Journal of Information Hiding and Multimedia Signal Processing ©2019 ISSN 2073-4212
Ubiquitous International Volume **10**, Number **2**, March **2019**

Reversible Data Hiding in Encrypted Images Based on Bit-plane Block Embedding

Jiang-Yi Lin

School of Computer and Information Engineering
Xiamen University of Technology
600 Ligong Road, Xiamen 361024, China;
Department of Information Engineering and Computer Science
Feng Chia University
100 Wenhua Road, Taichung 40724, Taiwan
jy1982chrimer@gmail.com

Yu Chen

School of Information Science and Engineering
Fujian University of Technology
33 Xuefu South Road, Fuzhou 350118, China
cheny@fjut.edu.cn

Chin-Chen Chang

Department of Information Engineering and Computer Science
Feng Chia University
100 Wenhua Road, Taichung 40724, Taiwan
alan3c@gmail.com

Yu-Chen Hu

Department of Computer Science and Information Management
Providence University
200, Section 7, Taiwan Boulevard, Taichung 43301, Taiwan
ychu@pu.edu.tw

Received April 2018; revised August 2018

Abstract. In this paper, we design a bit-plane block embedding (BPBE) algorithm to hide secret messages in binary images. Meanwhile, we proceed to apply BPBE for reversible data hiding in encrypted images. It embeds the part of least-significant-bit (LSB) planes into higher most-significant-bit (MSB) planes using BPBE for reserving room before encryption, so that additional data can be embedded into the LSB planes of encrypted images. If the receiver has the data hiding key only, he/she can extract the additional data but doesn't know about the exact information of the original image. And if the receiver is provided with the encryption key only, he/she can reconstruct the original image. When both keys are acquired, the data extraction and image recovery can be correctly completed. Experimental results illustrate that the proposed scheme can achieve a higher embedding rate (ER) compared to some state-of-the-art methods, and maintain an acceptable image quality.
Keywords: Bit-plane block embedding (BPBE); reversible data hiding (RDH); image encryption

1. **Introduction.**
The popularization and development of the Internet bring in the exploding of the data, which are in various forms like texts, images, audios and videos. Secret data can be either transmitted directly to the receiver or be embedded in the digital media before data transmission. Therefore, the protection of the secret data becomes an important issue.

A variety of methods and technologies have been developed to protect the secret data and to avoid being attacked. Among them, cryptography is a traditional method for data protection. After the encryption, the image becomes meaningless and will attract malicious attackers attention. Another technology for communication security is called reversible data hiding (RDH), which can imperceptibly alter digital images to embed secret information. When the receiver acquires the image, he/she can extract the secret messages and recover the image to its exact original state.

The RDH technique is widely applied in a variety of special scientific fields [1, 2, 3, 6, 7, 8, 9, 11, 13, 14, 15, 16, 17, 18], such as medical imagery, military imagery and law enforcement, where no distortion of original image is allowed. Two major approaches utilized in RDH are histogram shifting (HS) and difference expansion (DE). HS-based RDH was first introduced by Ni *et al.* [6]. In this method, a histogram was generated by the original pixel values. After the peak bin and its corresponding zero bin were selected, the other bins between them were shifted at one towards the zero bin to create one vacant histogram bin for data embedding. In order to improve the embedding rate (ER) of HS, other methods [2, 3, 14] utilized the prediction-errors to generate a sharper histogram. Sachnev *et al.* [11] employed a sorting technique by the local variance of prediction errors to achieve a better performance. Besides HS, DE is another primary approach for RDH. Tian divided the image into pixel pairs and calculated the difference value of each pair [13]. After expanding, by multiplying these difference values by 2, the least significant bit (LSB) became zero and thus could be utilized for data embedding. Alattar extended Tians method from pixel pairs to arbitrary size of pixel block [1]. Thodi *et al.* [15] firstly introduced the prediction-error expansion, which was widely adopted in RDH works.

Recently, driven by the needs of cloud computing and privacy protection, the research of reversible data hiding in encrypted images (RDH-EI) has received an increasing attention. According to the timing of room vacating, we can roughly classify RDH-EI into two categories: vacating room after encryption (VRAE) and vacating room before encryption (VRBE) [12].

In VRAE framework, the original image is encrypted directly, and the data-hider vacates room from the encrypted image for secret data embedding. Zhang first encrypted the original image to generate an encrypted image [20]. Then, the data-hider divided the encrypted image into a number of blocks. Each block was utilized to embed 1 secret bit by flipping 3 LSBs from half of the pixels. Hong *et al.* [4] improved Zhang's method [20] through considering the relation in neighboring blocks, and meanwhile utilized a special algorithm, which was called side-match, to gain a higher ER. The method proposed by Zhang [21] utilized a syndromes matrix called parity-check matrix to compress the encrypted image for reserving room to embed secret messages.

The methods in VRAE are trying hard to reserve room in encrypted images. However, as far as we know, the entropy of the image after encryption is achieving the maximum. As a result, the space reserved for embedding is rather small. The methods in VRBE, which reverse the order of encryption and reserve room by encrypting the image after room reserving, attempt to improve the ER. Ma *et al.* [5] divided the original image into two parts. The LSBs of the front part were embedded into the second part using a traditional RDH method. After that, the LSBs of the front part were reserving for data embedding. Zhang *et al.* [22] proposed an approximate method to reserve room in which some pixels are chosen and estimated by the remaining pixels surrounding them to acquire the predicted errors. After the pixel values are replaced by the predicted errors, a special encryption scheme is utilized to encrypt parts of the predicted errors to prevent the leakage of the information and the remaining part can be used for data hiding. Later on, Yi *et al.* [19] proposed an essential compression method called binary-block embedding (BBE) that reserves a large amount of room for data hiding. In this method, the original image is first decomposed to 8 bit-planes, and then the lower LSB bit planes are embedded into most-significant-bit (MSB) planes.

In this paper, we introduce the bit-plane block embedding (BPBE) algorithm to embed secret information in binary images. Meanwhile, we design a novel RDH-EI algorithm based on BPBE. In this RDH-EI algorithm, it first embeds part of the LSB bit planes of the original image into the selected MSB planes by using BPBE. Secondly, the content owner encrypts the image and sends it to the data hider. After acquiring the encrypted image, the data hider encrypts the secret messages and embeds them into LSB bit planes. Using the data hiding key and the encryption key respectively, the receiver can extract the secret messages and recover the original image. Hereby, the merits of this paper are listed as follows:

(1) We propose a novel algorithm called bit-plane block embedding (BPBE) for RDH. BPBE can be utilized in binary images and can be extended to the RDH for gray-scale images.

(2) ER can reach up to 0.2 bpp while the Peak Signal-to-Noise Ratio (PSNR) is still more than 30 dB. When given a small payload, the PSNR is even up to 50 dB.

The rest of this paper is organized as follows. In Section 2, the related work of VRBE methods are introduced. Section 3 illustrates the bit-plane block embedding. The detailed depiction of the proposed RDH-EI scheme can be found in Section 4, and followed by the analysis of experimental results in Section 5. In the end, some conclusions are drawn in Section 6.

2. **Related works.**

The essence of VRBE is to vacate room in the original images to embed secret information before encryption. Zhang *et al.* [22] proposed an RDH-EI algorithm based on histogram shifting where the histogram was generated by estimating values. In the first step, the content owner selected a small part of original image pixels though the encryption key which were marked as E, the other pixels were denoted as O. By examining the pixels belong to E, if the number of the surrounding O pixels was less than the threshold T, mark these pixels as $E1$, and the others as $E2$. Rearrange the pixels in original image to put $E1$ in the front, O in the last, as shown in Fig. 1. In the second step, pixels belong to $E1$ were estimated by its surrounding O pixels to generate predicted errors. These predicted errors were utilized to replace the $E1$ pixels. In order to avoid leakage of the distribution of information in $E1$, the content owner chose a part of $E1$ though the encryption key again, then encrypted these pixels to prevent the leakage. The last pixels which belong to $E2$ and O were all encrypted by benchmark encryption algorithm (e.g. AES). Finally, the encrypted image was generated. The data hider could embed the secret information into $E1$ by shifting the encrypted histogram of predicted errors without altering the pixels belonging to $E2$ and O. When the receiver acquired the marked encrypted images, data extraction and image recovery can be achieved by the data hiding key or/and the encryption key. The method in [22] can gain a higher PSNR, and it is suitable for small payload.

Yi *et al.* [19] developed a novel method in RDH-EI which was based on binary-block embedding (BBE). The BBE algorithm was an application of Huffman codes essentially. Given a binary image, divide it into non-overlapping $s \times s$ blocks, classify the blocks into

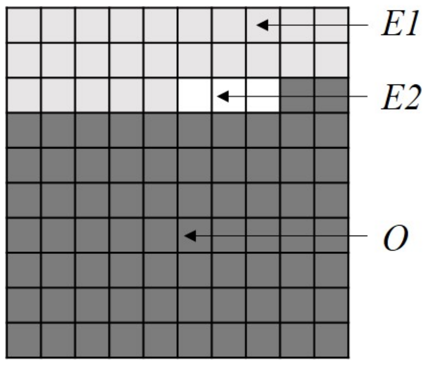

FIGURE 1. Example of the rearranged image

five categories with a threshold T, which is shown in Table 1. In each block, the notation $n1$ represents the count of 1, $n0$ represents the count of 0, and the notation m represents the minimal of them.

TABLE 1. Block description and block-labelling bits

Category	Condition	Block description	Block-labelling bits
1	m=$n0$=0	all pixels in block are 1	11
2	m=$n1$=0	all pixels in block are 0	10
3	m=$n0$, $m<T$	most pixels in block are 1	011
4	m=$n1$, $m<T$	most pixels in block are 0	010
5	$m \geq T$	cannot be used for embedding	00

All blocks could be utilized for data embedding except what belong to Category 5. In Categories 1 and 2, the block could be replaced by the labelling bits. The other bits in the block can be used for data embedding. For Categories 3 and 4, after labelling it with 3 labelling bits, parameter m and the position of m bits should be picked out and embedded in the block. An illustrative example is shown in Fig. 2. We should notice that, if the blocks belong to Category 5, the first two pixel values will be replaced by the labelling bits '00', thus, the first two original pixel values will be recorded for data extraction, as shown in Fig. 2(a).

In [19], the content owner decomposed the original image into 8 bit-planes. Pick out part of bits from LSB planes and embed these bits into the MSB planes using BBE. After encryption, the data hider can embed secret messages into these chosen positions by data hiding key. When acquired the marked encrypted image, the receiver can reconstruct the secret messages and recover the original image, respectively. This method can achieve a high capacity for the effectiveness of BBE, which is a compression in nature. However, the receiver either extracts the secret messages and recovers the original image or abandons the secret messages and reconstructs the original image directly. It has a drawback to obtain a marked image.

3. The Bit-plane Block Embedding (BPBE) Algorithm.

In this section, we propose a novel algorithm called bit-plane block embedding (BPBE) to embed messages into a binary image. Given a binary image I, divide it into a set of non-overlapping blocks with a size of $s \times s$ (s=3. It will be discussed in Section 5.1). The blocks can be summarized into five cases, which are listed as follow.

Case 1: The bit values in the block are all 1(or 0).

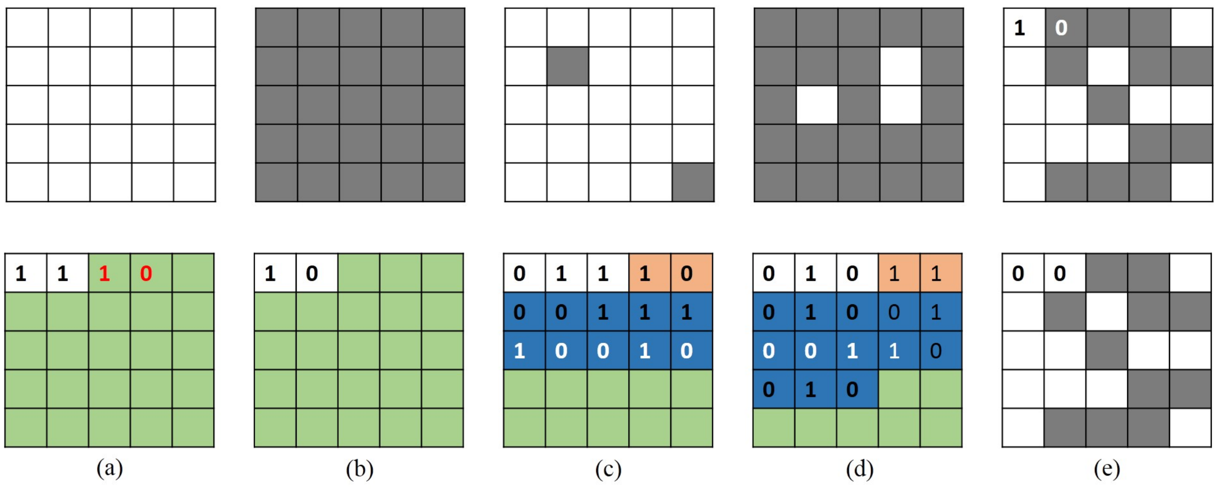

FIGURE 2. An example of BBE with 5×5 blocks and T=4. The white (or gray) boxes represent the pixels with values of 1(or 0). The green boxes represent the embedding areas. The orange, blue boxes mean the values of m and the corresponding positions of this original pixels, respectively. (a) and (b) are blocks belonging to Categories 1 and 2, which can be utilized for embedded 23 secret bits. (c) blocks belong to Category 3 with its embedding result. (d) blocks belong to Category 4 with its embedding result. (e) blocks belong to Category 5 which cannot be used for embedding

Case 2: Only one value in the block is 1(or 0), the others are opposite.
Case 3: Only two bit values in the block are 1(or 0), the others are opposite.
Case 4: Three bit values in the block are 1(or 0), the others are opposite.
Case 5: Four bit values in the block are 1(or 0), the others are opposite.

We utilize the blocks belonging to Case 1 and Case 2 for embedding, the other types of blocks remain unchanged. After the block is summarized, the number of blocks belonging to Case 1, Case 2 and Case 3 are accounted, which are denoted by n_1, n_2 and n_3, respectively. Each block belongs to Case 1 or Case 2 is used to embed 3 secret bits. As far as we know, 3 bit messages can represent 8 states that range from $(000)_2$ to $(111)_2$. Without the loss of generality, suppose that the bit values in this block are all 1 in Case 1. Along the sequences in raster-scan order (from top to bottom, left to right), we utilize the first 8 positions to embed secret messages, i.e., alter first bit 1 to indicate embedding message $(000)_2$, alter the eighth bit 1 to indicate embedding message $(111)_2$, as shown in Fig. 3(a). Considering the blocks belong to Case 2, there are also 8 same bits remaining (suppose only one value in the block is 0, the others are all 1). As a result, we can follow the similar idea to embed 3 secret bits as it does in blocks belonging to Case 1, as shown in Fig. 3(b).

It is worth noting that, although blocks belonging to Case 2 can be used for data embedding, there are two issues needed to be coped with. Firstly, the blocks belonging to Case 2 will conflict with Case 3 after embedding, thus confusing the decoder. Then, we define a bit stream v to distinguish between blocks belonging to Case 2 and Case 3. The length of v equals to n_2+n_3

$$v_i = \begin{cases} 1, & block_i \in Case\ 2 \\ 0, & block_i \in Case\ 3\ . \end{cases} \tag{1}$$

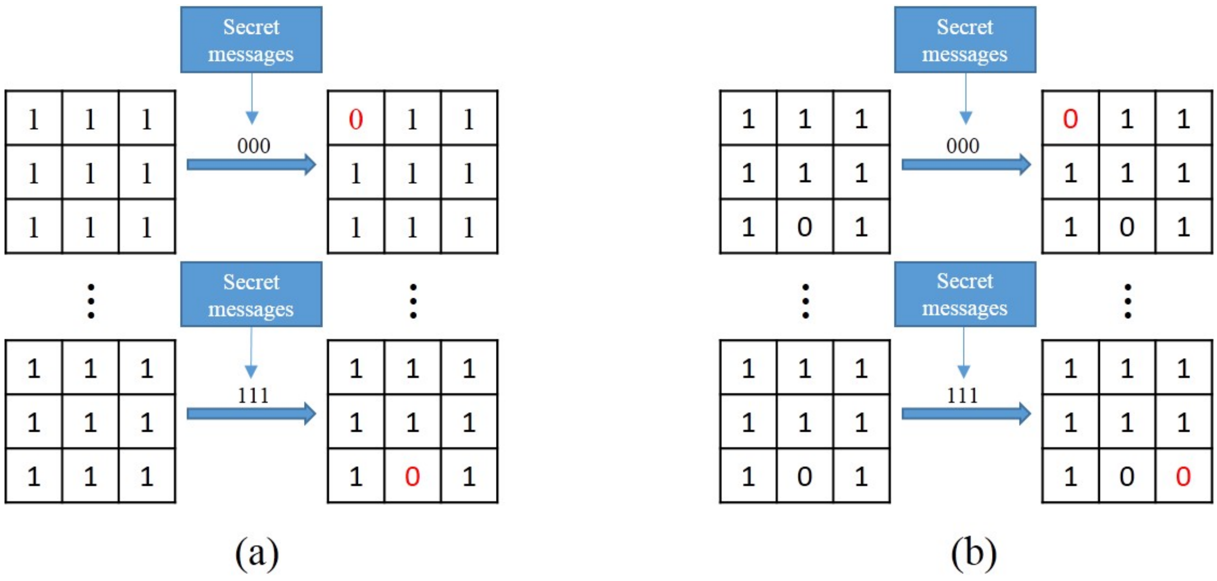

FIGURE 3. Example of the secret message embedding (a) Case 1 embedding, (b) Case 2 embedding

Secondly, how can decoder know which 0 is the embedding position after the blocks belonging to Case 2 finish embedding? To deal with this issue, we need another bit stream F to indicate which position of 0 is the embedding place. The size of bit stream F is n_2

$$F_i = \begin{cases} 1, & \textit{the first one is the embedding position} \\ 0, & \textit{the last one is the embedding position} \end{cases}. \tag{2}$$

At last, we can calculate the capacity C as follows:

$$C = 3 \times (n_1 + n_2) - |v| - |F| = 3 \times n_1 + n_2 - n_3 \ . \tag{3}$$

We add several bits at the end of bit stream v and F to indicate their length. For a 512×512 image, sixteen bits are enough to accommodate this information.

3.1. **Data embedding.** The payload p consists of three parts: M, v and F, where M denotes the secret messages, v denotes the bit stream which distinguishes blocks from Case 2 and Case 3, and F denotes the relative position relationship between the embedding position and the original bit value position of blocks belonging to Case 2. After the block is summarized, we can acquire the bit stream v. However, the values of the bit stream F depend on the secret messages M, since we embed the blocks belonging to Case 2 first. The detailed procedures of date embedding are provided as follows.

Data embedding procedure

Input: A binary image I and the binary secret messages M

Output: An embedded image I'

Step 1: Divide I into several non-overlapping blocks B according to the BPBE. Initialize v and F to be empty binary streams.

Step 2: Generate the bit stream v. For each block in B, if the block belongs to Case 2, then append $(1)_2$ to v. If the block belongs to Case 3, then append $(0)_2$ to v.

Step 3: Generate the bit stream F. For each block belonging to Case 2 in B, select first three bits from $M \cup v$, transfer the three bits to an integer l ($0 \leq l \leq 7$). If the only one bit value in block is 0 (or 1), alter the bit in the position of l to 0 (or 1). Examine the positions of original bit value of 0 (or 1) and l, if l is smaller, then append $(1)_2$ to F; otherwise append $(0)_2$ to F.

Step 4: For each block belonging to Case 1 in B, select the first three bits from $M \cup v \cup F$, transfer the three bits to an integer l $(0 \leq l \leq 7)$. If the bit values in block are all 1(or 0), alter the bit in the position of l to 0 (or 1).

Step 5: Output the embedded image I'.

3.2. **Data extraction and image recovering.** It is worth noticing that after data embedding, blocks belonging to Case 1 are all transferred to Case 2, while blocks belonging to Case 2 transfer to Case 3. Data extraction and image recovering are the reverse order of data embedding. The procedures are given as follows.

Data extraction and image recovering procedure

Input: An embedded image I'

Output: A binary image I and the binary secret messages M

Step 1: Divide I' into several non-overlapping blocks B according to the BPBE. Initialize p to be empty bit stream.

Step 2: For each block belonging to Case 2 in B, convert the position of only one value 1(or 0) into the corresponding binary form to get the secret bits. At the same time, we alter the bit value for recovering. Combine the binary forms of these position values, then part of secret messages M', v and F are gained, i.e., $p = M' \cup v \cup F$.

Step 3: For each block belonging to Case 3 in B, if the corresponding bit value in v equals to 0, that means this block is not used for embedding, so we continue to select the next block belonging to Case 3. If the corresponding bit value in v equals to 1, that means this block belongs to Case 2 originally, then we proceed to check out the corresponding bit value in F. If the bit value in F equals to 1, that means the first bit value 1 (or 0) is the embedding position. Otherwise, the second bit value 1 (or 0) is in the embedding position. Then we transfer the position of bit value 1 (or 0) into the corresponding binary form to get the secret bits. At the same time, we alter the bit value for recovering. Combine the binary forms of these position values and M' to gain the complete secret messages M and recover the original image.

Step 4: Output the original image and secret messages.

3.3. **Example of BPBE.** For a better understanding of the principle of BPBE, an example of omitting the bits indicating the lengths of v and F is illustrated in this section. Given a binary image with sized 6×9, divide it into 6 blocks $B = \{B_1, B_2, B_3, B_4, B_5, B_6\}$, in which there are three Case 1 blocks, two Case 2 blocks and one Case 3 block. Assume the secret message $M = (1100000010)_2$. According to the raster-scan order, bit stream $v = (110)_2$ which denotes that the B_2, B_4 are belonging to Case 2 and that B_6 is belonging to Case 3, as shown in Fig. 4(a).
The block B_2 is first utilized to embed 3 secret bits $(110)_2$, alter the corresponding position of these bits, i.e., 6, from $(0)_2$ to $(1)_2$. The embedding position is larger than the position of original bit value $(0)_2$, hence the bit stream $F = (0)_2$. Follow the similar principle, the first position of B_4 is changed from $(0)_2$ to $(1)_2$ for embedding 3 secret bits $(000)_2$, and the bit stream F becomes $(01)_2$, which is shown in Fig. 4(b).

After embedding the secret messages into the blocks belonging to Case 2, BPBE then starts to utilize blocks belonging to Case 1 for embedding. That is to say, block B_1 embed the secret bits $(001)_2$ and alter the second position of original bit value from $(0)_2$ to $(1)_2$. Blocks B_3 and B_5 can be used to embed the last 6 secret bits, which is shown in Fig. 4(c).

When the decoder obtains the marked image, as shown in Fig. 4(f), he/she starts by Case 2 extraction. As demonstrated in Fig. 4(e), the decoder alters the unique different

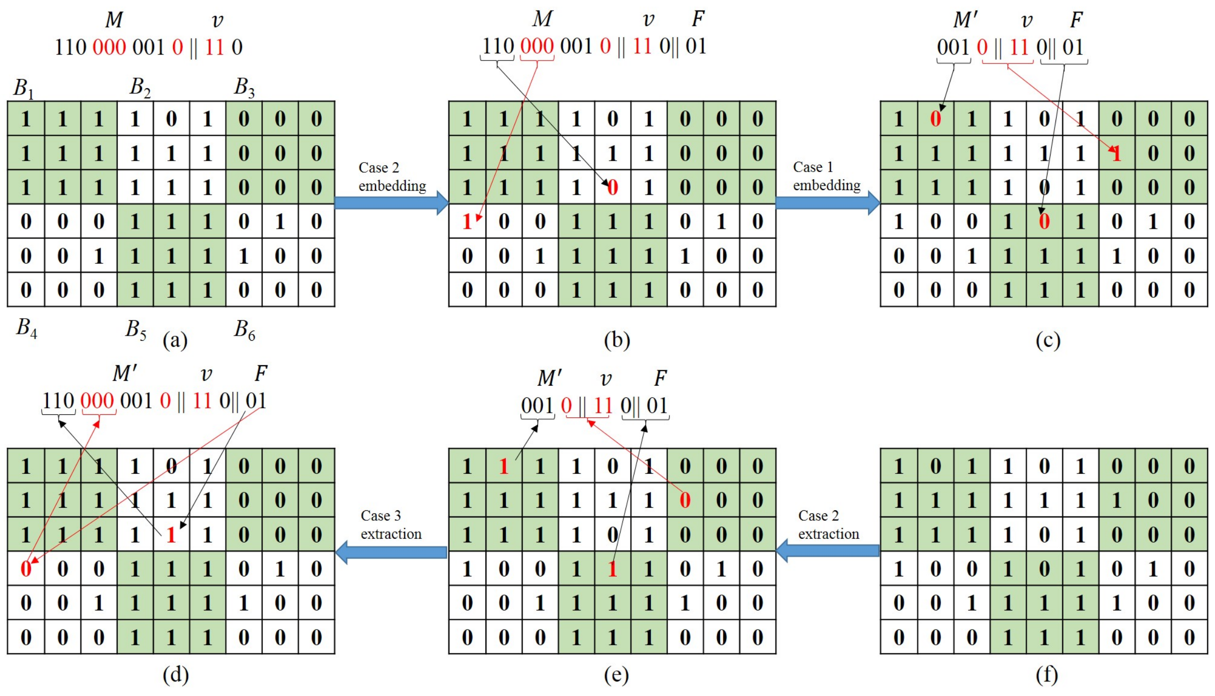

FIGURE 4. Example of embedding and extraction (a) original image and secret messages, (b) Case 2 embedding result, (c) Case 1 embedding result, (d) extract the original and get the secret messages, (e) Case 2 extraction result, (f) marked image

bit value and generates the bit stream $M' \cup v \cup F$ according to its corresponding position. The following step is to deal with the block belonging to Case 3. According to bit stream v, the decoder can distinguish which block belonging to Case 3 is the embedding block. Meanwhile, he/she can make a decision on which position is the embedding place with the help of bit stream F. For example, block B_2 belongs to Case 3, according to v, its corresponding bit value is $(1)_2$, which means that it is an embedding block. At the same time, its corresponding bit value in F is $(0)_2$, which means that the last $(0)_2$ is the embedding place. We can extract secret message $(110)_2$ and alter the second bit value from $(0)_2$ to $(1)_2$, as shown in Fig. 4(d).

4. Reversible data hiding in encrypted images based on BPBE.

In this section, we propose a novel reversible data hiding algorithm to embed secret messages in encrypted image based on BPBE, as shown in Fig. 5. The proposed method is composed of three primary steps: (1) reserving room and generation of the encrypted images, (2) secret data embedding and generation of the marked encrypted images, and (3) data extraction and image recovery. These steps are elaborated as follows.

4.1. Reserving room and generation of the encrypted images.

In this phase, the content owner started with applying BPBE algorithm to reserve room in the original images for data hiding. To prevent the leakage of the raw data, the content owner encrypted the original images after room reserving and sent it to the data hider.

Assume that a cover image I is able to embed secret data. As far as we know, each pixel in a gray-scale image is in the range of [0, 255], since we can decompose the cover image I into 8 bit-planes, which denote as bp_i ($1 \le i \le 8$). Here bp_1 is the MSB plane and bp_8 is the LSB plane. We calculate the capacity of each plane, and denote the result as C. If $C>0$, the plane is unable to embed secret data. In our proposed scheme, we select the

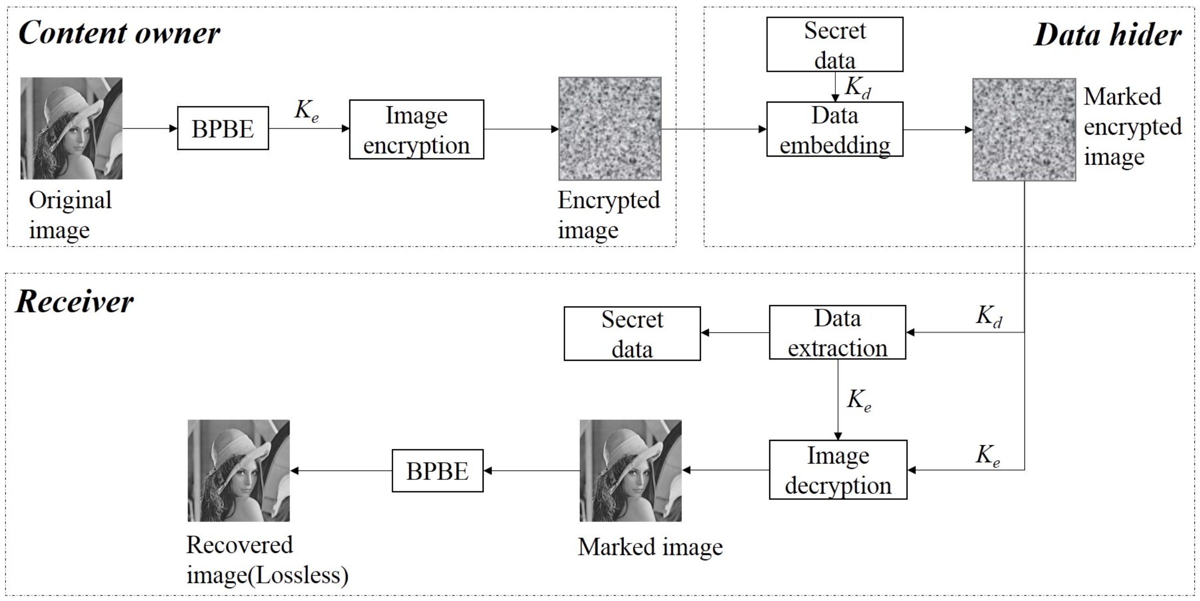

FIGURE 5. The framework of the proposed scheme based on BPBE

planes bp_4, bp_5, bp_6 for data embedding, calculate their capacity and denote the results as C_4, C_5 and C_6, respectively (We will discuss this in Section 5.2). Notation C_{sum} represents the sum of them. After that, we embed the first C_{sum} bits of LSB plane bp_8 into the chosen MSB planes by BPBE. The first C_{sum} bits in LSB plane bp_8 are reserved for data hider to embed secret messages. After the self-embedding in original image, we can encrypt it to generate an encrypted image, denoted as E, through the encryption key K_e and sent it to data hider.

In order to inform the data hider how many secret bits can be replaced in E, we use the first m positions in LSB plane of E to represent the length, e.g. m=16. Meanwhile, for the convenience of recovering the original image, we use another 3-bit n to represent an uncertainty that whether the selected MSB plane can be utilized for self-embedding or not, e.g. $(100)_2$ indicates that only bit plane bp_4 is enable for self-embedding.

4.2. Secret data embedding and generation of the marked encrypted images.

Once receiving the encrypted image E, the data hider decodes the first m bits of the LSB to acquire how many bits he can modify. After that, the data hider encrypts the secret data according to the data hiding key K_d, then embeds this message into LSB of E, although the data hider does not know what exactly the original image is. Finally, the marked encrypted image, denoted as ME, is generated.

4.3. Data extraction and image recovery.

Data extraction and image recovery can be carried out depending on whether the receiver has the data hiding key and/or encrypted key or not.

. Data extraction

Once the receiver acquires the data hiding key K_d, he/she extracts the first m bits of the LSB of the marked encrypted image ME to decode the length of the secret information. Secondly, he retrieves the secret messages through the length according to the LSB of ME. Finally, the source of the secret data is gained by using K_d.

. Image recovery

In the situation where the receiver has the encrypted key K_e only, he/she can extract the first m+n bits of the LSB of the ME at the beginning to decide how many bit planes can be used for self-embedding and the value of C_{sum}. In the following step, he/she performs image decryption by K_e directly and keep the secret data in the decrypted image. After that, a marked image is acquired. According to the value of n and C_{sum}, the receiver utilizes the BPBE algorithm to recover the original image without any error. If the receiver has the K_d and K_e at the same time, he can perform the data extraction and image recovery completely.

5. **Experimental results.** The proposed scheme is running on a personal computer whose operation system is Windows 10. Its CPU is Intel Xeon E3-1225 v5, 3.3GHz, and the memory of this computer is 8GB. The images utilized for experiment are selected from SIPI database [10], the size of each grayscale image is of 512×512 pixels. Before comparing our scheme to prior methods, we will first discuss the size of each block and the selection of MSB bit planes.

5.1. **The size of each block.** According to Eq. (3), when the block size is s=3, the capacity of a bit plane depends on the values of n_1, n_2 and n_3. Assume the block size becomes s=2. We can only utilize each block belonging to Case 1 to embed 2 secret bits. After data embedding, they will conflict with the blocks belonging to Case 2. As a result, we need the bit stream v to distinguish between blocks belonging to Case 1 and Case 2. Finally, the capacity of a bit plane is calculated by

$$C = 2 \times n_1 - |v| = n_1 - n_2 \ . \tag{4}$$

Follow the similar principle, when the block size changes to s=4, we can only utilize each block belonging to Case 1 to embed 4 secret bits. The capacity of a bit plane is calculated by

$$C = 4 \times n_1 - |v| = 3 \times n_1 - n_2 \ . \tag{5}$$

Table 2 shows the value s and its corresponding capacity of a bit plane in images *lena*, *airplane*, *barbara* and *baboon*. It is demonstrated in Table 2 that the block size which equals to s=3 is superior to the block sizes that equals to s=2 and s=4 in measure of ER. Another finding is that the BPBE algorithm has advantage in ER for smooth images, i.e., the smoother is the image, the higher is the ER.

5.2. **The selection of MSB bit planes.** After secret messages embedding, the marked image quality is another focus in reversible data hiding. The higher PSNR indicates the better image quality. Table 3 illustrates the bit planes selected for self-embedding and its corresponding PSNR in given payload. It is demonstrated in Table 3 and it is quite clear that, for the same image, the higher MSB bit plane is used to self-embedding, the lower the PSNR will be under certain ER. Even the higher MSB bit plane can embed more data, we select the planes bp_4, bp_5, and bp_6 for self-embedding which can achieve an acceptable PSNR.

5.3. **Compare with other related methods.** In this section, we proceed to compare the comprehensive performances between the proposed scheme and some related methods. We also select 4 traditional images *lena*, *airplane*, *barbara* and *baboon* for experiment. Calculate the PSNR according to the given certain ER, which is shown in Fig. 6.

As show in Figs. 6(a), 6(b) and 6(c), the proposed scheme is superior to the methods in [20], [4] and [21] that belong to VRAE measurement through ER and PSNR in smooth images. Although the method in [22] gain higher PSNR, our scheme has the advantage with its ER up to more than twice compared with it. It is a pity that, according to the characteristic of BPBE, our scheme is inferior to the methods in [20], [4] and [21]

TABLE 2. The value s and the corresponding capacity

Images	Bit Plane	Block Size	n_1	n_2	n_3	C(bits)
lena	bp_4	s=2	32388	20679	–	11709
		s=3	8248	4470	4456	**24758**
		s=4	3108	1599	–	7725
	bp_5	s=2	18882	28416	–	–
		s=3	2526	3691	5249	**6091**
		s=4	591	797	–	976
	bp_6	s=2	10183	32012	–	–
		s=3	322	1473	4477	–
		s=4	36	68	–	**40**
airplane	bp_4	s=2	39135	16394	–	22741
		s=3	12218	3532	3620	**36566**
		s=4	5516	1365	–	15183
	bp_5	s=2	27434	23341	–	4093
		s=3	6367	3693	4652	**18142**
		s=4	2440	1120	–	6200
	bp_6	s=2	16001	29553	–	–
		s=3	1986	2623	4866	**3715**
		s=4	601	465	–	1338
barbara	bp_4	s=2	25341	23914	–	1427
		s=3	5819	3375	4379	**16453**
		s=4	2127	1065	–	5316
	bp_5	s=2	15813	29561	–	–
		s=3	1936	2854	4880	**3782**
		s=4	419	554	–	703
	bp_6	s=2	9211	32490	–	–
		s=3	202	1395	4436	–
		s=4	3	40	–	–
baboon	bp_4	s=2	13498	30808	–	–
		s=3	1244	2286	4859	**1159**
		s=4	233	392	–	307
	bp_5	s=2	8947	32507	–	–
		s=3	178	1264	4179	–
		s=4	1	25	–	–
	bp_6	s=2	8293	32417	–	–
		s=3	103	1007	4103	–
		s=4	0	5	–	–

measurement by ER in rough image baboon, though it has an advantage in PSNR. The method in [22] has a complete edge in both ER and PSNR compared with our scheme, as shown in Fig. 6(d).

It can be discovered that, there is one plummet occurring in Figs. 6(a) and 6(c), while two plummets in Fig. 6(b) and Fig. 6(d) has no plummet. This is because two bit planes are utilized in *lena* and *barbara* for self-embedding, while three bit planes are utilized in *airplane* and only one bit plane is used in *baboon*.

TABLE 3. Bit plane(s) and their corresponding PSNR under certain ER

Images	Bit Plane(s)	PSNR (dB)
lena(ER=0.1bpp)	bp_1	20.07
	bp_2	26.27
	bp_3	31.88
	$bp_4+bp_5+bp_6$	37.14
airplane(ER=0.2bpp)	bp_1	20.88
	bp_2	26.37
	bp_3	32.18
	$bp_4+bp_5+bp_6$	36.96
barbara(ER=0.07bpp)	bp_1	21.74
	bp_2	27.36
	bp_3	32.88
	$bp_4+bp_5+bp_6$	38.40
baboon(ER=0.004bpp)	bp_1	24.14
	bp_2	29.53
	bp_3	36.04
	$bp_4+bp_5+bp_6$	44.02

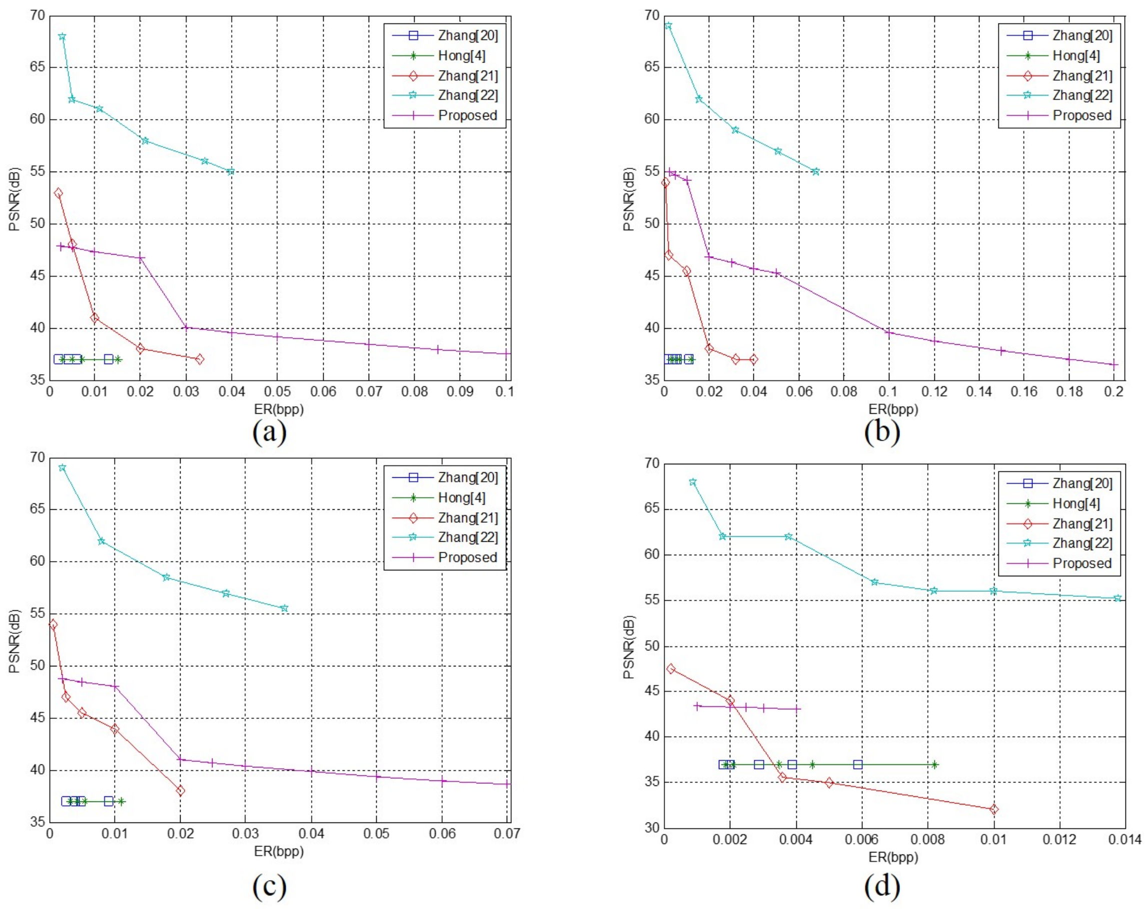

FIGURE 6. Performance comparison with other methods in different images. (a)*lena*, (b)*airplane*, (c)*barbara*, (d)*baboon*

6. **Conclusions.** In this paper, a novel reversible data hiding algorithm, bit-plane block embedding (BPBE), is proposed for binary images. According to the gray-scale images can be decomposed into bit planes, BPBE can be utilized for reserving room to embed secret messages in RDH-EI before encryption. Experimental results show that the proposed scheme is superior to methods belonging to VRAE concerning ER and PSNR. Especially, our scheme can obtain a higher ER in smoothing images.

Acknowledgment. This work is partially supported by the Natural Science Foundation of Fujian Province (2018J01572).

REFERENCES

[1] A. Alattar, Reversible watermark using the difference expansion of a generalized integer transform, *IEEE Transactions on Image Processing*, vol. 13, no. 8, pp. 1147-1156, 2004.

[2] M. Fallahpour, Reversible image data hiding based on gradient adjusted prediction, *IEICE Electronics Express*, vol. 5, no. 20, pp. 870-876, 2008.

[3] W. Hong, T. S. Chen, and C. W. Shiu, Reversible data hiding for high quality images using modification of prediction errors, *The Journal of Systems and Software*, vol. 82, no. 11, pp. 1833-1842, 2009.

[4] W. Hong, T. S. Chen, and H. Y. Wu, An improved reversible data hiding in encrypted images using side match, *IEEE Signal Processing Letters*, vol. 19, no. 4, pp. 199-202, 2012.

[5] K. Ma, W. Zhang, X. Zhao, N. Yu, and F. Li, Reversible data hiding in encrypted images by reserving room before encryption, *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 3, pp. 553-562, 2013.

[6] Z. Ni, Y. Q. Shi, N. Ansari, and W. Su, Reversible data hiding, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 3, pp. 354-362, 2006.

[7] C. Qin, C. C. Chang, Y. H. Huang and L. T. Liao, An inpainting-assisted reversible steganographic scheme using a histogram shifting mechanism, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 7, pp. 1109-1118, 2013.

[8] C. Qin and X. P. Zhang, Effective reversible data hiding in encrypted image with privacy protection for image content, *Journal of Visual Communication and Image Representation*, vol. 31, pp. 154-164, 2015.

[9] C. Qin, W. Zhang, F. Cao, X. P. Zhang and C. C. Chang, Separable reversible data hiding in encrypted images via adaptive embedding strategy with block selection, *Signal Processing*, vol. 153, pp. 109-122, 2018.

[10] SIPI Image Database, [Online]. Available: http://sipi.usc.edu/database/

[11] V. Sachnev, H. J. Kim, J. Nam, S. Suresh, and Y. Q. Shi, Reversible watermarking algorithm using sorting and prediction, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 7, pp. 989-999, 2009.

[12] 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, 2016.

[13] J. Tian, Reversible data embedding using a difference expansion, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 890-896, 2003.

[14] P. Y. Tsai, Y. C. Hu, and H. L. Yeh, Reversible image hiding scheme using predictive coding and histogram shifting, *Signal Processing*, vol. 89, no. 6, pp. 1129-1143, 2009.

[15] D. M. Thodi and J. J. Rodriguez, Prediction-error based reversible watermarking, *in Proceedings of IEEE International Conference on Image Processing*, 2004.

[16] S. W. Weng, Y. Zhao, J. S. Pan and R. R. Ni, Reversible watermarking based on invariability and adjustment on pixel pairs, *Signal Processing Letters*, vol. 15, pp. 721-724, 2008.

[17] S. W. Weng, J. S. Pan and L. D. Li, Reversible data hiding based on an adaptive pixel-embedding strategy and two-layer embedding, *Information Sciences*, vol. 369, pp. 144-159, 2016.

[18] S. W. Weng, Y. J. Liu and J. S. Pan, Reversible data hiding based on flexible block-partition and adaptive block-modification strategy, *Journal of Visual Communication and Image Representation*, vol. 41, pp. 185-199, 2016.

[19] Sh. Yi, and Y. Zhou, Binary-block embedding for reversible data hiding in encrypted images, *Signal Processing*, vol. 133, pp. 40-51, 2017.

[20] X. Zhang, Reversible data hiding in encrypted images, *IEEE Signal Processing Letters*, vol. 18, no. 4, pp. 255-258, 2011.

[21] X. Zhang, Separable reversible data hiding in encrypted image, *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 826-832, 2012.

[22] W. Zhang, K. Ma, and N. Yu, Reversibility improved data hiding in encrypted images, *Signal Processing*, vol. 94, no. 1, pp. 118-127, 2014.