with (7,4) Hamming code [34] is proposed. In the first-layer embedding, since all the 16 pixels of a 4×4-sized block are involved in data embedding, this block can carry 16 bits. It is well known that the (7,4) Hamming code has the advantage that hides 3 bits by modifying only one bit. We make full use of this advantage in the 2nd-layer data embedding, so that one already-modified block in the 1st-layer data embedding still can embed 6 bits or 12 bits. After data embedding, we need to extract two quantization levels by the way of at most 2-bit additional information (L_M). In a word, by means of two-layer embedding, our method increases largely the hiding capacity while maintaining comparable visual quality when compared with Malik *et al.*'s method.

The rest of this paper is organized as follows. We will give a brief introduction to the related works in Section II. Section III presents the proposed scheme, followed by the experimental results in Section IV. Finally, we give conclusions in Section V.

II. RELATED WORKS

A. THE AMBTC COMPRESSION TECHNIQUE

In 1984, the AMBTC compression technique was proposed by Lema and Mitchell, which focuses on preserving the local characteristics of spatial image blocks [10]. Specifically, an original image I is split into non-overlapping $n \times n$-sized blocks $\{I_i\}_{i=1}^{N}$, where N is the total number of blocks. For each block I_i, the mean value $\mu_i = \frac{1}{n \times n} \sum_{j=1}^{n \times n} I_{i,j}$ and the standard deviation $\sigma_i = \frac{1}{n \times n} \sum_{j=1}^{n \times n} |I_{i,j} - \mu_i|$ are calculated, where $I_{i,j}$ indicates the j-th pixel of I_i, the notation $|\cdot|$ represents the absolute operation. The lower mean value L_i is calculated by averaging the pixels of I_i smaller than μ_i, while the higher mean value H_i is computed by averaging the pixels of I_i larger than or equal to μ_i. Subsequently, $B_{i,j}$ is marked by 1 if $I_{i,j} < \mu_i$; otherwise, $B_{i,j}$ is marked by 0, where $B_{i,j}$ denotes the j-th bit of the bitmap B_i. Therefore, a trio (H_i, L_i, B_i) is used to represent the AMBTC compressed code of block $I_{i,j}$. The reconstruction of AMBTC codes is simple. We use the notation R_i to denote the reconstructed block from the trio (H_i, L_i, B_i). If $B_{i,j} = 1$, then $R_{i,j} = H_i$; otherwise, $R_{i,j} = L_i$, where $R_{i,j}$ is the j-th pixel of R_i.

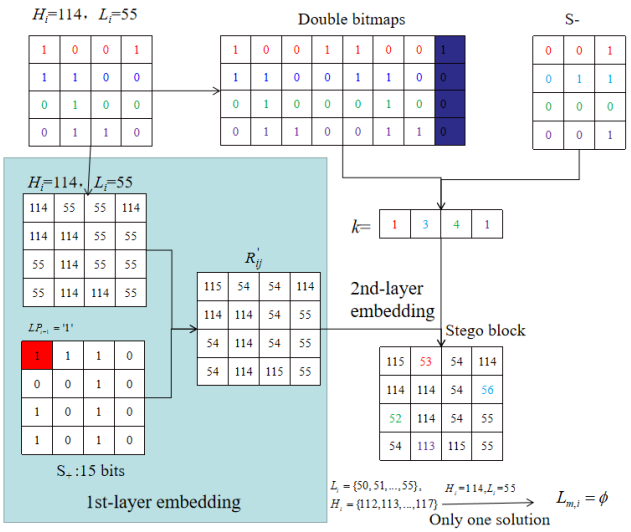


FIGURE 1. A simple example of the data embedding phase.

An example is given to illustrate the compression and reconstruction processes of the AMBTC compression technique. Since $n \times n$ is set 4×4 in our proposed RDH scheme, the size of block I_i used in Fig. 1 is also 4×4. Suppose that $I_{i,j}$ =[162, 157, 163, 161; 162, 157, 163, 161; 162, 157, 163, 161; 162, 157, 163, 161]. The mean value μ_i is calculated as 160.75, and therefore, $L_i = 157$, $H_i = 162$, B_i = [1011; 1011; 1011; 1011]. The reconstructed image block R_i =[162, 157, 162, 162; 162, 157, 162, 162; 162, 157, 162, 162; 162, 157, 162, 162] is generated by replacing the '0' and '1' in B_i with $L_i = 157$ and $H_i = 162$, respectively.

B. (7, 4) HAMMING CODE

Since the (7, 4) Hamming code [34] was first proposed by Richard Hamming in 1950 as a linear error correction code, it has been widely used in data hiding as an efficient steganography method to achieve satisfactory image visual quality. The advantage of the (7, 4) Hamming code is that it can detect and correct 1 error bit for one code composed of 4 original bits and 3 parity check bits with the help of the parity check matrix.

Specifically, the four original bits d_1, d_2, d_3, d_4 are used to yield three parity check bits p_1, p_2, p_3, by multiplying with the code generator matrix G of the (7, 4) Hamming code. Therefore, one code C with size of 7 is formed by combining 4 original bits with 3 parity bits. The detailed procedure can be represented by

$$\begin{aligned} C &= d \times G \\ &= (d_1, d_2, d_3, d_4) \times \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \\ &= (p_1, p_2, d_1, p_3, d_2, d_3, d_4)\,. \end{aligned} \tag{1}$$

And, three parity check bits p_1, p_2, p_3 can be obtained by the following Eq. (2).

$$\begin{aligned} p_1 &= d_1 \oplus d_2 \oplus d_4, \\ p_2 &= d_1 \oplus d_3 \oplus d_4, \\ p_3 &= d_2 \oplus d_3 \oplus d_4, \end{aligned} \tag{2}$$

where $\oplus$ is the exclusive-or operation.

If the original four bits $(d_1, d_2, d_3, d_4) = (0101)_2$, then three parity check bits are $(p_1, p_2, p_3) = (010)_2$. Therefore, $C = (0100101)_2$.

On the decoding side, the receiver can use the same parity check matrix H as in data embedding to detect and correct whether the message C has been tampered or not. Assuming the received message is R, Eq. (3) is utilized to judge whether the R is tampered or not by the value of z.

$$z = H \times R^T, \tag{3}$$

where z is called the syndrome vector. Specifically, $z = 0$ indicates the R is not tampered, i.e., $R = C$. Otherwise, R is tampered. Taking $C = (0100101)_2$ for example, if the fifth bit of C is flipped, then R=$(0100001)_2$. We can calculate $z = (101)_2 = 5$ using Eq. (3). $z \neq 0$ implies that one error bit occurs in the fifth bit of R, and therefore, the original data can be recovered by flipping the fifth bit of R. Finally, $C = (0100001)_2$.

C. MATRIX CODING [35]

The main idea of matrix coding is described as follows. To begin with, we use a pseudo-random number generator to generate a decimal array S which is used to represent the secret data, i.e., $S = \{s_j | j = 1, 2, \cdots, n\}$, where s_j is used to denote the j-th element of the secret data S, and $s_j \in \{0, 1, \cdots, 7\}$. For the simplicity of description, we use s to replace s_j by ignoring the subscript j of s_j in the rest of this paper. According to the criterion of (7, 4) Hamming code mentioned in the Section II-B, we collect seven LSBs of n original pixels to form a 7-bit binary number x and embed s into x to generate y by keeping x unaltered or only flipping one bit of x, where x and y are used to denote the original and marked 7-bit binary numbers, respectively, and $s \in \{0, 1, \cdots, 7\}$. According to the description above, the advantage of the matrix coding lies in the fact that it can embed 3 bits in x by at most changing one bit of x. After embedding, all the bits of y are appended to the LSBs of n original pixels to form the corresponding stego pixels. On the decoding side, seven LSBs of n stego pixels are extracted to construct y, and then s is extracted via the following equation:

$$s = \text{conv}\Big(\text{mod}\left(H \times y^T, 2\right)\Big)_{10}, \tag{4}$$

where the superscript T represents the transpose operation, mod $(\cdot, 2)$ is the modulo 2 operation which is used to obtain 3-bit binary secret data, the notation conv $(\cdot)$ is a function used to convert numbers from binary representation to decimal representation.

If $n = 7$, then the matrix coding can achieve satisfactory embedding performance because only one LSB of seven pixels are modified to embed 3 bits. However, $n \neq 7$ may lead to large modifications for some original pixel. Taking $n = 3$ for example, suppose that x is generated by extracting the 2 LSBs (i.e., the 2nd and 1st bits) of the first pixel, two LSBs (i.e., the 2nd and 1st bits) of the second pixel and three bits (i.e., the 3rd, 2nd and 1st bits) of the third pixel. If the 3rd bit of the third pixel is flipped during data embedding, the embedding distortion for the third pixel is unacceptable.

III. THE PROPOSED SCHEME

In this paper, we propose a two-layer RDH method based on (7,4) Hamming code for AMBTC compressed images.

It is mainly composed of two parts: data embedding phase, data extraction and image recovery phase. In the data embedding phase, the embedding method has two layers. The first layer is based on Lin *et al.*'s method. The second layer is based on matrix embedding with (7,4) Hamming code. In order to restore the original H_i and L_i, we need a location map to record some multi-solution cases. During the extract phase, we use exclusion methods to restore the original H_i and L_i and extract confidential messages. The embedding method, 6 types of embedded block and an example of embedding phase are described in Section III-A, the extract method and its example are described in Section III-B.

A. DATA EMBEDDING PHASE

The data embedding phase is composed of two-layer data embedding. An original grayscale image I of size $W_I \times H_I$ is separated into non-overlapping blocks $\{I\}_{i=1}^{N}$ of size 4×4, and the AMBTC code of block I_i is denoted as a trio (L_i, H_i, B_i), where W_I and H_I are used to indicate the width and height of I, respectively, and N is the total number of blocks. Let R_i be the AMBTC-compressed block reconstructed by (L_i, H_i, B_i). Here, the 1st-layer and 2nd-layer embedding will be separately introduced in the following two subsections.

1) THE 1ST-LAYER DATA EMBEDDING

Input: AMBTC-compressed image block R_i, secret data S_+ used in the 1st-layer embedding.
Output: Stego image R'_i.

If $s \in S_+$ and $s == 0$
    $R'_{i,j} = R_{i,j}$;
elseif $s \in S_+$ and $s == 1$
    if $R_{i,j} == H_i$
        $R'_{i,j} = H_i + 1$;
    elseif $R_{i,j} == L_i$
        $R'_{i,j} = L_i - 1$;
    end
end

where $R_{i,j}$ and $R'_{i,j}$ denote the j-th original and stego pixels of block R_i after the 1st-layer embedding, respectively.

2) THE 2ND-LAYER DATA EMBEDDING

The 2nd-layer data embedding based on matrix embedding (7, 4) Hamming code is performed after the 1st-layer data embedding. The matrix embedding (7, 4) makes an AMBTC-compressed block capable of carrying 12 bits or 6 bits according to the local complexity. The key idea of the matrix embedding (7,4) is described below: 3-bit secret data s to be embedded from S_- is embedded into carrier x by using Eq. (5), where $s \in \{0, 1\}$ and x is a 7-bit binary number, while are obtained from the bitmap B_i.

$$\begin{aligned} z &= s \oplus H \times x, \\ y &= Emd(x, s) = x \oplus F(z), \end{aligned} \tag{5}$$

where $F(z)$ is the coset leader of which syndrome is z, $Emd(\cdot)$ represents the embedding process. $F(z) = e_k$, where e_k indicates the k^{th} unit vector of size 7 with the k^{th} position being 1 and $k \in \{0, 1, \cdots, 7\}$. For example, e_3 is the third unit vector whose third position is 1, i.e., $e_3 = [0\,0\,1\,0\,0\,0\,0]$. In contrast, $e_0 = [0\,0\,0\,0\,0\,0\,0]$ is a supplementary definition.

After a preliminary understanding of the matrix embedding (7, 4), the 2nd-layer embedding is classified into three cases according to $H_i - L_i$: $H_i - L_i > 2$ and $H_i - L_i \leq T_h$, $H_i - L_i > T_h$, $H_i - L_i \leq 2$.

Case 1: If $H_i - L_i > 2$ and $H_i - L_i \leq T_h$, the total six bits can be hidden into B_i, where T_h is a predefined threshold, and usually, $T_h \geq 4$ is set. Specifically, the first three bits can be hidden in the first eight pixels of B_i. The last three secret bits can be hidden in the last eight pixels of B_i. To begin with, we select the first seven bits of B_i to form x. Afterwards, z is computed by Eq. (5). Since $F(z)$ has 8 cases, each case corresponds to sequentially one pixel located in one of eight positions: $(i, 1), \cdots, (i, 4), (i, 5), \cdots, (i, 8)$. When $F(z) = e_k$, then the corresponding pixel $R_{i,k+1}$ is modified as below:

$$R''_{i,k+1} = \begin{cases} R'_{i,k+1} + 1 & \text{if } R'_{i,k+1} = H_i + 1, \\ R'_{i,k+1} - 1 & \text{if } R'_{i,k+1} = H_i, \\ R'_{i,k+1} + 1 & \text{if } R'_{i,k+1} = L_i, \\ R'_{i,k+1} - 1 & \text{if } R'_{i,k+1} = L_i - 1, \end{cases} \tag{6}$$

where $k = 0, 1, \cdots, 7$ and $R''_{i,j}$ denote the j-th pixel of block R_i after the 2nd-layer embedding.

The last eight pixels of R_i can be encoded using the similar manner, but the detailed steps for this are ignored.

Case 2: If $H_i - L_i > T_h$, three bits are embedded into four pixels in each row, and therefore, the total 12 bits can be hidden into a block. Eight bits are obtained by extracting four bits of each row of B_i twice, and then, the first seven bits are used to construct x. By means of x and s, z is calculated via Eq. (5), and further, $F(z)$ is generated. If $F(z) = e_k$ and k is set one of 0, 1, 2, 3, the corresponding pixel $R_{i,k+1}$ must be increased or decreased by 1 according to Eq. (6). If $F(z) = e_k$ and k is set one of 4, 5, 6, 7, the corresponding pixel $R_{i,k-3}$

TABLE 1. Six classes of blocks.

Six classes	L_M	Conditions	Examples
unused blocks	$\varnothing$	$(H_i = L_i$ or $H_i = 255$ or $L_i = 0)$	$R_i'' = \begin{bmatrix} 128 & 128 & 128 & 128 \\ 128 & 128 & 128 & 128 \\ 128 & 128 & 128 & 128 \\ 128 & 128 & 128 & 128 \end{bmatrix}$ $H_i^1 = L_i^1 = 218$
Single layer embedding	$\varnothing$	$(0 < H_i - L_i \leq 2$ and $H_i \leq 253$ and $L_i \geq 2$ and $N_{v,i} = 4)$ or $((H_i = 254$ and $H_i \neq L_i)$ and $N_{v,i} = 4)$ or $((L_i = 1$ and $H_i \neq L_i)$ and $N_{v,i} = 4)$	$R_i'' = \begin{bmatrix} 163 & 163 & 162 & 160 \\ 163 & 163 & 162 & 160 \\ 162 & 163 & 163 & 161 \\ 163 & 163 & 162 & 160 \end{bmatrix}$ $(H_i^1 = 162, L_i^1 = 161')$
	1 bit	$(0 < H_i - L_i \leq 2$ and $H_i \leq 253$ and $L_i \geq 2$ and $N_{v,i} = 3)$ or $((H_i = 254$ and $H_i \neq L_i)$ and $N_{v,i} = 3)$ $((L_i = 1$ and $H_i \neq L_i)$ and $N_{v,i} = 3)$	$R_i'' = \begin{bmatrix} 136 & 134 & 133 & 134 \\ 136 & 134 & 134 & 134 \\ 136 & 133 & 133 & 134 \\ 136 & 134 & 134 & 133 \end{bmatrix}$ $(H_i^1 = 135, L_i^1 = 134)$ $(H_i^2 = 136, L_i^2 = 134)$
Two-layer embedding	$\varnothing$	$H_i - L_i > 2$ and $(H_i \leq 252$ and $L_i \geq 3)$ and only one solution	$R_i'' = \begin{bmatrix} 115 & 53 & 54 & 114 \\ 114 & 114 & 54 & 56 \\ 52 & 114 & 54 & 55 \\ 54 & 113 & 115 & 55 \end{bmatrix}$ $H_i^1 = 114, L_i^1 = 55$
	1 bit	$H_i - L_i > 2$ and $(H_i \leq 252$ and $L_i \geq 3)$ and two solutions	$R_i'' = \begin{bmatrix} 41 & 42 & 42 & 78 \\ 42 & 42 & 41 & 78 \\ 41 & 41 & 42 & 78 \\ 41 & 41 & 42 & 74 \end{bmatrix}$ $(H_i^1 = 75, L_i^1 = 42)$ $(H_i^2 = 76, L_i^2 = 42)$
	2 bits	$H_i - L_i > 2$ and $(H_i \leq 252$ and $L_i \geq 3)$ and two more solutions	$R_i'' = \begin{bmatrix} 42 & 43 & 38 & 42 \\ 43 & 43 & 36 & 38 \\ 43 & 42 & 38 & 42 \\ 43 & 38 & 36 & 42 \end{bmatrix}$ $(H_i^1 = 42, L_i^1 = 37)$ $(H_i^2 = 42, L_i^2 = 36)$ $(H_i^3 = 42, L_i^3 = 38)$

must be increased or decreased by 2 as follows:

$$R''_{i,k-3} = \begin{cases} R'_{i,k-3} + 2 & \text{if } R'_{i,k-3} = H_i + 1, \\ R'_{i,k-3} - 2 & \text{if } R'_{i,k-3} = H_i, \\ R'_{i,k-3} + 2 & \text{if } R'_{i,k-3} = L_i, \\ R'_{i,k-3} - 2 & \text{if } R'_{i,k-3} = L_i - 1, \end{cases} \tag{7}$$

where $k = 4, 5, \cdots, 7$.

Case 3: For a block satisfying $H_i - L_i \leq 2$, for reversibility, it must be excluded from the 2nd-layer data embedding.

3) THE EMBEDDING PROCESS OF TWO-LAYER DATA EMBEDDING

For recovering the AMBTC compressed image correctly, we need to determine H_i and L_i before data extraction and image recovery by judging whether H_i and L_i satisfy one of the following conditions or not: $H_i - L_i > T_h$ or $H_i - L_i \geq 2$ and $H_i - L_i \leq T_h$. However, for some blocks, there is more than one solution satisfying the above conditions. Before considering the number of solutions, all blocks are classified into three classes according to whether they are used for data embedding or not: unused blocks due to $H_i = L_i$, blocks only used for single layer embedding, and blocks available for two-layer embedding. After considering the number of solutions, all the blocks must be classified into six classes, as shown in Table 1, where $N_{v,i}$ is used to denote the number of pixel values with different values in R_i. The detailed embedding process for each class is described as follows:

Input: Original gray-scale image I, secret data S_+ and S_-, predefined threshold T_h.
Output: AMBTC-compressed stego image R''.

Step 1: The original image I is partitioned into 4×4-sized non-overlapped blocks $\{I_i\}_{i=1}^N$, and then, each block I_i is compressed using AMBTC to form

its compressed trio, i.e., (H_i, L_i, B_i), where $i = \{1, 2, \cdots, N\}$, and N is the total number of blocks. R_i is used for representing the reconstructed block from the trio (H_i, L_i, B_i).

Step 2: Scan $\{R_i\}_{i=1}^{N}$ according to the raster scanning order. The following three steps are performed to classify the scanned block into three classes:

(2.a) Referring to Table 1, if R_i satisfies one the following three conditions: $H_i = L_i$, $H_i = 255$, and $L_i = 0$, it is deemed as an unused block, that is, it is not used for data embedding. Note that $H_i = L_i$ implies $N_{v,i} = 0$. Since an unused block can be easily identified by judging whether it satisfies the condition $H_i = L_i$ or not, $L_{M,i}$ is not required to record this block.

(2.b) If $0 < H_i - L_i \leq 2$, and $H_i \leq 254$, $L_i \geq 1$, the LM_{i-1} of the previously-processed block R_{i-1} is checked. If $LM_{i-1} = \varnothing$, then 16 bits are all extracted from S_+. Otherwise, suppose the length of LM_{i-1} is l_n, and then, 16 bits are formed by concatenating LM_{i-1} and $16 - l_n$ bits from secret data S_+. Finally 16 bits are embedded into R_i following the Eq.(8):

$$R'_{i,j} = \begin{cases} R_{i,j} + 1, & \text{if } s = 1,\ R_{i,j} = H_i, \\ R_{i,j} - 1, & \text{if } s = 1,\ R_{i,j} = L_i, \\ R_{i,j}, & \text{if } s = 0,\ R_{i,j} = H_i, \\ R_{i,j}, & \text{if } s = 0,\ R_{i,j} = L_i, \end{cases} \tag{8}$$

where $s \in \{0, 1\}$.

(2.b.1) After the 1st-layer embedding, if $N_{v,i}$ is 3 corresponding to the fourth line of Table 1, there exists two combinations of (H_i, L_i) satisfying $0 < H_i - L_i \leq 2$, and $H_i \leq 254$, $L_i \geq 1$. To this end, we need 1 bit $L_{M,i}$ to record these two solutions. Note that $L_{M,i}$ must be embedded into the next block R_{i+1}.

(2.b.2) After the 1st-layer embedding, if $N_{v,i}$ is 4 corresponding to the third line of Table 1, we can obtain the original H_i and L_i during data extraction without need of any additional information.

Step 3: If $H_i - L_i > 2$, $H_i \leq 253$ and $L_i \geq 2$, the 1st-layer embedding is performed according to the same way as Step (2.b). The 2nd-layer embedding is described in detail below:

(3.a) If $H_i - L_i \leq T_h$, the total six bits can be hidden into R_i, and usually, $T_h \geq 4$ is set.

Referring to Section III-A.2, x is formed by extracting the first seven bits of B_i according to the raster scan order. After obtaining x, $F(z) = e_k$ is generated via Eq. (5), where $k \in \{0, 1, 2, \cdots, 7\}$. Correspondingly, $R_{i,k}$ is modified to embed 3-bit data from S_- according to Eq. (6). Similarly, the last eight pixels of R_i can also be embedded with 3-bit data, but the detailed embedding process for this is omitted.

After data embedding, if the number of pixels unequal to H_i, L_i, $H_i + 1$, $L_i - 1$ is 2 (referring to the sixth line of Table (1), then go to Step (3.c).

(3.b) If $H_i - L_i > T_h$, the total 12 bits can be hidden into R_i. Specifically, x is obtained by extracting each row of B_i twice. Since $F(z)$ has 8 cases while each row only contains four pixels, each pixel corresponds to two cases. That is to say, $R_{i,k+1}$ corresponds to e_k and e_{k+4}, where k belongs to $\{0, 1, 2, 3\}$. If $R_{i,k+1}$ corresponds to e_k, then the detailed modification to $R_{i,k+1}$ refers to Eq. (6). Otherwise, $R_{i,k+1}$ is modified via Eq. (7). The similar manner is also applied to process the remaining three rows.

After the 2nd-layer data embedding, if the number of pixels unequal to H_i, L_i, $H_i + 1$, $L_i - 1$ is 4 (referring to the seventh line of Table (1), then go to Step (3.c).

(3.c) After the 2nd-layer data embedding, we know that $min\{R''_{i,j}\}_{j=1}^{16} \in \{L_i - 3, L_i - 2, L_i - 1, L_i, L_i + 1, L_i + 2\}$ and $max\{R''_{i,j}\}_{j=1}^{16} \in \{H_i - 2, H_i - 1, H_i, H_i + 1, H_i + 2, H_i + 3\}$. Therefore, all the combinations of (H_i, L_i) are generated depending on $L_i \in S_L = \{min\{R'_{i,j}\}_{j=1}^{16} + m_1\}$ $(m_1 \in \{-2, -1, \cdots, 3\})$ and $H_i \in S_H = \{max\{R'_{i,j}\}_{j=1}^{16} + m_2\}$ $(m_2 \in \{-3, -2, \cdots, 2\})$.

All candidate quantization pairs (H_i, L_i) satisfying $H_i - L_i > 2$, $H_i - L_i \leq T_h$, and $H_i \leq 253$, $L_i \geq 2$, are selected from S_L and S_H. For a candidate pair (H_i, L_i), when the number of pixels unequal to H_i, L_i, $H_i + 1$, $L_i - 1$ is 2 and these two pixels are not located in the same row, this pair (H_i, L_i) is defined as one solution satisfying $H_i - L_i > 2$, $H_i - L_i \leq T_h$, and $H_i \leq 253$, $L_i \geq 2$.

All candidate quantization pairs (H_i, L_i) satisfying $H_i - L_i > T_h$, $H_i \leq 252$ and $L_i \geq 3$, are selected from S_L and S_H. For a candidate pair (H_i, L_i), when the number of pixels unequal to H_i, L_i, $H_i + 1$, $L_i - 1$ is 4 and these four pixels are not located in the same row, this pair (H_i, L_i) is treated as one solution satisfying $H_i - L_i > T_h$, $H_i \leq 252$ and $L_i \geq 3$.

If there is only a solution, $L_{M,i}$ is not required, and thus it is an empty set. If the number of solutions is 2, $L_{M,i} \in \{0, 1\}$ is used to identify these two solutions. If the number of solutions is larger than 2, two bits $L_{M,i} \in \{00, 01, 10, 11\}$ is used to determine these solutions. And $L_{M,i}$ that points out the correct solution is hidden into the next block.

Step 4: For one block, if its pixels are all not the same and $H_i - L_i > 1$, then it is termed "embaddable block". For the last embeddable block, it can only be used for the 1st-layer embedding and the first H_i, L_i can be used for data embedding, while the each of the remaining 14 pixels can embed 1-bit data according to Eq. (8). The purpose of doing so is to ensure that this block does not generate $L_{M,i}$. Repeat the above steps, until all the embeddable block are processed.

4) A SIMPLE EXAMPLE

The following is a simple example given in Fig. 1 to show the embedding process of a block R_i with $H_i - L_i > T_h$. Taking the block R_i with $H_i = 114$, $L_i = 55$, $B_i = [1\ 0\ 0\ 1; 1\ 1\ 0\ 0; 0\ 1\ 0\ 0; 0\ 1\ 1\ 0]$ for example, suppose that the to-be-embedded bitstream used for the 1st-layer embedding is $S_+ = 1110001010101010_2$ and let the to-be-embedded secret data used for the 2nd-layer embedding be $S_- = 001011000001_2$, where the subscript 2 represents the binary bitstream. By means of H_i, L_i and B_i, we know $R_i = [114\ 55\ 55\ 114; 114\ 114\ 55\ 55; 55\ 114\ 55\ 55; 55\ 114\ 114\ 55]$. After S_+ is embedded into R' in the 1st-layer embedding, $R'_i = [115\ 55\ 55\ 114; 114\ 114\ 54\ 55; 54\ 114\ 54\ 55; 54\ 114\ 115\ 55]$.

Before carrying out the 2nd-layer data embedding, each row of B_i is repeated twice to form a new matrix of size 4×8. The first seven bits of each row is taken as x. For example, for the first row of B_i, it is obviously observed that $x = (1001100)_2$. s is used to indicate the first 3 bits $(001)_2$ of S_-. z is computed below:

$$\begin{aligned} z &= s \oplus Hx \\ &= \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \oplus \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \end{aligned} \tag{9}$$

Once z is known, $F(z) = e_1$ can be gained. Since $H_i - L_i > T_h$ holds and $F(z) = e_1$, the second pixel $R'(i, 2)$ valued 54 is modified so as to embed 3-bit data s according to Eq. (6). That is, $R''(i, 2) = 53$. The similar manner is used for the remaining three rows of B. After each row of B_i is processed, the final stego block R'' is generated.

B. DATA EXTRACTION AND IMAGE RECOVERY PHASE

The stego image R''_i is split into 4×4-sized non-overlapped image blocks according to the same order as data embedding. The extraction procedure is performed according to the reverse order as data embedding (i.e., the order from the last block to the first block).

Input: Stego image R''.
Output: AMBTC-compressed image R, secret data S_+ and S_-.

Step 1: Check whether each block is an unused block or not according to the reverse order as in the embedding process. One unused block has one of the following three characteristics: $H_c = L_c$ or $H_c = 255$ or $H_c = 0$, where $c \in N, N-1, \cdots, 2, 1$. That is, it can be easily identified depending on the above three characteristics. Since it is kept unaltered during data embedding, $R_c = R''_c$.

Step 2: Find the firstly-used block, and determine the first H_c and the first L_c of this block. Since the first H_c and the first L_c are not changed during data embedding, one embedded bit is extracted from each of the remaining 14 pixels in R_c as follows:

$$s = \begin{cases} R''_{c,j} - H_c, & \text{if } R'_{c,j} \geq H_c, \\ L_c - R''_{c,j}, & \text{if } R'_{c,j} \leq L_c. \end{cases} \tag{10}$$

After 14 bits are extracted, the original pixels are retrieved as below:

$$R'_{c,j} = \begin{cases} R'_{c,j} - 1, & \text{if } s = 1, R'_{c,j} = H_c + 1, \\ R'_{c,j} + 1, & \text{if } s = 1, R'_{c,j} = L_c - 1, \\ R'_{c,j}, & \text{if } s = 0, R'_{c,j} = H_c, \\ R'_{c,j}, & \text{if } s = 0, R'_{c,j} = L_c. \end{cases} \tag{11}$$

The extracted 14 bits are composed of T_h (8 bits), $L_{m,c-1}$ and secret bits. If $n_{s,c-1}$ in the next block is 1, the size of $L_{m,c-1}$ is zero, where $n_{s,c-1}$ denotes the number of solutions in R_{c-1}. Otherwise, we can calculate the length of $L_{m,c-1} = \lceil log_2 n_{s,c-1} \rceil$.

Step 3: Calculate $N_{v,i}$ of R''_c, and the detailed block restoration is divided into two cases according to $N_{v,i}$:

Case 1: When $N_{v,c}$ is 2, there are four possible pairs, i.e., (H_c, L_c), $(H_c + 1, L_c)$, $(H_c, L_c - 1)$ and (H_c+1, L_c-1). Therefore, 2 bits $L_{M,c}$ extracted from the secret data of the previously-retrieved block are used to identify the correct solution, i.e., (H_c, L_c). Since this block is embedded only once, we can easily extract the embedded secret data from R''_c according to Eq.(10). After 16 bits are extracted, the original pixels are retrieved according to Eq.(11). It is worthy mentioning that besides the firstly-used block, for each remaining block, the extracted 16 bits are composed of $L_{m,c-1}$ and secret bits.

Case 2: When $N_{v,c}$ is greater than 2, $L_c \in \{min\{R'_{c,j}\}_{j=1}^{16} + m_1\}$ ($m_1\{-2, -1, \cdots, 3\}$ and $H_c \in \{max\{R'_{c,j}\}_{j=1}^{16} + m_2\}$ ($m_2\{-3, -2, \cdots, 2\}$). All candidate pairs (H_c, L_c) satisfying the conditions of $H_c > L_c$ are selected. it is divided into three subcases according to $N_{u,c}$, where $N_{u,c}$ denotes the number of pixels unequal to H_c, L_c, $H_c + 1$, and $L_c - 1$:

Case 2.1: If $N_{u,c} = 0$ and $N_{v,c} = 4$, the solution is unique, and we can directly determine this solution (H_c, L_c). Since the solution is unique, $L_{m,c}$ is not required; if $N_{u,c} = 0$ and $N_{v,c} = 3$, the number of solutions is 2, and thus 1 bit $L_{M,c}$ extracted from the secret data of the previously-retrieved block is required to identify the correct solution. After (H_c, L_c) is known, the detailed data extraction and image restoration is the same as that in Case 1 due to that this block is embedded only once.

Case 2.2: If $N_{u,c} = 2$ and these two pixels are not located in the same row. Then, 2-bit $L_{m,c}$ extracted from the previously-retrieved block is used to identify the correct solution. Since this block is embedded twice, we need to firstly extract the embedded 16 bits during the 1st-layer data embedding by means of (H_c, L_c).

$$s = \begin{cases} 1, & \text{if } R''_{c,j} = H_c + 1 \text{ or } R''_{c,j} = H_c + 2, \\ 1, & \text{if } R''_{c,j} = L_c - 2 \text{ or } R''_{c,j} = L_c - 1, \\ 0, & \text{if } R''_{c,j} = H_c \text{ or } R''_{c,j} = H_c - 1, \\ 0, & \text{if } R''_{c,j} = L_c \text{ or } R''_{c,j} = L_c + 1, \end{cases} \tag{12}$$

where $j = \{1, 2, \ldots, 16\}$. The extracted 16 bits from the 1st-layer embedding are composed of $L_{m,c-1}$ and secret bits.

During the 2nd-layer embedding, B_c is firstly computed as follows:

$$B_c = \begin{cases} 1, & \text{if } R''_{c,j} \geq H_c - 1, \\ 0, & \text{if } R''_{c,j} \leq L_c + 1. \end{cases} \tag{13}$$

Then, the first 7 bit of B_i are used to form $x = (B_{c,1}, B_{c,2}, \ldots, B_{c,7})$. For the first eight pixels of $R''_{c,j}$, assume k is the position of the changed pixels and $k \in \{1, 2, \cdots, 8\}$. We scan each of the first eight pixels and find the pixel whose value is equal to one of four values $\{H_c + 2, H_c - 1, L_c + 1, L_c - 2\}$. Thus, the location of this pixel is k.

Next, the 7-bit number y obtained by modifying x during data embedding (see Eq. (5) for details) is calculated as follow:

$$y = \begin{cases} x, & \text{if } k = 1, \\ (x_1, \cdots, 1 - x_{k-1}, \cdots, x_7), & \text{if } k > 1. \end{cases} \tag{14}$$

The first 3 bits s are obtained by the formula $s = H \times y^T$, where T denotes the transpose operation. The last 3 bits can be extracted from the last 8 bits of B_c with the same manner as that of the first 3 bits.

After all embedded data are extracted, the original pixels are retrieved as below:

$$R_{c,j} = \begin{cases} H_c, & \text{if } R''_{c,j} \geq H_c - 1, \\ L_c, & \text{if } R''_{c,j} \leq L_c + 1. \end{cases} \tag{15}$$

Case 2.3: If $N_{u,c} = 4$ and these four pixels are not in the same row. Then, 2-bit $L_{m,c}$ extracted from the previous block are used to identify the correct solution. Since this block is performed twice, we can extract the secret data applied in the 1st-layer embedding according to the following equation:

$$s = \begin{cases} 1, & \text{if } R''_{c,j} \geq H_c + 1, \\ 1, & \text{if } R''_{c,j} \leq L_c - 1, \\ 0, & \text{if } R''_{c,j} \geq H_c - 2, R''_{c,j} \leq H_c, \\ 0, & \text{if } R''_{c,j} \geq L_c, R''_{c,j} \leq L_c + 2. \end{cases} \tag{16}$$

Afterwards, during the 2nd-layer extraction, $B_{c,j}$ is computed as follows:

$$B_{c,j} = \begin{cases} 1, & \text{if } R''_{c,j} \geq H_c - 2, \\ 0, & \text{if } R''_{c,j} \leq L_c + 2. \end{cases} \tag{17}$$

Then, x is formed by extracting the first row of B_c twice, i.e., $x = (B_{c,1}, \ldots, B_{c,4}, B_{c,1}, \ldots, B_{c,4})$. Assume k is the position of one changed pixel after the 2nd-layer embedding, and then k is

$$k = \begin{cases} j_1, & \text{if } R''_{c,j_1} \in \{H_c + 2, H_c - 1, L_c + 1, L_c - 2\}, \\ j_1 + 4, & \text{if } R''_{c,j_1} \in \{H_c + 3, H_c - 2, L_c + 2, L_c - 3\}, \end{cases} \tag{18}$$

where $j_1 = \{1, 2, 3, 4\}$.

The y is computed as follow:

$$y = \begin{cases} x, & \text{if } k = 1, \\ (x_1, \ldots, 1 - x_{k-1}, \ldots, x_7), & \text{if } k > 1. \end{cases} \tag{19}$$

The first 3 bits is obtained by the formula $s = Hy^T$. The remaining 9 bits can be obtained by the remaining three rows of B_i with the same manner. After all the data are extracted, the original pixels are retrieved as below:

$$R_{c,j} = \begin{cases} H_c, & \text{if } R''_{c,j} \geq H_c - 2, \\ L_c, & \text{if } R''_{c,j} \leq L_c + 2. \end{cases} \tag{20}$$

Step 4: Repeat Steps 2 to 3, until all blocks are processed. The secret bits extracted from each block used for data embedding during the 1st-layer embedding are concentrated to form S_+, while all secret bits are extracted during the 2nd-layer embedding to form S_-.

We continue the example shown in Fig.2 to illustrate the process of data extraction and image recovery. Assume one stego block R''_c = [115 53 54 114; 114 114 54 56; 52 114 54 55; 52 114 54 55; 54 113 115 55]. The maximum and the minimum of this stego block are 115 and 52, respectively. Due to $L_c \in \{min\{R''_{ij}\} + m_1\}$ and $H_c \in \{max\{R''_{ij}\} + m_2\}$, in this example, $L_c \in \{50, 51, \cdots, 55\}$ and $H_c \in \{112, 113, \cdots, 117\}$, where $m_1 \in \{-2, -1, \cdots, 3\}$ and $m_2 \in \{-3, -2, \cdots, 2\}$. All the combinations of (H_c, L_c) construct the solution space of (H_c, L_c) which satisfies the conditions that $N_{u,c} = 4$ and the pixels unequal to H_c, L_c, $H_c + 1$, $L_c - 1$ are not in the same row. For simplicity, we take $L_c = 55$ and $H_c = 114$ as an example to explain the extraction process. Since $L_c = 55$, $L_c - 1 = 54$. In contrast, $H_c + 1 = 115$ when $H_c = 114$. For the first row of R''_c, except 54, 114 and 115, only 53 equal to $L_c - 2$ satisfies the requirement. Similarly, 56 equal to $L_c + 1$ in the second row, 52 equal to $L_c - 3$ in the third row, and 113 equal to $H_c - 1$ in the fourth row are selected. Therefore, the total number of pixels unequal to H_c, $H_c + 1$, L_c, $L_c - 1$ is 4, i.e., $N_{u,c} = 4$, implying that the combination of ($L_c = 55$, $H_c = 114$) is only one solution satisfying the condition of $H_c - L_c > T_h$. Finally,

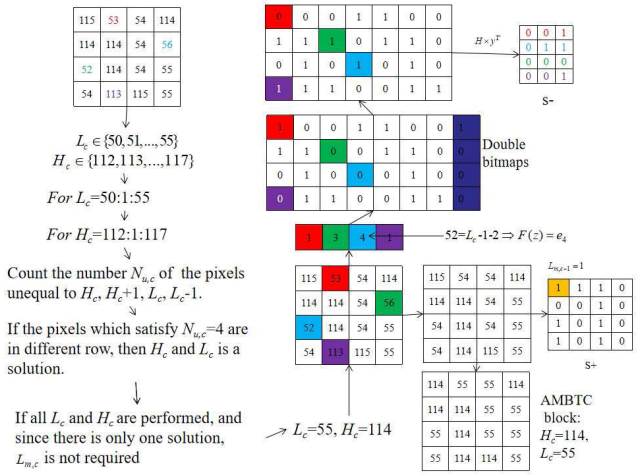


FIGURE 2. An example of data extraction and AMBTC image recovery phase.

after each possible combination of (L_c, H_c) is traversed, H_c and L_c can be determined if the number of solutions is one.

By Eq. (20), we know that the changed pixel of the first row is 53. According to the same manner, the changed pixels of the 2nd, 3th and 4th rows are 56, 52 and 113, respectively. Therefore, four changed positions 1, 3, 4 and 1 can be obtained. Then, the corresponding bits of the double bitmaps is changed based on Eq. (17). Next, S_- are extracted by the formula $s = H \times y^T$, and $S_+ = (110001010101010)_2$ can be easily obtained via Eq. (20). Assume $L_{m,c-1} = 1$, and thus, the number of hidden bits in S_+ is $16 - 1 = 15$. Finally, according to Eq. (20), the reconstructed block $R_{c,j} = [114, 55, 55, 114;\ 114, 114, 55, 55;\ 55, 114, 55, 55;\ 55, 114, 114, 55]$ is obtained.

IV. EXPERIMENTAL RESULTS

In this section, we performed a series of experiments and analysed to demonstrate the performance of our proposed method in embedding capacity and visual quality of stego images. All of the experiments were implemented in MATLAB R2014b on a PC with Intel® Core (TM) i7-8750H CPU @2.20 GHz, 16 GB RAM. The 15 classic grayscale images with sizes of 512 × 512 shown in Fig. 3, i.e., "Lena", "F16", "Barbara", "Goldhill", "Wine", "Bird", "Zelda","Boat", "Baboon","Peppers", "Man", "House", "Couple", "Lake" and "Elaine", are selected from the USC-SIPI data [36] and served as the test images. The secret data S_+ and S_- are randomly generated by a pseudo random number. The two factors including peak signal to noise radio (PSNR) and pure hiding capacity (H_P) are used to evaluate the performance between our scheme and compared methods. PSNR is defined as follows:

$$\text{PSNR} = 10\log_{10}\left(\frac{255^2}{\text{MSE}}\right), \tag{21}$$

TABLE 2. Performance of our proposed scheme.

Image	PSNR_A	PSNR_O	$H_P + L_S$	L_S	H_P
Lena	33.20	33.03	426189	1353	424836
F16	31.95	31.83	395617	1742	393875
Barbara	29.37	29.29	436954	1242	435712
Goldhill	32.84	32.66	445372	1348	444024
Wine	32.49	32.36	383758	1429	382329
Bird	29.99	29.90	457808	1361	456447
Zelda	36.65	36.26	436750	1334	435416
Boat	31.15	31.03	448537	1235	447302
Baboon	26.98	26.93	458104	1154	456950
Peppers	33.39	33.19	442930	1328	441602
Man	30.22	30.13	437886	1158	436728
House	30.97	30.87	412324	1439	410885
Couple	31.25	31.13	449533	1389	448144
Lake	30.19	30.09	441616	1349	440267
Elaine	33.88	33.65	450387	1126	449261
Average	31.63	31.49	434917	1332	433585

where

$$\text{MSE} = \frac{1}{H_I \times W_I}\sum_{i=1}^{H_I}\sum_{j=1}^{W_I}(I_{i,j} - R''_{i,j})^2, \tag{22}$$

$I_{i,j}$ and $R''_{i,j}$ are the pixels located at the (i, j) of the original image I and AMBTC compressed image/stego image R'' after two-layer embedding, respectively.

Besides H_P, the bit per bit (bpp) (i.e., the ratio of H_P to the size of a cover image in Eq. (23) is also used to measure the amount of the hidden bits in a cover image.

$$\text{bpp} = \frac{H_P}{W_I \times H_I}. \tag{23}$$

A. THE PERFORMANCE OF OUR PROPOSED SCHEME

The performances of our proposed scheme for 15 test images are shown in Table 2 when $T_h = 4$ is set, which includes PSNR_A (i.e., the PSNR between the original image I and the AMBTC-compressed image R), PSNR_O (i.e., the PSNR between the original image I and the stego AMBTC-compressed image R''), the total hiding capacity (i.e., H_P plus L_m) provided by an AMBTC-compressed image, L_m and H_P, where $L_m = |\sum_{i=1}^{N} L_{m,i}|$ represents the sum of all location bits.

From Table 2, we know that PSNR_O is very close to PSNR_A. The average difference between PSNR_A and PSNR_O is $31.63 - 31.49 = 0.14$ dB, implying that our data embedding operation maintains satisfactory image quality. With respect to the hiding capacity, since L_m must be required in our scheme, larger L_m means smaller H_P since the total hiding capacity for an AMBTC-compressed image is fixed. L_m ranges from 1126 to 1742, with the average of 1332, that is to say, only eight percent of N blocks are required to be marked while more than ninety-two percent of N blocks can be restored without the need of L_m, where

(a) Lena (b) Airplane (c) Barbara (d) Goldhill

(e) Wine (f) Bird (g) Zelda (h) Boat

(i) Baboon (j) Peppers (k) Man (l) House

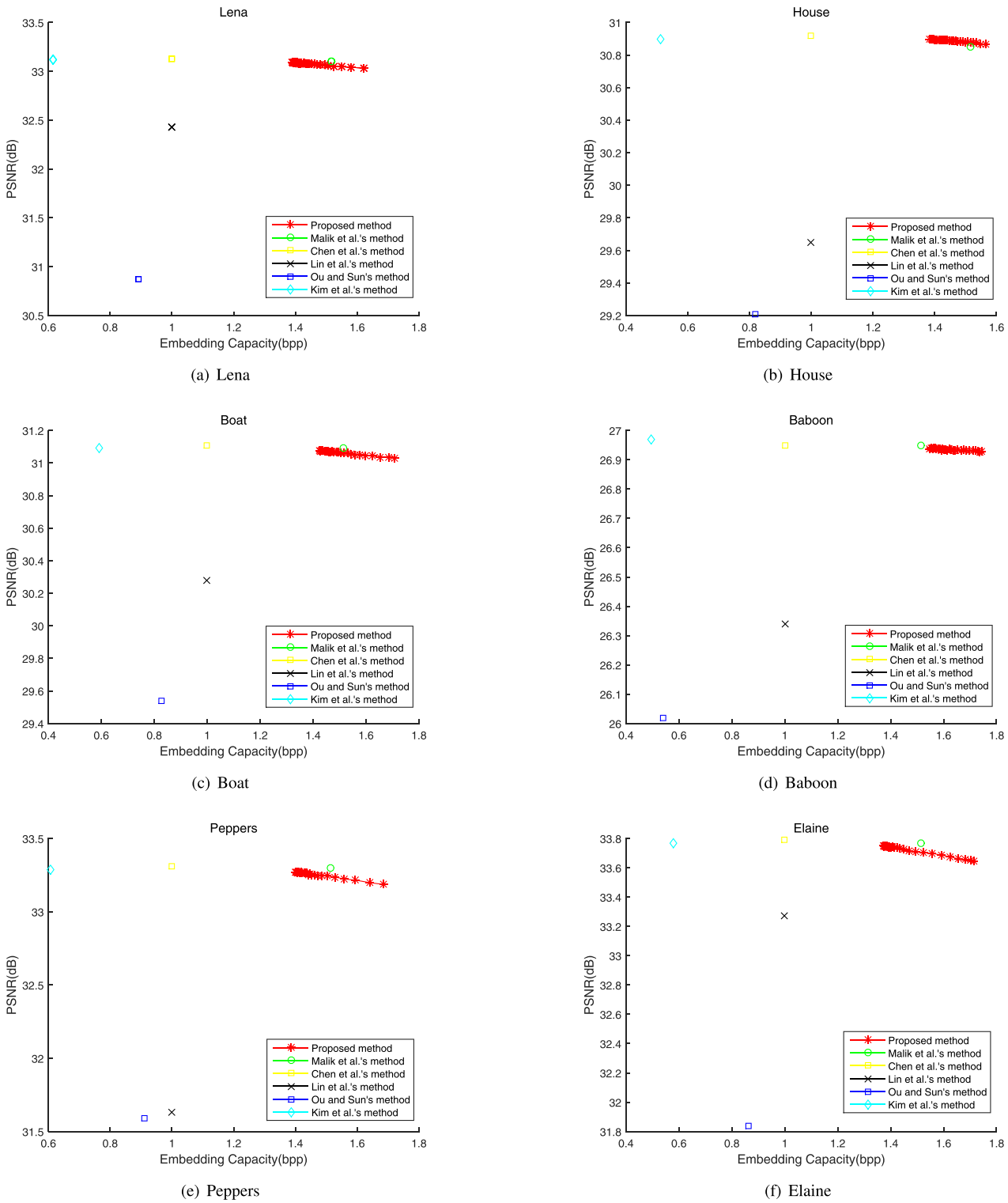
(m) Couple (n) Lake (o) Elaine

FIGURE 3. 15 test images.

$N = 512 \times 512/(4 \times 4) = 16384$. For individual images like F16, House and Wine, they have relatively low H_P due to their local complexity. While all other images have the H_P of over 420,000 bits. One of the most representative image is ''Baboon'', because it has the richest textures. it has the highest H_P of 456,950 bits.

B. PERFORMANCE COMPARISON

To further illustrate the performance of the proposed scheme, five AMBTC-based hiding schemes are used to compare with the proposed scheme, which includes Malik *et al.*'s method [32], Chen *et al.*'s method [37], Lin *et al.*'s method [19], Ou and Sun's method [14] and Kim *et al.*'s method [38]. Fig. 4 shows performance comparison under different T_h between our scheme and five compared schemes for six test images including Lena, House, Boat, Baboon, Peppers and Elaine, where $T_h \in \{4, 5, \ldots, 30\}$. Table 3 gives the capacity comparison of six compared methods under approximately the same PSNR. Fig. 5 shows the performance comparison between our method and four compared methods for the BOSS data set, which contains 10,000 images [39].



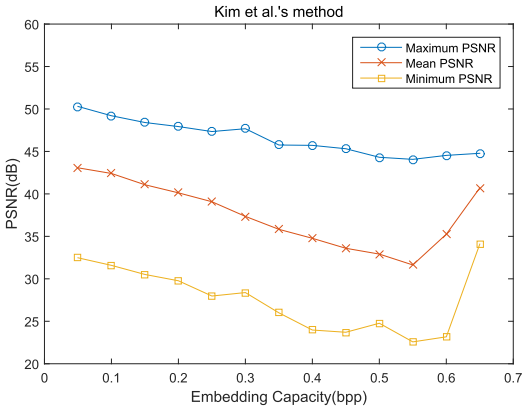
(a) Lena (b) House

(c) Boat (d) Baboon

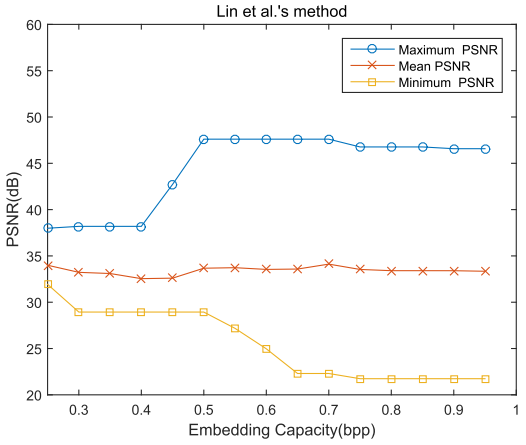
(e) Peppers (f) Elaine

FIGURE 4. **Performance comparisons between our scheme and five compared schemes for six test images.**

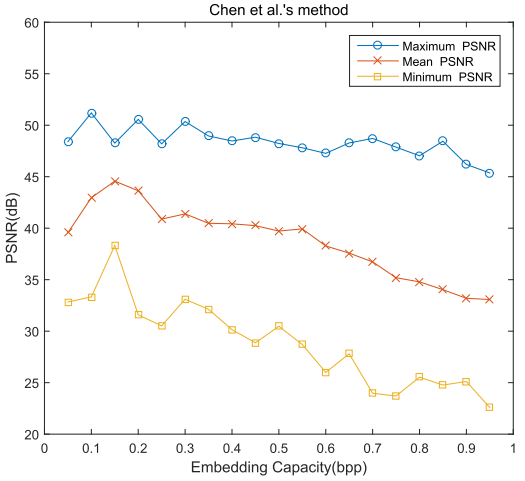
Similar to our 1st-layer embedding, both Chen *et al.*'s and Lin *et al.*'s methods can embed 1 bit into each pixel. Specifically, one pixel valued H_i is embedded with 1-bit by remaining changed or being increased by 1, while one pixel valued L_i is embedded with 1-bit by remaining changed or being decreased by 1. Different from



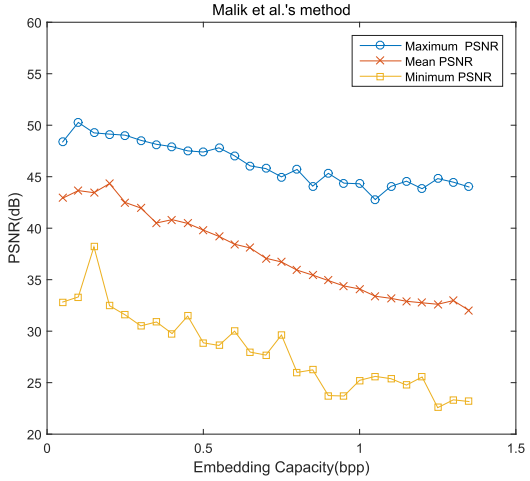
(a) Kim et al.'s method ($\overline{H_P} = 0.48(bpp), \overline{PSNR} = 34.35(dB)$)



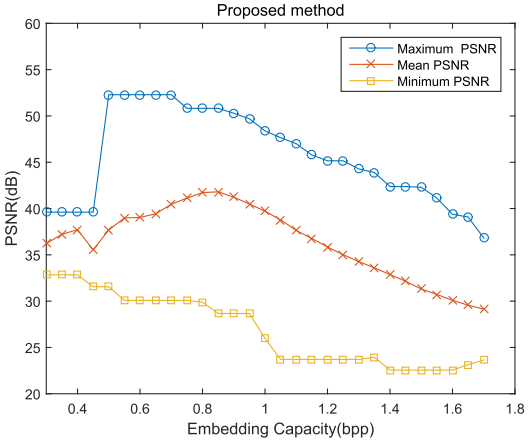
(b) Lin et al.'s method ($\overline{H_P} = 0.98(bpp), \overline{PSNR} = 33.41(dB)$)



(c) Chen et al.'s method ($\overline{H_P} = 0.87(bpp), \overline{PSNR} = 34.37(dB)$)



(d) Malik et al.'s method ($\overline{H_P} = 1.06(bpp), \overline{PSNR} = 34.32(dB)$)



(e) Proposed method ($\overline{H_P} = 1.44(bpp), \overline{PSNR} = 34.23(dB)$)

FIGURE 5. **Performance comparisons between our method and four compared methods for the BOSS data set.**

Lin *et al.*'s method, Chen *et al.*'s method replaces H_i with $AVG + VAR$ and L_i with $AVG - VAR$, where AVG and VAR represent the mean value and standard deviation of an AMBTC-compressed block, respectively. Due to the replacement of H_i with $AVG + VAR$ (and L_i with $AVG - VAR$), the PSNR induced by Chen *et al.*'s method is lower than

TABLE 3. Performance of the proposed scheme with the PSNR and the H_P by comparing with other four schemes for 6 test images.

Method	Factor	Lena	House	Boat	Baboon	Peppers	Elaine
Proposed	H_P	424,836	412,324	447,302	456950	441602	449261
method	PSNR	33.03	30.87	31.03	26.93	33.19	33.65
Malik et al.'s	H_P	397348	397187	397380	397105	397057	397098
method [30]	PSNR	33.10	30.85	31.09	26.95	33.30	33.77
Chen *et al.*'s	H_P	262128	261760	262128	262128	262144	261408
method [35]	PSNR	33.13	30.92	31.11	26.95	33.31	33.79
Lin *et al.*'s	H_P	262128	261760	262128	262128	262144	261408
method [17]	PSNR	32.43	29.65	30.28	26.34	31.63	33.27
Ou and Sun's	H_P	234004	214609	217264	141919	238969	226549
method [13]	PSNR	30.87	29.21	29.54	26.02	31.59	31.84
Kim *et al.*'s	H_P	161485	134568	155492	129801	159416	151867
method [36]	PSNR	33.12	30.90	31.09	26.97	33.29	33.77

that of Lin *et al.*'s method. Table 3 indicates that under the same H_P, the PSNR values of Chen *et al.*'s method are better than that of Lin *et al.*'s method for all test images. Since the single layer embedding is adopted in Chen *et al.*'s and Lin *et al.*'s methods, the H_P approaches to 1 bpp. Kim *et al.*'s method achieves data embedment by HS for AMBTC-compressed images. In Kim *et al.*'s method, for a block, all the pixels are classified into two classes, where one class corresponds to the '0' of B_i while the other class corresponds to the '1'. The number of pixels in each class is calculated, and then the class having a larger number of pixels is used for data embedding. Since only a part of all the pixels in a block are used for data embedding, their method can only obtain the H_P of greater than 0.5 bpp and less than 1 bpp. Considering that pixels are modified at most by 1, Kim *et al.*'s method can achieve the highest PSNR among all six compared methods. It can be clearly seen from Table 3, for Ou and Sun's method, the H_P of approximately 1 bpp can be achieved but the lowest PSNRs are obtained for all test images. Compared with the other five compared methods, Malik *et al.*'s method achieved the highest PSNR under the same H_P. This is because each embeddable pixel is modified at most by 1 (i.e., ± 1) when carrying $log_2 3$ bits.

From Fig. 4, it can be seen that the red '*' represents 27 results of the proposed scheme under different T_h ranging from 4 to 30. $T_h = 4$ is the right-most red '*' and $T_h = 30$ is the left-most red '*'. As T_h increases, the PSNR gets larger but the H_P becomes smaller. Fig. 5 is used to show performance comparisons among five compared methods in the BOSS date set, where the X-axis represents the embedding capacity, and the Y-axis indicates the PSNR value. Due to that the BOSS date set contains 10,000 test images, the maximum, mean and minimum values of all PSNR values for a given payload are illustrated in Fig. 5. The average $\overline{H_P}$ and $\overline{PSNR}$ of 10,000 images are calculated for each of five compared methods. Fig. 5 shows that $\overline{H_P} = 1.44$(bpp), $\overline{PSNR} = 34.23$(dB) for our proposed scheme, while $\overline{H_P} = 1.04$(bpp), $\overline{PSNR} = 34.32$(dB) for Malik *et al.*'s method. By the way of Fig. 5, it is concluded that our method can achieve higher payload under almost the same PSNR as Malik *et al.*'s method. This is because we adopt the two-layer data hiding strategy combining (7,4) Hamming code. Depending on two-layer embedding, each embeddable block can be embedded with 22 or 28 bits. In contrast, in Malik *et al.*'s method, each embeddable block can carry about 22 bits. By making full use of the advantage of (7,4) Hamming code, i.e., modifying one bit of one pixel to carry 3 secret message at most, our method can maintain satisfactory image quality.

V. CONCLUSION

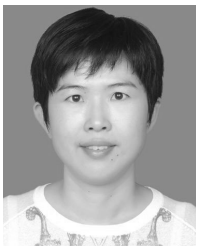
In this paper, we present a two-layer RDH scheme in combination with (7, 4) Hamming code. In the 1st-layer embedding, each pixel of one AMBTC-compressed image block can be embedded with 1 bit. That is to say, each block can embed 16 bits. The 2nd-layer embedding is applied with the aim of further increasing embedding capacity as much as possible on the basis of maintaining the visual quality. In the 2nd-layer embedding, by making full use of the advantage of (7,4) Hamming code, each block can further be embedded with 6 bits or 12 bits. Experimental results also demonstrate our method can achieve higher payloads under almost the same PSNR, compared with five compared methods.

REFERENCES

[1] S.-L. Li, K.-C. Leung, L. Cheng, and C.-K. Chan, "A novel image-hiding scheme based on block difference," *Pattern Recognit.*, vol. 39, no. 6, pp. 1168–1176, Jun. 2006.
[2] W. Hong, T.-S. Chen, and J. Chen, "Reversible data hiding using Delaunay triangulation and selective embedment," *Inf. Sci.*, vol. 308, pp. 140–154, Jul. 2015.
[3] C.-K. Chan and L. Cheng, "Hiding data in images by simple LSB substitution," *Pattern Recognit.*, vol. 37, no. 3, pp. 469–474, Mar. 2004.
[4] D. Zhang, Z. Pan, and H. Li, "A contour-based semi-fragile image watermarking algorithm in DWT domain," in *Proc. 2nd Int. Workshop Educ. Technol. Comput. Sci.*, vol. 3, 2010, pp. 228–231.
[5] X. Wu and W. Sun, "Robust copyright protection scheme for digital images using overlapping DCT and SVD," *Appl. Soft Comput.*, vol. 13, no. 2, pp. 1170–1182, Feb. 2013.
[6] R. M. Gray, "Vector quantization," *IEEE ASSP Mag.*, vol. 1, no. 2, pp. 4–29, Apr. 1984.

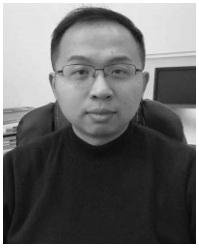
[7] T. Kim, "Side match and overlap match vector quantizers for images," *IEEE Trans. Image Process.*, vol. 1, no. 2, pp. 170–185, Apr. 1992.
[8] K. Wang, Z.-M. Lu, and Y.-J. Hu, "A high capacity lossless data hiding scheme for JPEG images," *J. Syst. Softw.*, vol. 86, no. 7, pp. 1965–1975, Jul. 2013.
[9] E. Delp and O. Mitchell, "Image compression using block truncation coding," *IEEE Trans. Commun.*, vol. COMM-27, no. 9, pp. 1335–1342, Sep. 1979.
[10] M. Lema and O. Mitchell, "Absolute moment block truncation coding and its application to color images," *IEEE Trans. Commun.*, vol. COMM-32, no. 10, pp. 1148–1157, Oct. 1984.
[11] R. Kumar, D.-S. Kim, and K.-H. Jung, "Enhanced AMBTC based data hiding method using hamming distance and pixel value differencing," *J. Inf. Secur. Appl.*, vol. 47, pp. 94–103, Aug. 2019.
[12] R. Kumar and K.-H. Jung, "A systematic survey on block truncation coding based data hiding techniques," *Multimedia Tools Appl.*, vol. 78, no. 22, pp. 32239–32259, Nov. 2019.
[13] J.-C. Chuang and C.-C. Chang, "Using a simple and fast image compression algorithm to hide secret information," *Int. J. Comput. Appl.*, vol. 28, no. 4, pp. 329–333, Jan. 2006.
[14] D. Ou and W. Sun, "High payload image steganography with minimum distortion based on absolute moment block truncation coding," *Multimedia Tools Appl.*, vol. 74, no. 21, pp. 9117–9139, Nov. 2015.
[15] Y.-H. Huang, C.-C. Chang, and Y.-H. Chen, "Hybrid secret hiding schemes based on absolute moment block truncation coding," *Multimedia Tools Appl.*, vol. 76, no. 5, pp. 6159–6174, Mar. 2017.
[16] Y.-Y. Chen and K.-Y. Chi, "Cloud image watermarking: High quality data hiding and blind decoding scheme based on block truncation coding," *Multimedia Syst.*, vol. 25, no. 5, pp. 551–563, Oct. 2019.
[17] R. Kumar, N. Kumar, and K.-H. Jung, "A new data hiding method using adaptive quantization & dynamic bit plane based AMBTC," in *Proc. 6th Int. Conf. Signal Process. Integr. Netw. (SPIN)*, Mar. 2019, pp. 854–858.
[18] C.-H. Li, Z.-M. Lu, and Y.-X. Su, "Reversible data hiding for Btc-compressed images based on bitplane flipping and histogram shifting of mean tables," *Inf. Technol. J.*, vol. 10, no. 7, pp. 1421–1426, Jul. 2011.
[19] C.-C. Lin and X.-L. Liu, "A Reversible data hiding scheme for block truncation compressions based on histogram modification," in *Proc. 6th Int. Conf. Genet. Evol. Comput.*, Aug. 2012, pp. 157–160.
[20] I.-C. Chang, Y.-C. Hu, W.-L. Chen, and C.-C. Lo, "High capacity reversible data hiding scheme based on residual histogram shifting for block truncation coding," *Signal Process.*, vol. 108, pp. 376–388, Mar. 2015.
[21] F. Li, K. Bharanitharan, C.-C. Chang, and Q. Mao, "Bi-stretch reversible data hiding algorithm for absolute moment block truncation coding compressed images," *Multimedia Tools Appl.*, vol. 75, no. 23, pp. 16153–16171, Dec. 2016.
[22] C.-C. Lin, C.-C. Chang, and Z.-M. Wang, "Reversible data hiding scheme using adaptive block truncation coding based on an edge-based quantization approach," *Symmetry*, vol. 11, no. 6, p. 765, Jun. 2019.
[23] K. Wang, Y. Hu, and Z.-M. Lu, "Reversible data hiding for block truncation coding compressed images based on prediction-error expansion," in *Proc. 8th Int. Conf. Intell. Inf. Hiding Multimedia Signal Process.*, Jul. 2012, pp. 317–320.
[24] W. Sun, Z.-M. Lu, Y.-C. Wen, F.-X. Yu, and R.-J. Shen, "High performance reversible data hiding for block truncation coding compressed images," *Signal Image Video Process.*, vol. 7, no. 2, pp. 297–306, Mar. 2013.
[25] W. Hong, Y.-B. Ma, H.-C. Wu, and T.-S. Chen, "An efficient reversible data hiding method for AMBTC compressed images," *Multimedia Tools Appl.*, vol. 76, no. 4, pp. 5441–5460, Feb. 2017.
[26] Y.-Y. Tsai, C.-S. Chan, C.-L. Liu, and B.-R. Su, "A reversible steganographic algorithm for BTC-compressed images based on difference expansion and median edge detector," *Imag. Sci. J.*, vol. 62, no. 1, pp. 48–55, Jan. 2014.
[27] C.-C. Chang, T.-S. Chen, Y.-K. Wang, and Y. Liu, "A reversible data hiding scheme based on absolute moment block truncation coding compression using exclusive OR operator," *Multimedia Tools Appl.*, vol. 77, no. 7, pp. 9039–9053, Apr. 2018.
[28] W. Hong, "Efficient data hiding based on block truncation coding using pixel pair matching technique," *Symmetry*, vol. 10, no. 2, p. 36, Jan. 2018.
[29] W. Hong, X. Zhou, and S. Weng, "Joint adaptive coding and reversible data hiding for AMBTC compressed images," *Symmetry*, vol. 10, no. 7, p. 254, Jul. 2018.
[30] C.-C. Lin, X.-L. Liu, W.-L. Tai, and S.-M. Yuan, "A novel reversible data hiding scheme based on AMBTC compression technique," *Multimedia Tools Appl.*, vol. 74, no. 11, pp. 3823–3842, Jun. 2015.
[31] J. Pan, W. Li, and C. C. Lin, "Novel reversible data hiding scheme for ambtc-compressed images by reference matrix," in *Proc. MISNC*, 2014, pp. 427–436.
[32] A. Malik, G. Sikka, and H. K. Verma, "An AMBTC compression based data hiding scheme using pixel value adjusting strategy," *Multimedia Syst. Signal Process.*, vol. 29, no. 4, pp. 1801–1818, Oct. 2018.
[33] C.-C. Chang, T. D. Kieu, and Y.-C. Chou, "A high payload steganographic scheme based on (7,4) Hamming code for digital images," in *Proc. Int. Symp. Electron. Commerce Secur.*, Aug. 2008, pp. 16–21.
[34] R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J.*, vol. 29, no. 2, pp. 147–160, Apr. 1950.
[35] J. Fridrich and D. Soukal, "Matrix embedding for large payloads," *IEEE Trans. Inf. Forensics Security*, vol. 1, no. 3, pp. 390–395, Sep. 2006.
[36] *The USC-SIPI Image Database*. Accessed: Nov. 9, 1977. [Online]. Available: http//sipi.usc.edu/database
[37] Y.-Y. Chen, C.-H. Hsia, S. Y. Jhong, and H.-J. Lin, "Data hiding method for AMBTC compressed images," *J. Ambient Intell. Hum. Comput.*, pp. 1–9, 2018.
[38] C. Kim, D. Shin, L. Leng, and C.-N. Yang, "Lossless data hiding for absolute moment block truncation coding using histogram modification," *J. Real-Time Image Process.*, vol. 14, no. 1, pp. 101–114, Jan. 2018.
[39] P. Bas, T. Filler, and T. Pevný, "'Break our steganographic system': The ins and outs of organizing BOSS," in *Information Hiding*, T. Filler, T. Pevný, S. Craver, and A. Ker, Eds. Berlin, Germany: Springer, 2011, pp. 59–70.

JUAN LIN received the B.S. degree from Jimei University, in 2004, and the M.S. and Ph.D. degrees from Osaka University, Japan, in 2008 and 2011, respectively. He is currently an Associate Professor with the Fuqing Branch of Fujian Normal University, China. His interesting researches are reversible data hiding and signal processing.



SHAOWEI WENG (Member, IEEE) received the Ph.D. degree from the Institute of Information Science, Beijing Jiaotong University, in July 2009. She is currently a Professor with the School of Information Science and Engineering, Fujian University of Technology. She publishes more than 30 articles and applies five national patents. Her research interests include image processing, data hiding and digital watermarking, pattern recognitions, and computer vision. She is also in charge of two Natural Science Foundation of China (NSFC) Project.

TIANCONG ZHANG received the B.S. degree from North China Electric Power University, in 2002. Until 2020, he has been with Guangdong Guohua Yuedian Taishan Power Generation Company Ltd., Jiangmen, China. He is currently an Engineer with the School of Information Science and Engineering, Fujian University of Technology. His research interests include electrical engineering, image processing, and data hiding.



BO OU received the B.S. and Ph.D. degrees from Beijing Jiaotong University, Beijing, China, in 2008 and 2014, respectively. Since 2014, he has been with the Faculty of College of Computer Science and Electronic Engineering, Hunan University, China, where he is currently an Associate Professor. His research interests include image processing and data hiding.

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. He is currently a Guest Professor with Hangzhou Danzi University. Prior to joining Feng Chia University, he was an Associate Professor with Chiao Tung University, a Professor in National Chung Hsing University, and the Chair Professor with National Chung Cheng University. He had also been a Visiting Researcher and a Visiting Scientist with Tokyo University and Kyoto University, Japan. During his service in Chung Cheng, he served as the Chairman of the Institute of Computer Science and Information Engineering, Dean of College of Engineering, Provost and then Acting President of Chung Cheng University, and the Director of Advisory Office in Ministry of Education, Taiwan. His current research interests include database design, computer cryptography, image compression, and data structures. He is currently a Fellow of IEE, U.K. He has received many research awards and honorary positions by and in prestigious organizations both nationally and internationally. Since his early years of career development, he consecutively won Outstanding Talent in Information Sciences of the R. O. C., the AceR Dragon Award of the Ten Most Outstanding Talents, Outstanding Scholar Award of the R. O. C., the Outstanding Engineering Professor Award of the R. O. C., the Distinguished Research Awards of National Science Council of the R. O.C., the Top Fifteen Scholars in Systems and Software Engineering of the *Journal of Systems and Software*. On numerous occasions, he was invited to serve as a Visiting Professor, the Chair Professor, the Honorary Professor, the Honorary Director, the Honorary Chairman, a Distinguished Alumnus, a Distinguished Researcher, and a Research Fellow by universities and research institutes.